Mohamed Medhat Gaber (Ed.)

Scientific Data Mining and Knowledge Discovery

Principles and Foundations

Springer

Scientific Data Mining and Knowledge Discovery

“This page left intentionally blank.”

Mohamed Medhat Gaber
Editor

Scientific Data Mining and Knowledge Discovery

Principles and Foundations

Springer

Editor
Mohamed Medhat Gaber
Caulfield School of Information Technology
Monash University
900 Dandenong Rd.
Caulfield East, VIC 3145
Australia
mohamed.m.gaber@gmail.com

Color images of this book you can find on www.springer.com/978-3-642-02787-1

ISBN 978-3-642-02787-1 e-ISBN 978-3-642-02788-8
DOI 10.1007/978-3-642-02788-8
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2009931328

ACM Computing Classification (1998): I.5, I.2, G.3, H.3

Cover design: KuenkelLopka GmbH

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

This book is dedicated to:
My parents: Dr. Medhat Gaber and
Mrs. Mervat Hassan
My wife: Dr. Nesreen Hassaan
My children: Abdul-Rahman and Mariam

“This page left intentionally blank.”

Contents

Contributors

Mohamed Abouelhoda Cairo University, Orman, Gamaa Street, 12613 Al Jizah, Giza, Egypt Nile University, Cairo-Alex Desert Rd, Cairo 12677, Egypt

Charu C. Aggarwal IBM T. J. Watson Research Center, NY, USA, AL 35805, USA, charu@us.ibm.com

Ghazi Al-Naymat School of Information Technologies, The University of Sydney, Sydney, NSW 2006, Australia, ghazi@it.usyd.edu.au

Ahmed Bouferguene Campus Saint-Jean, University of Alberta, 8406, 91 Street, Edmonton, AB, Canada T6C 4G9

Mohamed Medhat Gaber Centre for Distributed Systems and Software Engineering, Monash University, 900 Dandenong Rd, Caul_eld East, VIC 3145, Australia, Mohamed.Gaber@infotech.monash.edu.au

John S. Gero Krasnow Institute for Advanced Study and Volgenau School of Information, Technology and Engineering, George Mason University, USA, john@johngero.com

Moustafa Ghanem Imperial College, South Kensington Campus, London SW7 2AZ, UK

Sara J. Graves University of Alabama in Huntsville, AL 35899, USA, sgraves@itsc.uah.edu

Stephan Grimm FZI Research Center for Information Technologies, University of Karlsruhe, Baden-Württemberg, Germany, grimm@fzi.de

Achim Hoffmann University of New South Wales, Sydney 2052, NSW, Australia

Nafaa Jabeur Department of Computer Science, Dhofar University, Salalah, Sultanate of Oman, nafaa_jabeur@du.edu.om

Shahjahan Khan Department of Mathematics and Computing, Australian Centre for Sustainable Catchments, University of Southern Queensland, Toowoomba, QLD, Australia, khans@usq.edu.au

Ashesh Mahidadia University of New South Wales, Sydney 2052, NSW, Australia

Wei Peng Platform Technologies Research Institute, School of Electrical and Computer, Engineering, RMIT University, Melbourne VIC 3001, Australia, w.peng@rmit.edu.au

Cara Stein University of Alabama in Huntsville, AL 35899, USA, cgall@itsc.uah.edu

Hassan Safouhi Campus Saint-Jean, University of Alberta, 8406, 91 Street, Edmonton, AB, Canada T6C 4G9

Nabil Sahli Department of Computer Science, Dhofar University, Salalah, Sultanate of Oman, nabil_sahli@du.edu.om

Steve Tanner University of Alabama in Huntsville, AL 35899, USA, stanner@itsc.uah.edu

Lu Wencong Shanghai University, 99 Shangda Road, BaoShan District, Shanghai, Peoples Republic of China, wclu@shu.edu.cn

Jon Williamson Kings College London, Strand, London WC2R 2LS, England, UK, j.williamson@kent.ac.uk

Introduction

Mohamed Medhat Gaber

"It is not my aim to surprise or shock you – but the simplest way I can summarise is to say that there are now in the world machines that think, that learn and that create. Moreover, their ability to do these things is going to increase rapidly until – in a visible future – the range of problems they can handle will be coextensive with the range to which the human mind has been applied" by Herbert A. Simon (1916-2001)

1 Overview

This book suits both graduate students and researchers with a focus on discovering knowledge from scientific data. The use of computational power for data analysis and knowledge discovery in scientific disciplines has found its roots with the revolution of high-performance computing systems. Computational science in physics, chemistry, and biology represents the first step towards automation of data analysis tasks. The rational behind the development of computational science in different areas was automating mathematical operations performed in those areas. There was no attention paid to the scientific discovery process. Automated Scientific Discovery (ASD) [1–3] represents the second natural step. ASD attempted to automate the process of theory discovery supported by studies in philosophy of science and cognitive sciences. Although early research articles have shown great successes, the area has not evolved due to many reasons. The most important reason was the lack of interaction between scientists and the automating systems.

With the evolution in data storage, large databases have stimulated researchers from many areas especially machine learning and statistics to adopt and develop new techniques for data analysis. This has led to a new area of data mining and knowledge discovery. Applications of data mining in scientific applications have

M.M. Gaber (✉)
Centre for Distributed Systems and Software Engineering,
Monash University, 900 Dandenong Rd, Caulfield East,
VIC 3145, Australia
e-mail: Mohamed.Gaber@infotech.monash.edu.au

M.M. Gaber (ed.), *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*, DOI 10.1007/978-3-642-02788-8_1,

been studied in many areas. The focus of data mining in this area was to analyze data to help understanding the nature of scientific datasets. Automation of the whole scientific discovery process has not been the focus of data mining research.

Statistical, computational, and machine learning tools have been used in the area of scientific data analysis. With the advances in Ontology and knowledge representation, ASD has great prospects in the future. In this book, we provide the reader with a complete view of the different tools used in analysis of data for scientific discovery. The book serves as a starting point for students and researchers interested in this area. We hope that the book represents an important step towards evolution of scientific data mining and automated scientific discovery.

2 Book Organization

The book is organized into four parts. Part I provides the reader with background of the disciplines that contributed to the scientific discovery. Hoffmann and Mahidadia provided a detailed introduction to the area of machine learning in Chapter *Machine Learning*. Chapter *Statistical Inference* by Khan gives the reader a clear start-up overview of the field of statistical inference. The relationship between scientific discovery and philosophy of science is provided by Williamson in Chapter *The Philosophy of Science and its Relation to Machine Learning*. Cognitive science and its relationship to the area of scientific discovery is detailed by Peng and Gero in Chapter *Concept Formation in Scientific Knowledge Discovery* from a Constructivist View. Finally, Part I is concluded with an overview of the area of Ontology and knowledge representation by Grimm in Chapter *Knowledge Representation and Ontologies*. This part is highly recommended for graduate students and researchers starting in the area of using data mining for discovering knowledge in scientific disciplines. It could also serve as excellent introductory materials for instructors teaching data mining and machine learning courses. The chapters are written by experts in their respective fields.

After providing the introductory materials in Part I, Part II provides the reader with computational methods used in the discovery of knowledge in three different fields. In Chapter *Spatial Techniques*, Jabeur and Sahli provide us with a chapter of the different computational techniques in the Geospatial area. Safouhi and Bouferguene in Chapter *Computational Chemistry* provide the reader with details on the area of computational chemistry. Finally, Part II is concluded by discussing the well-established area of bioinformatics outlining the different computational tools used in this area by Aboelhoda and Ghanem in chapter *String Mining in Bioinformatics*.

The use of data mining techniques to discover scientific knowledge is detailed in three chapters in Part III. Chapter *Knowledge Discovery and Reasoning in Geospatial Applications* by Sahli and Jabeur provides the reader with techniques used in reasoning and knowledge discovery for Geospatial applications. The second chapter in this part, Chapter *Data Mining and Discovery of Chemical Knowledge*, is written by Wencong providing the reader with different projects, detailing the results,

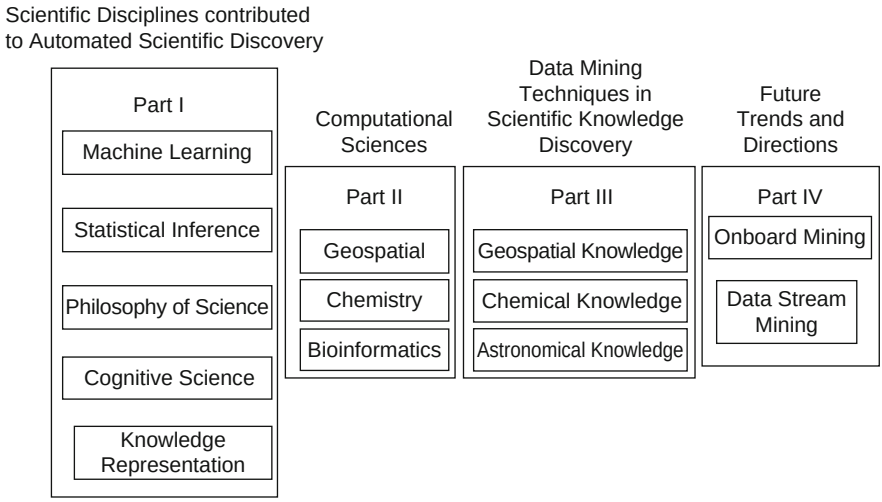


Fig. 1 Book Organization

of using data mining techniques to discover chemical knowledge. Finally, the last chapter of this part, Chapter *Data Mining and Discovery of Astronomical Knowledge*, by Al-Naymat provides us with a showcase of using data mining techniques to discover astronomical knowledge.

The book is concluded with a couple of chapters by eminent researchers in Part IV. This part represents future directions of using data mining techniques in the ares of scientific discovery. Chapter *On-Board Data Mining* by Tanner et al. provides us with different projects using the new area of onboard mining in spacecrafts. Aggarwal in Chapter *Data Streams: An Overview and Scientific Applications* provides an overview of the areas of data streams and pointers to applications in the area of scientific discovery.

The organization of this book follows a historical view starting by the well-established foundations and principles in Part I. This is followed by the traditional computational techniques in different scientific disciplines in Part II. This is followed by the core of this book of using data mining techniques in the process of discovering scientific knowledge in Part III. Finally, new trends and directions in automated scientific discovery are discussed in Part IV. This organization is depicted in Fig. 1

3 Final Remarks

The area of automated scientific discovery has a long history dated back to the 1980s when Langley et al. [3] have their book "Scientific Discovery: Computational Explorations of the Creative Processes" outlining early success stories in the area.

Although the research in this area has not been progressing as such in the 1990s and the new century, we believe that with the rise of areas of data mining and machine learning, the area of automated scientific discovery will witness an accelerated development.

The use of data mining techniques to discover scientific knowledge has recently witnessed notable successes in the area of biology [4] and with less impact in the area of chemistry [5], physics and astronomy [6]. The next decade will witness more success stories with discovering scientific knowledge automatically due to the large amounts of data available and the faster than ever production of scientific data.

References

1. R.E. Valdes-Perez, Knowl. Eng. Rev. **11**(1), 57–66 (1996)
2. P. Langley, Int. J. Hum. Comput. Stud. **53**, 393–410 (2000)
3. P. Langley, H.A. Simon, G.L. Bradshaw, J.M. Zytkow (1987) *Scientific Discovery: Computational Explorations of the Creative Processes* (MIT, Cambridge, MA)
4. J.T.L. Wang, M.J. Zaki, H.T.T. Toivonen, D. Shasha, in *Data Mining in Bioinformatics*, eds. by X. Wu, L. Jain. Advanced Information and Knowledge Processing (Springer London, 2005)
5. N. Chen, W. Lu, J. Yang, G. Li, *Support Vector Machine in Chemistry* (World Scientific Publishing, Singapore, 2005)
6. H. Karimabadi, T. Sipes, H. White, M. Marinucci, A. Dmitriev, J. Chao, J. Driscoll, N. Balac, J. Geophys. Res. **112**(A11) (2007)

Part I
Background

“This page left intentionally blank.”

Machine Learning

Achim Hoffmann and Ashesh Mahidadia

The purpose of this chapter is to present fundamental ideas and techniques of machine learning suitable for the field of this book, i.e., for automated scientific discovery. The chapter focuses on those symbolic machine learning methods, which produce results that are suitable to be interpreted and understood by humans. This is particularly important in the context of automated scientific discovery as the scientific theories to be produced by machines are usually meant to be interpreted by humans.

This chapter contains some of the most influential ideas and concepts in machine learning research to give the reader a basic insight into the field. After the introduction in Sect. 1, general ideas of how learning problems can be framed are given in Sect. 2. The section provides useful perspectives to better understand what learning algorithms actually do. Section 3 presents the Version space model which is an early learning algorithm as well as a conceptual framework, that provides important insight into the general mechanisms behind most learning algorithms. In section 4, a family of learning algorithms, the AQ family for learning classification rules is presented. The AQ family belongs to the early approaches in machine learning. The next, Sect. 5 presents the basic principles of decision tree learners. Decision tree learners belong to the most influential class of inductive learning algorithms today. Finally, a more recent group of learning systems are presented in Sect. 6, which learn relational concepts within the framework of logic programming. This is a particularly interesting group of learning systems since the framework allows also to incorporate background knowledge which may assist in generalisation. Section 7 discusses Association Rules – a technique that comes from the related field of Data mining. Section 8 presents the basic idea of the Naive Bayesian Classifier. While this is a very popular learning technique, the learning result is not well suited for human comprehension as it is essentially a large collection of probability values. In Sect. 9, we present a generic method for improving accuracy of a given learner by generating multiple classifiers using variations of the training data. While this works well in most cases, the resulting classifiers have significantly increased complexity

A. Hoffmann (✉)
University of New South Wales, Sydney 2052, NSW, Australia

M.M. Gaber (ed.), *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*, DOI 10.1007/978-3-642-02788-8_2,

and, hence, tend to destroy the human readability of the learning result that a single learner may produce. Section 10 contains a summary, mentions briefly other techniques not discussed in this chapter and presents outlook on the potential of machine learning in the future.

1 Introduction

Numerous approaches to learning have been developed for a large variety of possible applications. While learning for classification is prevailing, other learning tasks have been addressed as well which include tasks such as learning to control dynamic systems, general function approximation, prediction as well as learning to search more efficiently for a solution of combinatorial problems.

For different types of applications specialised algorithms have been developed. Although, in principle, most of the learning tasks can be reduced to each other. For example, a prediction problem can be reduced to a classification problem by defining classes for each of the possible predictions.[1] Equally, a classification problem can be reduced to a prediction problem, etc.

The Learner's Way of Interaction

Another aspect in learning is the way how a learning system interacts with its environment. A common setting is to provide the learning system with a number of classified training examples. Based on that information, the learner attempts to find a general classification rule which allows to classify correctly both, the given training examples as well as unseen objects of the population. Another setting, *unsupervised learning*, provides the learner only with unclassified objects. The task is to determine which objects belong to the same class. This is a much harder task for a learning algorithm than if classified objects are presented. Interactive learning systems have been developed, which allow interaction with the user while learning. This allows the learner to request further information in situations, where it seems to be needed. Further information can range from merely providing an extra classified or unclassified example randomly chosen to answering specific questions which have been generated by the learning system. The latter way allows the learner to acquire information in a very focused way. Some of the ILP systems in Sect. 6 are interactive learning systems.

[1] In prediction problems there is a sequence of values given, on which basis the next value of the sequence is to be predicted. The given sequence, however, may usually be of varying length. Opposed to that are many classification problems based on a standard representation of a fixed length. However, exceptions exist here as well.

Another more technical aspect concerns how the gathered information is internally processed and finally organised. According to that aspect the following types of representations are among the most frequently used for supervised learning of classifiers:

- Decision trees
- Classification rules (production rules) and decision lists
- PROLOG programs
- The structure and parameters of a neural network
- Instance-based learning (nearest neighbour classifiers etc.)[2]

In the following, the focus of the considerations will be on learning classification functions. A major part of the considerations, however, is applicable to a larger class of tasks, since many tasks can essentially be reduced to classification tasks. Although, the focus will be on *concept learning* which is a special case of classification learning, concept learning attempts to find representations which resemble in some way concepts humans may acquire. While it is fairly unclear, how humans actually do that, in the following we understand under *concept learning* the attempt to find a "comprehensible"[3] representation of a classification function.

2 General Preliminaries for Learning Concepts from Examples

In this section, a unified framework will be provided in which almost all learning systems fit in, including neural networks, that learn concepts, i.e. classifiers, from examples. The following components can be distinguished to characterise concept learning systems:

- A set of examples
- A learning algorithm
- A set of possible learning results, i.e. a set of concepts

Concerning the set of examples, it is an important issue to find a suitable *representation* for the examples. In fact, it has been recognised that the representation of examples may have a major impact on success or failure of learning.

[2] That means gathering a set of examples and a similarity function to determine the most similar example for a given new object. The most similar example is being used for determining the class of the presented object. Case-based reasoning is also a related technique of significant popularity, see e.g. [1, 2].

[3] Unfortunately, this term is also quite unclear. However, some types of representations are certainly more difficult to grasp for an average human than others. Foe example, cascaded linear threshold functions, as present in multi-layer perceptions, seem fairly difficult to comprehend, as opposed to, e.g., boolean formulas.

2.1 Representing Training Data

The representation of training data, i.e. of examples for learning concepts, has to serve two ends: On one hand, the representation has to suit the user of the learning system, in that it is easy to reflect the given data in the chosen representation form. On the other hand, the representation has to suit the learning algorithm. Suiting the learning algorithm again has at least two facets: Firstly, the learning algorithm has to be able to digest the representations of the data. Secondly, the learning algorithm has to be able to find a suitable concept, i.e. a useful and appropriate generalisation from the presented examples.

The most frequently used representation of data is some kind of attribute or feature vectors. That is, objects are described by a number of attributes.

The most commonly used kinds of attributes are one of the following:

- Unstructured attributes:
  - Boolean attributes i.e. either the object does have an attribute or it does not. Usually specified by the values $\{f, t\}$, or $\{0, 1\}$, or sometimes in the context of neural networks by $\{-1, 1\}$.
  - Discrete attributes, i.e. the attribute has a number of possible values (more then two), such as a number of colours $\{red, blue, green, brown\}$, shapes $\{circle, triangle, rectangle\}$, or even numbers where the values do not carry any meaning, or any other set of scalar values.
- Structured attributes, where the possible values have a presumably meaningful relation to each other:
  - Linear attributes. Usually the possible values of a linear attribute are a set of numbers, e.g. $\{0, 1, ..., 15\}$, where the ordering of the values is assumed to be relevant for generalisations. However, of course also non-numerical values could be used, where such an ordering is assumed to be meaningful. For example, colours may be ordered according to their brightness.
  - Continuous attributes. The values of these attributes are normally reals (with a certain precision) within a specified interval. Similarly as with linear attributes, the ordering of the values is assumed to be relevant for generalisations.
  - Tree-structured attributes. The values of these attributes are organised in a subsumption hierarchy. That is, for each value it is specified what other values it subsumes. This specification amounts to a tree-structured arrangement of the values. See 5 for an example.

Using attribute vectors of various types, it is fairly easy to represent objects of manifold nature. For example, cars can be described by features as colour, weight, height, length, width, maximal speed, etc.

2.2 Learning Algorithms

Details of various learning algorithms are given later in this chapter. However, generally speaking, we can say, that every learning algorithm searches implicitly or explicitly in a space of possible concepts for a concept that sufficiently fits the presented examples. By considering the set of concepts and their representations through which a learning algorithm is actually searching, the algorithm can be characterised and its suitability for a particular application can be assessed. Section 2.3 discusses how concepts can be represented.

2.3 Objects, Concepts and Concept Classes

Before discussing the representation of concepts, some remarks on their intended meaning should be made. In concept learning, concepts are generally understood to subsume a certain set of objects. Consequently, concepts can formally be described with respect to a given set of possible objects to be classified. The set of *possible objects* is defined by the kind of representation chosen for representing the examples. Considering for instance attribute vectors for describing objects, there is usually a much larger number of *possible objects* than the number of objects which may actually occur. This is due to the fact, that in the case of attribute vectors, the set of possible objects is simply given by the Cartesian product of the sets of allowed values for each of the attributes. That is, every combination of attribute values is allowed although, there may be no “pink elephants”, “green mice”, or “blue rabbits”.

However, formally speaking, for a given set of objects X, a concept c is defined by its extension in X, i.e. we can say c is simply a subset of X. That implies that for a set of n objects, i.e. for $|X| = n$ there are 2^n different concepts. However, most actual learning systems will not be able to learn all possible concepts. They will rather only be able to learn a certain subset. Those concepts which can potentially be learnt, are usually called the *concept class* or *concept space* of a learning system. In many contexts, concepts which can be learnt are also called *hypotheses* and *hypothesis space* respectively. Later, more formal definitions will be introduced. Also, in the rather practical considerations to machine learning a slightly different terminology is used than in the more mathematically oriented considerations.

However, in general it can be said that an actual learning system L, given n possible objects, works only on a particular subset of all the 2^n different possible concepts which is called the concept space C of L. For C, both of the following conditions hold:

1. For every concept $c \in C$ there exists training data, such that L will learn c.
2. For all possible training data, L will learn some concept c, such that $c \in C$. That is, L will never learn a concept $c \notin C$.

Considering a *set of concepts* there is the huge number of 2^{2^n} different sets of concepts on a set of n objects. To give a numerical impression: Looking at 30 boolean features describing the objects in X under consideration, would amount to $n = 2^{30} \approx 1000000000 = 10^9$ different possible objects. Thus, there exist $\approx 2^{1000000000}$ different possible concepts and $\approx 2^{2^{1000000000}} \approx 10^{10^{300000000}}$ different concept spaces, an astronomically large number.

Another characteristic of learning algorithms besides their concept space, is the particular order in which concepts are considered. That is, if two concepts are equally or almost equally confirmed by the training data, which of these two concepts will be learnt?

In Sect. 2.4, the two issues are treated in more detail to provide a view of learning which makes the similarities and dissimilarities among different algorithms more visible.

2.4 Consistent and Complete Concepts

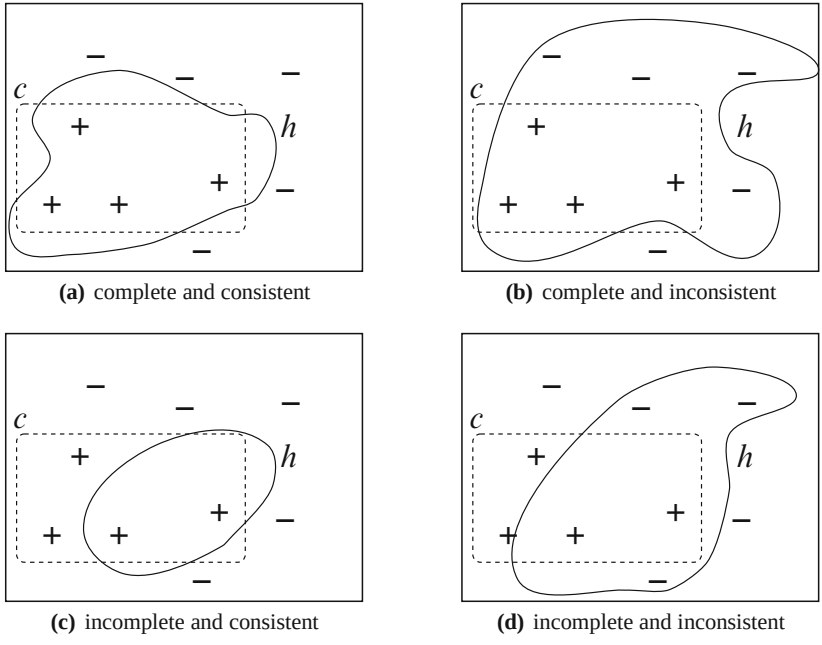
In machine learning some of the technical terms describing the relation between a hypothesis of how to classify objects and a set of classified objects (usually the training sample) are used differently in different contexts. In most mathematical/theoretical considerations a hypothesis h is called *consistent* with the training set of classified objects, if and only if the hypothesis h classifies all the given objects in the same way as given in the training set. A hypothesis h' is called *inconsistent* with a given training set if there is an object which is differently classified in the training set than by the hypothesis h'.

Opposed to that, the terminology following Michalski [3] considering concept learning assumes that there are only two classes of objects. One is the class of *positive* examples of a concept to be learned and the remaining objects are *negative* examples. A hypothesis h for a concept description is said to *cover* those objects which it classifies as positive examples. Following this perspective, it is said that a hypothesis h is *complete* if h covers all positive examples in a given training set. Further, a hypothesis h is said to be *consistent* if it does not cover any of the given negative examples. The possible relationships between a hypothesis and a given set of training data are shown in Fig. 1.

3 Generalisation as Search

In 1982, Mitchell introduced [4] the idea of the *version space*, which puts the process of generalisation into the framework of searching through a space of possible "versions" or concepts to find a suitable learning result.

The version space can be considered as the space of all concepts which are consistent with all learning examples presented so far. In other words, a learning



(a) complete and consistent

(b) complete and inconsistent

(c) incomplete and consistent

(d) incomplete and inconsistent

Fig. 1 The four possible relationships between a hypothesis and a set of classified examples. The correct concept c is shown as *dashed* line. The hypothesis h as *solid line*. A *consistent* hypothesis covers all positive examples. A *complete* hypothesis covers no negative example

algorithm considers initially, before any training data has been presented, the complete concept space as possible outcomes of the learning process. After examples are presented, this space of still possible outcomes of the learning process is gradually reduced.

Mitchell provided data structures which allow an elegant and efficient maintenance of the version space, i.e. of concepts that are consistent with the examples presented so far.

Example. To illustrate the idea, let us consider the following set of six geometrical objects *big square, big triangle, big circle, small square, small triangle,* and *small circle*, and abbreviated by $b.s$, $b.t$, ... , $s.t$, $s.c$, respectively. That is, let $X = \{b.s, b.t, b.c, s.s, s.t, s.c\}$.

And let the set of concepts C that are potentially output by a learning system L be given by

$$C = \{\{\}, \{b.s\}, \{b.t\}, \{b.c\}, \{s.s\}, \{s.t\}, \{s.c\}, \{b.s, b.t, b.s\}, \{s.s, s.t, s.s\}, \{b.s, s.s\}, \{b.t, s.t\}, \{b.c, s.c\}, X\}.$$

That is, C contains the empty set, the set X, all singletons and the abstraction of the single objects by relaxing one of the requirements of having a specific size or having a specific shape.

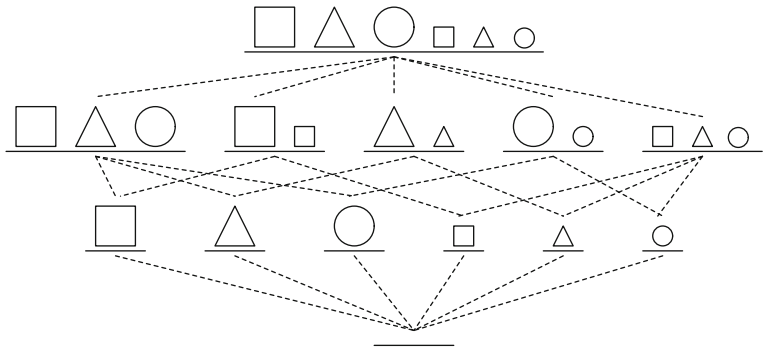


Fig. 2 The partial order of concepts with respect to their coverage of objects

In Fig. 2, the concept space C is shown and the partial order between the concepts is indicated by the dashed lines. This partial order is the key to Mitchell's approach. The idea is to always maintain a set of *most general concepts* and a set of *most specific concepts* that are consistent and complete with respect to the presented training examples.

If a most *specific* concept c_s does contain some object x which is given as a positive example, then all concepts which are supersets of s contain the positive example, i.e. are consistent with the positive example as well as c_s itself. Similarly, if a most *general* concept c_g does not contain some object x which is given as a negative example, then all concepts which are subsets of s_g do not contain the negative example, i.e. are consistent with the negative example as well as c_g itself.

In other words, the set of consistent and complete concepts which exclude all presented negative examples and include all presented positive examples is defined by the sets of concepts S and G being the most specific and most general concepts consistent and complete with respect to the data. That is, all concepts of C which lie between S and G are complete and consistent as well. A concept c lies between S and G, if and only if there are two concepts $c_g \in G$ and $c_s \in S$ such that $c_s \subseteq c \subseteq c_g$. An algorithm that maintains the set of consistent and complete concepts is sketched in Fig. 3. Consider the following example to illustrate the use of the algorithm in Fig. 3:

Example. Let us denote the various sets S and G by S_n and G_n, respectively after the nth example has been processed. Before the first example is presented, we have $G_0 = \{X\}$ and $S_0 = \{\{\}\}$.

Suppose a big triangle is presented as positive example. Then, G remains the same, but the concept in S has to be generalised. That is, we obtain $G_1 = G_0\{X\}$ and $S_1 = \{\{b.t\}\}$.

Suppose the second example being a small circle as negative example: Then S remains the same, but the concept in G has to be specialised. That is, we obtain

Given: A concept space C from which the algorithm has to choose one concept as the target concept c_t. A stream of examples of the concept to learn. (The examples are either positive or negative examples of the target concept c_t.)

begin
Let S be the set of most specific concepts in C; usually the empty concept.
Let G be the set of most general concepts in C; usually the single set X.
while there is a new example e **do**
if e is a positive example
then Remove in G all concepts that do not contain e.
Replace every concept $c_o \in S$ by the set of
most specific generalisations with respect to e and S.
endif
if e is a negative example
then Remove in S all concepts that contain e.
Replace every concept $c_o \in G$ by the set of
most general specialisations with respect to e and G.
endif
endwhile
end.

Note: The set of of **most specific generalisations** of a concept c with respect to an example e and a set of concepts G are those concepts $c_g \in C$ where $c \cup \{e\} \subseteq c_g$ and there is a concept $c_G \in G$ such that $c_g \subseteq c_G$ and there is no concept $c_{g'} \in C$ such that $c \cup \{e\} \subseteq c_{g'} \subset c_g$. The set of of **most general specialisations** of a concept c with respect to an example e and a set of concepts S are those concepts $c_s \in C$ where $c_s \subseteq c \setminus \{e\}$ and there is a concept $c_S \in S$ such that $c_S \subseteq c_s$ and there is no concept $c_{s'} \in C$ such that $c_s \subset c_{s'} \subseteq c \setminus \{e\}$.

Fig. 3 An algorithm for maintaining the version space

$G_2 = \{\{b.s, b.t, b.c\}, \{b.t, s.t\}\}$ and $S_2 = S_1 = \{\{b.t\}\}$. Note that G_2 contains two different concepts which neither contain the negative example but which are both supersets of the concept in S_2.

Let the third example be a big square as a positive example. Then, in G we remove the second concept since it does not contain the new positive example and the concept in S has to be generalised. That is, we obtain $G_3 = \{\{b.s, b.t, b.c\}\}$ and $S_3 = \{\{b.s, b.t, b.c\}\}$.

That is, $S_3 = G_3$ which means, that there is only a single concept left which is consistent and complete with respect to all presented examples. That is, the only possible result of any learning algorithm that learns only concepts in C that are consistent and complete is given by $\{b.s, b.t, b.c\}$.

In general, the learning process can be stopped if S equals G meaning that S contains the concept to be learned. However, it may happen that $S \neq G$ and an example is presented which forces either S being generalised or G being specialised, but there is no generalisation (specialisation) possible according to the definition in Fig. 3.

This fact would indicate, that there is no concept in C which is consistent with the presented learning examples. Reason for that is either that the concept space did

not contain the target concept, i.e. C was inappropriately chosen for the application domain, Or that the examples contained noise, i.e. that some of the presented data was incorrect. This may either be a positive example presented as a negative or vice versa, or an example inaccurately described due to measurement errors or other causes. For example, the positive example *big triangle* may be misrepresented as the positive example *big square*.

If the concept space C does not contain all possible concepts on the set X of chosen representations, the choice of the concept space presumes that the concept to be learned is in fact in C, although this is not necessarily the case. Utgoff and Mitchell [5] introduced in this context the term *inductive bias*. They distinguished *language bias* and *search bias*. The language bias determines the concept space which is searched for the concept to be learned (the target concept). The search bias determines the *order* of search within a given concept space. The proper specification of inductive bias is crucial for the success of a learning system in a given application domain.

In the following sections, the basic ideas of the most influential approaches in (symbolic) machine learning are presented.

4 Learning of Classification Rules

There are different ways of learning classification rules. Probably the best known ones are the successive generation of disjunctive normal forms, which is done by the AQ family of learning algorithms, which belongs to one of the very early approaches in machine learning. Another well-known alternative is to simply transform decision trees into rules. The C4.5 [6] program package, for example, contains also a transformation program, which converts learned decision trees into rules.

4.1 Model-Based Learning Approaches: The AQ Family

The AQ algorithm was originally developed by Michalski [7], and has been subsequently re-implemented and refined by several authors (e.g. [8]). Opposed to ID3[4] the AQ algorithm outputs a set of *'if...then...'* classification rules rather than a decision tree. This is useful for expert system applications based on the production rule paradigm. Often it is a more comprehensible representation than a decision tree. A sketch of the algorithm is shown in Table 1. The basic AQ algorithm assumes no noise in the domain. It searches for a concept description that classifies the training examples perfectly.

[4] C4.5, the successor of ID3 actually contains facilities to convert decision trees into *if ... then ...* rules.

Table 1 The AQ algorithm: Generating a cover for class C

Procedure AQ(POS, NEG) **returning** COVER**:**
Input: A set of positive examples POS
and a set of negative examples NEG.
Output: A set of rules (stored in `cover`) which recognises all positive examples and none of the negative examples.

let COVER be the empty cover;
while COVER does not cover all positive examples in POS
  select a SEED, i.e. a positive example not covered by COVER;
  call procedure STAR(SEED, NEG) to generate the STAR, i.e. a set of
    complexes that cover SEED but no examples in NEG;
  select the best complex BEST from the star by user-defined criteria;
  add BEST as an extra disjunct to COVER;
return COVER.

Procedure STAR(SEED, NEG) **returning** STAR**:**

let STAR be the set containing the empty complex;
while there is a complex in STAR that covers some
  negative example $E_{neg} \in$NEG,
  Specialise complexes in STAR to exclude $\mathsf{E}_{\mathsf{neg}}$ by:
    let EXTENSION be all `selectors` that cover SEED but not $\mathsf{E}_{\mathsf{neg}}$;
% `selectors` are attribute-value specifications
% which apply to `seed` but not to $\mathsf{E}_{\mathsf{neg}}$.
    let STAR be the set $\{x \wedge y | x \in \mathsf{STAR}, y \in \mathsf{EXTENSION}\}$;
    remove all complexes in STAR that are subsumed by other
      complexes in STAR;
  Remove the worst complexes from STAR
    until size of STAR $\leq$ user-defined maximum (*maxstar*).
return STAR.

The AQ algorithm

The operation of the AQ algorithm is sketched in Table 1. Basically, the algorithm generates a so-called *complex* (i.e. a conjunction of attribute-value specifications). A *complex* covers a subset of the positive training examples of a class. The complex forms the **condition** part of a production rule of the following form:

'if **condition** then predict **class**'.

The search proceeds by repeatedly specialising candidate complexes until a complex is found which covers a large number of examples of a single class and none of other classes. As indicated, AQ learns one class at a time. In the following, the process for learning a single concept is outlined.

Learning a Single Class

To learn a single class c, AQ generates a set of rules. Each rule recognises a subset of the positive examples of c. A single rule is generated as follows: First a "seed" example E from the set of positive examples for c is selected. Then, it is tried to generalise the description of that example as much as possible. Generalisation means here to abstract as many attributes as possible from the description of E.

AQ begins with the extreme case that all attributes are abstracted. That is, AQs first rule has the form '`if` true `then` predict class `c`.' Usually, this rule is too general. However, beginning with this rule, stepwise specialisations are made to exclude more and more negative examples. For a given negative example, *neg* covered by the current rule AQ searches for a specialisation which will exclude *neg*. A specialisation is obtained by adding another condition to the condition part of the rule. The condition to be added is a so-called *selector* for the seed example. A selector is an attribute value combination which applies to the seed example but not to the negative example *neg* currently being considered.

This process of searching for a suitable rule is continued until the generated rule covers only examples of class `c` and no negative examples, i.e. no examples of other classes.

Since there is generally more than one choice of including an attribute-value specification, a set of "best specialisations-so-far" are retained and explored in parallel. In that sense, AQ conducts a kind of beam search on the hypothesis space. This set of solutions which is steadily improved is called a *star*. After all negative examples are excluded by the rules in the star, the best rule is chosen according to a user-defined evaluation criterion. By that process, AQ guarantees to produce rules which are *complete* and *consistent* with respect to the training data, if such rules exist. AQ's only hard constraint for the generalisation process is not to cover any negative example by a generated rule. Soft constraints determine the order of adding conditions (i.e. attribute value specifications).

Example. Consider the training examples given in Fig. 2. Learning rules for the class of pleasant weather would work as follows:

A positive example E is selected as a seed, say Example 4 having the description $E = [(a = \mathit{true}) \wedge (b = \mathit{false}) \wedge (c = \mathit{true}) \wedge (d = \mathit{false})]$.

From this seed, initially *all* attributes are abstracted, i.e. the first rule is if true then pleasant.

Since this rule clearly covers also weather situations which are known as unpleasant, the rule has to be specialised. This is done, by re-introducing attribute-value specifications which are given in the seed example. Thus, each of the four attributes is considered. For every attribute it is figured out whether its re-introduction excludes any of the negative examples.

Considering attribute a:

The condition ($a = \mathit{false}$) is inconsistent with Examples 1 and 2, which are both negative examples. Condition ($b = \mathit{false}$) excludes the Examples 1 and 2, which are negative and it excludes the positive Example 3 as well. Condition ($c = \mathit{true}$)

excludes the positive Example 5 and the negative Example 6. Finally, condition ($d = false$) excludes three negative Examples 2, 6, and 7, while it does not exclude any positive example.

Intuitively, specialising the rule by adding condition ($d = false$) appears to be the best.

However, the rule

if ($d = false$) then pleasant

still covers the negative Example 1. Therefore, a further condition has to be added. Examining the three possible options leads to the following:

The condition ($a = false$) is inconsistent with Examples 1 and 2, which are both negative examples, i.e. adding this condition would result in a consistent and complete classification rule.

Condition ($b = false$) excludes the Examples 1 and 2, which are negative and it excludes the positive Example 3 as well. After adding this condition the resulting rule would no longer cover the positive Example 3, while all negative examples are excluded as well.

Condition ($c = true$) excludes the positive Example 5 and the negative Example 6 and is thus of no use.

Again, it appears natural to add the condition ($a = false$) to obtain a satisfying classification rule for pleasant weather:

if ($a = false$) $\wedge$ ($d = false$) then pleasant

4.2 Non-Boolean Attributes

In many practical applications of machine learning, objects are not represented by a set of boolean attributes.

The example given earlier considered the simplest case where the objects were described by boolean attributes only. Considering further types of attributes, as mentioned in Sect. 5.3, some extensions to the demonstrated approach are necessary. Basically, more attribute-value specifiers as in the boolean case have to be considered. In the boolean case the possible specifications were ($a = false$) or ($a = true$).

- For discrete attributes without any particular relation among its different values, the attribute specifications can easily be extended from only boolean values to the full range of attribute values. That is, the possible specifications are ($A = v_1$), ($A = v_2$), ..., ($A = v_n$). Also, subsets of values can be used for constructing selectors, i.e. for including the seed example and excluding the negative examples. These are called *internal disjunctions*.
- Internal disjunctions: Disjunctions which allow more than one value or interval for a single attribute. Since the disjuncts concern the same attribute, the disjunction is called *internal*. Examples are (colour = red *or* green *or* blue).
- For linear attributes, see Fig. 4 for a linear attribute. A linear attribute is an attribute, where an example has a particular value within a range of linearly ordered

values. Concepts are defined by defining an admissible interval within the linearly ordered attribute values, as e.g. $(A < v_1)$, $(A \geq v_1)$, ..., $(A < v_n)$, $(A \geq v_n)$. Also 'two-sided' intervals of attribute values like $(v_1 < A \leq v_2)$ can be handled by AQ[3].

- For continuous attributes, the specifications are similar to the case of linear attributes, except that instead of considering the value range of the attribute, the values that actually occur in the given positive and negative examples are considered and ordered to be $v_1, v_2, ..., v_k$. Subsequently as thresholds, the values of $\frac{v_i + v_{i+1}}{2}$ are calculated and used as in the case of linear attributes.
- Tree-structured attributes: See Fig. 5. Tree-structured attributes replace the linear ordering of the attribute value range by a tree-structure. The value of a node n in the tree structure is considered to cover all values which are either assigned directly to one of n's successor nodes or are *covered* by one of n's successor nodes.

  The defined partial ordering is used to specify attribute values: Every possible attribute value is considered. Some attribute values do not subsume other values; these are treated as in the case of the discrete attributes. Those values which subsume other values are used to group meaningful attribute values together. For example, $(a = polygon)$ would subsume all values down the tree, i.e. triangle, square, etc.

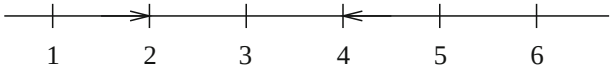


Fig. 4 Linear Attributes

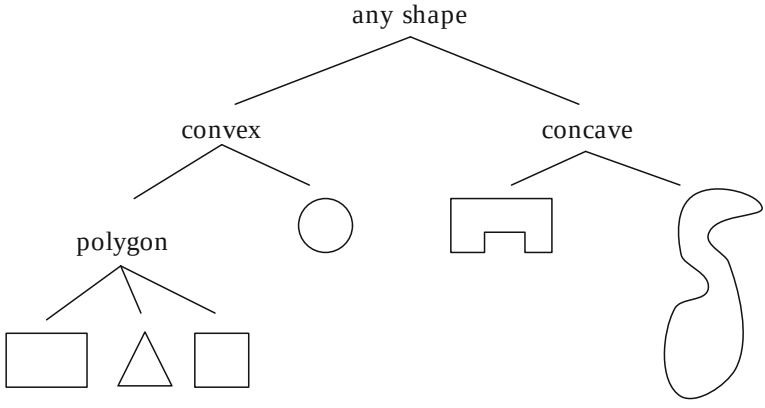


Fig. 5 An example of a tree-structured attribute "shape"

The advanced versions of the AQ family (see, e.g. [3]) of learning algorithms deal with all these different attribute types by determining selectors as *minimal dominating atoms*. A minimal dominating atom is a single attribute with a specified admissible value range. This is that value range, which excludes the given negative example and covers as many positive examples as possible. That is, in the case of value ranges for linear or continuous attributes, an interval is determined, by excluding the values of as many negative examples as possible and by including the values of the positive examples.

4.3 Problems and Further Possibilities of the AQ *Framework*

Searching for Extensions of a Rule

The search for specialising a too general classification rule is heuristic in AQ due to its computational complexity.[5] A kind of greedy approach is conducted by adding one constraint at a time to a rule. Since there is usually more than one choice to add a further constraint to a rule, all such ways of adding a constraint are tried, by adding all new rules to the so-called *star*. The star contains only a pre-specified maximum number of rule candidates.

If after new rules are added to the star, the number of rules in the star exceeds the specified maximum number, rules are removed according to a user-specified preference or quality criterion. As quality function, typically heuristics are used by the AQ system like

> 'Number of correctly classified examples divided by total number of examples covered.'

Learning Multiple Classes

In the case of learning multiple classes, AQ generates decision rules for each class in turn. Learning a class c is done by considering all examples with classification c as positive examples and considering all others as negative examples of the concept to learn. Learning a single class occurs in stages. Each stage generates a single production rule, which recognises a part of the positive examples of c. After creating a new rule, the examples that are recognised by a rule are removed from the training set. This step is repeated until all examples of the chosen class are covered. Learning the classification of a single class as above is then repeated for all classes.

[5] Note that the set cover problem is known to be **NP**-complete [10], which is very related to various quality criteria one may have in mind for a rule discriminating between negative and positive examples. That is, for many quality measures, the task to find the *best* rule will be **NP**-hard.

Learning Relational Concepts Using the AQ Approach

The presented approach has also been extended to learn relational concepts, containing predicates and quantifiers instead of just fixed attributes. For more details, see, e.g. [3].

Extending AQ

Various extensions to the basic AQ algorithm presented earlier have been developed. One important class of extensions addresses the problem of noisy data, e.g. the CN2 algorithm [9]. For the application of systems based on the AQ algorithm to real-world domains, methods for handling noisy data are required. In particular, mechanisms for avoiding the *over-fitting* of the learned concept description to the data are needed. Thus, the constraint that the induced description must classify the training data perfectly has to be relaxed.

AQ has problems to deal with noisy data because it tries to fit the data completely. For dealing with noisy data, only the major fraction of the training examples should be covered by the learning rules. Simultaneously, a relative simplicity of the learned classification rule should be maintained as a heuristic for obtaining plausible generalizations.

More recent developments include AQ21 [11] which, among other features such as better handling of noisy situations, is also capable of generating rules with exceptions.

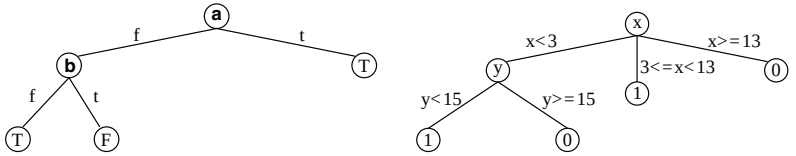
5 Learning Decision Trees

Decision trees represent one of the most important class of learning algorithms today. Recent years have seen a large number of papers devoted to the theoretical as well as empirical studies of constructing decision trees from data. This section presents the basic ideas and research issues in this field of study.

5.1 Representing Functions in Decision Trees

There are many ways for representing functions, i.e. mappings from a set of input variables to a set of possible output values. One such way is to use decision trees. Decision trees gained significant importance in machine learning. One of the major reasons is that there exist simple yet efficient techniques to generate decision trees from training data.

Abstractly speaking, a decision tree is a representation of a function from a possibly *infinite* domain into a *finite* domain of values. That is,



(a) A simple decision tree for a function $f : \{f, t\} \times \{f, t\} \rightarrow \{F, T\}$.

(b) A function $f(x, y)$ of the form $f : \mathbf{N} \times \mathbf{N} \rightarrow \{0, 1\}$.

Fig. 6 Two decision trees on different domains

$$D : X \rightarrow C,$$

where X is the possibly infinite set of objects and C the set of classes assigned to the objects by the function D realized by a decision tree.

The representation of such a function by a decision tree is at the same time also a guide for how to efficiently compute a value of the represented function. Fig. 6(a) shows a decision tree of a simple boolean function. The decision tree is a tree in which all leaf nodes represent a certain function value. To use a decision tree for determining the function value for a given argument, one starts in the root node and chooses a path down to one leaf node. Each non-leaf node in the tree represents a decision on how to proceed the path, i.e. which successor node is to be chosen next. The decision criterion is represented by associating conditions[6] with each of the edges leading to the successor nodes. Usually, for any non-terminal node n a single attribute is used to decide on a successor node. Consequently, that successor node is chosen for which the corresponding condition is satisfied. In Fig. 6(a), the decision in the root node depends solely on the value of the variable a. In the case of $a = F$, the evaluation of the tree proceeds at the left successor node, while being $a = t$ would result in considering the right successor node. In the latter case the evaluation had already reached a leaf node which indicates that $f(t, t) = f(t, f) = T$. In the case of $a = f$, the value of b determines whether the left or the right successor node of node 2 has to be chosen, etc.

5.2 The Learning Process

The learning of decision trees is one of the early approaches to machine learning. In fact, Hunt [12] developed his *Concept Learning System* CLS in the 1960s, which was already a decision tree learner. A decision tree representation of a classification function is generated from a set of classified examples.

Consider the examples in Table 2: Assume, we want to generate a decision tree for the function f which determines the value P only for the examples $3 - 5$ such as the tree in Fig. 7.

[6] normally mutually exclusive conditions...

Table 2 A set of examples for the concept of pleasant weather. "P" indicates pleasant weather, while "U" indicates unpleasant weather

Number	a = sunny	b = hot	c = humid	d = windy	class = $f(a, b, c, d)$
1	true	true	true	false	U
2	true	true	true	true	U
3	false	true	true	false	P
4	false	false	true	false	P
5	false	false	false	false	P
6	false	false	false	true	U
7	false	false	true	true	U

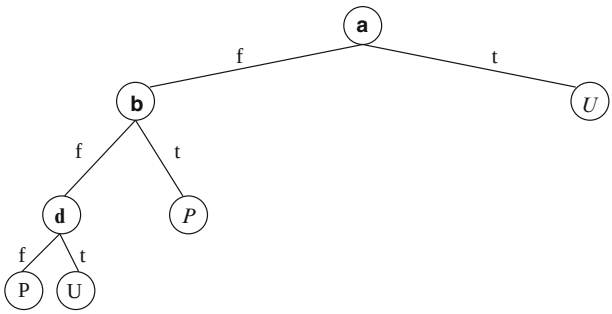


Fig. 7 A decision tree representing the boolean function partially defined in the table above (The italic P (and U) represents the inclusion of actually undefined function values which are set to P (or to U respectively) by default)

The learning algorithm can be described at an abstract level as a function from sets of feature vectors to decision trees. Generalisation occurs indirectly: The input example set does not specify a function value for the entire domain. Opposed to that a decision tree determines a function value for the entire domain, i.e. for all possible feature vectors.

The basic idea of Quinlan's ID3 algorithm [13], which evolved later to program package C4.5 [6], is sketched in Fig. 8. The general idea is to split the given set of training examples into subsets such that the subsets eventually obtained contain only examples of a single class. Splitting a set of examples S into subsets is done by choosing an attribute A and generating the subsets of S such that all examples in one subset have the same value in the attribute A. In principle, if an attribute has more than two values, two or more groups of values may be chosen such that all examples which have a value in the attribute A that belongs to the same group are gathered in the same subset. In order to cope with noise, it is necessary, to stop splitting sets of examples into smaller and smaller subsets before all examples in one subset belong to the same class.

Therefore, a decision tree learning algorithm has the two following functions that determine its performance:

Input: A set of examples E, each consisting of a set of m attribute values corresponding to the attributes $A_1, ..., A_m$ and class label c. Further, a termination condition $T(S)$ is given, where S is a set of examples and an evaluation function $ev(A, S)$ where A is an attribute and S a set of examples. The termination condition is usually that all the examples in S have the same class value.
Output: A decision tree.

1. Let $S := E$.
2. If $T(S)$ is true, then **stop**.
3. For each attribute A_i determine the value of the function $ev(A_i, S)$. Let $A_j = \max_{i \in \{1,...,m\}} ev(A_i, S)$. Divide the set S into subsets by the attribute values of A_j. For each such subset of examples E_k call the decision-tree learner recursively at step (1) with E set to E_k. Choose A_j as the tested attribute for the node n and create for each subset E_k a corresponding successor node n_k.

Fig. 8 A sketch of the ID3 algorithm

- A *termination condition* which determines when to refrain from further splitting of a set of examples.
- An *evaluation function* which chooses the "best" attribute on which the current set of examples should be split.

The Termination Condition: $T(S)$

As indicated in Fig. 8, the termination condition $T(S)$ plays an important role in inducing decision trees. The simplest form of a termination condition says 'stop' when all examples in S have the same class.

More sophisticated versions of the termination condition stop even when not all examples in S have the same class. This is motivated by the assumption that either the examples in the sample contain noise and/or that only a statistical classification function can be expected to be learned.

The Evaluation Function: $ev(a, S)$

The ID3 algorithm as shown in Fig. 8 performs only a one-level look ahead to select the attribute for the next decision node. In that sense, it is a greedy algorithm. Quinlan introduced in his original paper [13], an information theoretic measure, which performs fairly well. Other heuristic proposals for a selection criterion include pure frequency measures as in CLS [12], or the *Gini index* as used in CART[14] or a statistical test as in [15]. See also Bratko [16] for a discussion of these measures and a simple implementation of a decision tree learner in PROLOG. An exhaustive search for finding a minimal size tree is not feasible in general, since the size of the search space is too large (exponential growth).

Quinlan's entropy measure[7] estimates how many further splits will be necessary after the current set of examples is split (by using the splitting criterion being evaluated): Consider a set of examples E, a set of classes $C = \{c_i | 1 \leq i \leq n\}$, and an attribute A with values in the set $\{v_j \mid 1 \leq j \leq m\}$. The information in this distribution needed to determine the class of a randomly chosen object is given by:

$$info(C) = -\sum_{i \in \{1,\ldots,n\}} p_i \log_2 p_i \tag{1}$$

where p_i is the probability of an example falling into class c_i. The probability P_i will be estimated by the relative frequency of an example in E falling into class i. That is, p_i will be estimated as $\frac{|E_i|}{|E|}$. After splitting on an attribute, still some further information may be needed, depending on the subset of examples associated to each of the branches. That is, after a split, the information needed on average can be computed by computing the information needed for each of the subsets according to formula 1 and by weighting the result by the probability, for taking the respective branch of the decision tree. Then, the following gives the information needed on average to determine the class of an object *after* the split on attribute A_j:

$$info(C|A) = -\sum_{j \in \{1,\ldots,m\}} \sum_{i \in \{1,\ldots,n\}} p_{ij} \log_2 p_{ij}, \tag{2}$$

where p_{ij} denotes the probability for an object to have attribute value v_j and falling into class i. This is the measure proposed by Quinlan [13]. Again p_{ij} will be estimated by the relative frequency of an example in E of having attribute value v_j and falling into class i. Intuitively, the amount of information needed on average after a split on a particular attribute should be as small as possible. The choice of a splitting attribute *minimises* on this measure.

Quinlan defines the inverse, the information gain achieved by a split as follows:

$$Gain(E, A) = info(E) - info(E|A) \tag{3}$$

As a consequence, the objective is then to maximise the information gain by choosing a splitting criterion.

Example. Considering the examples given in Table 2 and assuming the relative frequency of examples in the given sample equals their probability, the following values would be computed:

Initially the required information needed to determine the class of an example is given by:

[7] The entropy measure has been first formulated in C. Shannon and Weaver [17] as a measure of information. The intuition behind it is that it gives the average number of bits necessary for transmitting a message using an optimal encoding. In the context of decision trees, the number of bits required for transmitting a message corresponds to the number of splits required for determining the class of an object.

$$info(E) = -\left(\frac{4}{7}\log_2\frac{4}{7} + \frac{3}{7}\log_2\frac{3}{7}\right) \approx 0.98$$

Considering the complete set of seven objects and splitting on a:

$$info(E|a) = -\left(\frac{2}{7}(1\log_2 1) + \frac{5}{7}\left(\frac{3}{5}\log_2\frac{3}{5} + \frac{2}{5}\log_2\frac{2}{5}\right)\right) \approx 0.69$$

and splitting on b:

$$info(E|b) = -\left(\frac{3}{7}\left(\frac{2}{3}\log_2\frac{2}{3}+\frac{1}{3}\log_2\frac{1}{3}\right) + \frac{4}{7}\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right)\right)\approx 0.96$$

and splitting on c:

$$info(E|c) = -\left(\frac{5}{7}\left(\frac{3}{5}\log_2\frac{3}{5}+\frac{2}{5}\log_2\frac{2}{5}\right) + \frac{2}{7}\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right)\right)\approx 0.978$$

and splitting on d:

$$info(E|d) = -\left(\frac{3}{7}(1\log_2 1) + \frac{4}{7}\left(\frac{1}{4}\log_2\frac{1}{4} + \frac{3}{4}\log_2\frac{3}{4}\right)\right) \approx 0.46$$

Hence, splitting on attribute d requires on average the smallest amount of further information for deciding the class of an object. In fact, in three out of seven cases the class is known to be *unpleasant* weather after the split. The remaining four examples are considered for determining the next split in the respective tree branch. That is, for the following step only the subset of examples shown in Table 3 has to be considered.

Then, we get the following values for the required information after splitting on attribute a:

$$info(E|a) = -\left(\frac{1}{4}(1\log_2 1) + \frac{3}{4}(1\log_2 1)\right) = 0$$

and splitting on attribute b:

$$info(E|b) = -\left(\frac{1}{2}(1\log_2 1) + \frac{1}{2}\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right)\right) = 0.5$$

Table 3 The reduced set of examples after splitting on attribute d and considering only those examples with the attribute value *d=false*

Number	A = sunny	B = hot	C = humid	D = windy	Class = $f(a,b,c,d)$
1	true	true	true	false	U
3	false	true	true	false	P
4	false	false	true	false	P
5	false	false	false	false	P

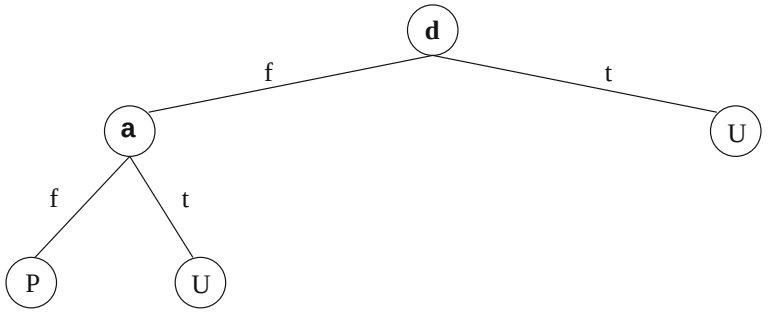


Fig. 9 The decision tree obtained using Quinlan's information-theoretic measure. In fact, the above decision tree is the shortest possible for the given example set

and splitting on attribute c:

$$info(E|c) = -\left(\frac{1}{4}(1\log_2 1) + \frac{3}{4}\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right)\right) \approx 0.688$$

Consequently, the next attribute chosen to split on is attribute a, which results in the decision tree shown in Fig. 9.

Practical experience has shown that this information measure has the drawback of favouring attributes with many values. Motivated by that problem, Quinlan introduced in C4.5 [6] a normalised Entropy measure, the gain ratio, which takes the number of generated branches into account. The gain ratio measure [6], considers the potential information that may be gained by a split of E into $E_1, ..., E_k$, denoted by $Split(E, A)$. The potential information is that each branch has a unique class assigned to it, i.e. it can be defined as follows:

$$Split(E, A) = -\sum_{i=1}^{k} \frac{|E_i|}{|E|}\log_2\frac{|E_i|}{|E|} \tag{4}$$

where A splits the set of examples E into the disjoint subsets $E_1, ..., E_k$.

The gain ratio measure, then, is defined as follows:

$$Gainratio(E, A) = \frac{Gain(E, A)}{Split(E, A)}$$

In the release 8 of C4.5 [18], the gain ratio computed for binary splits on continuous attributes is further modified to improve predictive accuracy.

Good and Card [19] provide a Bayesian analysis of the diagnostic process with reference to errors. They assume a utility measure $u(i, j)$ for accepting class c_j when the correct class is actually c_i. Based on that they developed a selection criterion which takes the optimisation of utility into account.

5.3 Representational Aspects in Decision Tree Learning

Continuous Attributes

Quinlan's original ID3 algorithm handles attributes with discrete values only. Unfortunately, many problem domains contain continuous descriptors, such as height or weight, which are difficult to handle discretely. The ID3 algorithm does not utilise the implicit information contained in the fact that the values of an attribute are (meaningfully) ordered. As a consequence, the original ID3 algorithm has been extended in C4.5 [6] to handle continuous attributes as well. Quinlan's model trees [20] can also generate continuous output functions. The basic idea is to find a binary split on the value range. For that purpose all values $v_1, ..., v_m$ that occur in the actually given examples are considered and ordered according to their values. Subsequently, every possible split by choosing a threshold between v_i and v_{i+1} for all $i \in \{1, ..., m-1\}$ are considered and the best split is chosen.

Unknown Attribute Values

In a number of applications it may happen that an example is not completely described, i.e. that some of its attribute values are missing. This may be due to missing measurements of certain attributes, errors in or incompleteness of reports, etc. For example, when dealing with large historical databases, often some values for attributes are unknown. In medical cases, not for every patient a specific test has been taken – hence it is rather normal that some values are missing. However, one standard approach to cope with the problem of unknown values is to estimate the value using the given examples which have a specified value. This approach is taken in, e.g. ASSISTANT [21] as well as C4.5 [6].

However, one can actually distinguish at least the following reasons for the missing values which suggest different treatments: Missing because not important (don't care), not measured, and not applicable (e.g. a question like "Are you pregnant" is not applicable to male patients). These reasons could be very valuable to exploit in growing a tree or in concept learning in general.

Splitting Strategies

It is interesting to note, that if an attribute has more than two values it may still be useful to partition the value set only into two subsets. This guarantees that the decision tree will contain only binary splits. The problem with a naive implementation of this idea is that it may require 2^{n-1} evaluations, where n is the number of attribute values. It has been proved by Breiman et al. [14] that for the special case of only two class values of the examples, there exists an optimal split with no more than $n-1$ comparisons. In the general case, however, heuristic methods must be used.

One simple idea to deal with that problem is as follows: Examples are ordered by the values of the attribute in question and then the best split which partitions the examples into two sets is chosen. This idea has been implemented, e.g. in CART[14] and in C4.5 [6].

5.4 Over-Fitting and Tree Pruning

From a statistical point of view, the decision tree learning algorithm of Fig. 8 has the drawback that it tends to over-fit the data. That is, the learned tree is too large and – as a consequence – classifies less accurate than a smaller tree would do. This is basically due to noise in the training data. Two main methods have been studied to overcome this problem:

1. Modify the evaluation function to terminate search when no attribute is considered to provide *significant* information about the class. See e.g. [22].
2. Prune the obtained decision tree to reduce its complexity while keeping its classification accuracy at a reasonable level. See. e.g. [6, 14, 76, 77].

It is widely assumed, that pruning after generating the full decision tree is the most effective method [14]. A number of pruning techniques have been proposed. Pruning may occur in different ways:

- The tree may be pruned beginning with its leaf nodes. See e.g. [23].
- Pruning may take place by merging intermediate nodes of the tree together. For example, *weakest link pruning* [14].

Pruning beginning with its leaf nodes can be done as follows, see, e.g. [23] Ascending in the tree beginning in its leaf nodes and testing at every nonleaf node whether replacing the subtree of node n by one of its leaf nodes would improve the classification accuracy, For determining the classification accuracy statistical tests have to be used on a separate set of test examples. These test examples must be statistically independent from the training examples used for learning the decision tree to be pruned. If a sufficient number of test examples are not easily available, this may cause a problem. One approach to deal with that problem is using cross validation [14].

For more recent developments, such as the commercially available variant of C4.5, known as See5 or C5.0, see, e.g. [24].

6 Inductive Logic Programming

Inductive Logic Programming (in short ILP) is a method to construct logic programs describing a target concept based on examples of this target concept. Although, ILP has limited practical applications so far, its potential seems very promising.

The ILP framework allows to incorporate flexibly domain knowledge into the learning process. Furthermore, ILP can learn relational concepts which can, for example, compare different attributes with each other in a concept definition. This is opposed to, e.g. decision tree learners like C4.5, which tests one attribute at a time and compares the attribute rather with a constant value than with another attribute.

The term *Inductive Logic Programming* is motivated by the idea of considering the hypothesis space being the set of logic programs. Then, the result of learning will be a particular logic program. If the learning process is an inductive process, i.e. generating a more general logic program from a given set of examples, then the generation of a *logic program* takes place in an *inductive* way. The field of inductive logic programming comprises a number of techniques and implemented systems which can be subsumed under this idea. One particularly appealing idea in this field is that background knowledge can easily be incorporated into the framework. That is, the hypothesis space can be shaped by describing certain knowledge known to the user a priori, by giving a partial logic program. Then, the input to the learning system is both, training examples as well as a partial logic program. The output is a completed logic program. Since the hypothesis space is the set of logic programs, or at least a subset of these, the descriptional power of what can be learned is substantially larger than that of propositional learners like the decision tree learner discussed earlier. In the following, the central ideas of inductive logic programming will be presented.

One of the very early systems which incorporated relational background knowledge into the generalisation process was Michalski's INDUCE [25]. Inspiring for many of the current approaches was the work on model inference by Shapiro [26], Plotkin's work on least general generalisation [27], the work by Sammut and Banerji [28] on the interactive learning system MARVIN, etc.

Before we formally describe the ILP setting, let us first briefly consider one simple example (from [16] and [29]). Let us assume we already have the following three predicates (relations) defining some family relations:

$$parent(pam, bob), parent(tom, bob), parent(tom, liz)$$
$$parent(bob, ann), parent(bob, pat), parent(pat, jim)$$

$$female(pam), female(liz), female(pat), female(ann)$$

$$male(tom), male(bob), male(jim)$$

We can consider the above predicates as our background knowledge $\mathcal{B}$ about the problem. Now we want to learn a definition of the new predicate $has\_daughter(X)$, given the following set P of positive examples (meaning tom and bob each has a daughter),

$$has\_daughter(tom), \ has\_daughter(bob)$$

and the following set N of negative examples (meaning pam and jim each do not have a daughter),

$$has\_daughter(pam),\ has\_daughter(jim)$$

The task now is to learn a new definition of the predicate $has\_daughter(X)$, using the above background knowledge (three predicates $parent$, $male$ and $female$), such that a new definition of $has\_daughter$ is true for all the positive examples and not true for any negative examples. One possible hypothesis (call it H) an ILP program might output is the following:

$$has\_daughter(X) \leftarrow parent(X, Y), female(Y).$$

The above hypothesis satisfies our constraints by explaining both the positive examples, and not explaining any negative examples. Note that here the new definition of $has\_daughter$ uses two existing relations from the available background knowledge.

Now, let us define one of the most popular ILP settings [29]. Let $\mathcal{B}$ represents background knowledge (normally a set of Horn-clauses), P represents positive examples (normally ground literals) and N represents negative examples (normally ground literals). The task is to find a hypothesis H such that

- $\forall p \in P : H \cup \mathcal{B} \models p$ (completeness of the hypothesis H)
- $\forall n \in N : H \cup \mathcal{B} \not\models n$ (consistency of the hypothesis H)

In practice, the above definition involves the problem that it may be difficult or even algorithmically impossible[8] to determine whether a particular example is entailed by the given background knowledge and a chosen hypothesis.

Most often, the SLD-resolution proof procedure[9] with bounded or unbounded depth is used for testing whether an example is entailed by a hypothesis h and the given background knowledge B.

Usually there are two types of background knowledge distinguished: *Extensional* background knowledge and *intensional* background knowledge. The *extensional background knowledge* is restricted to be a set of ground facts; i.e. background knowledge is extensional if it contains single literals with constants as arguments only. This is obviously a fairly strong constraint on the expressive power of background knowledge.

Opposed to that is *intensional background knowledge* allowed to contain non-ground clauses as well. That is, it may contain Horn clauses of more than a single literal and a clause may contain variables as well. Most of the empirical ILP systems are using the extensional notion of coverage. Interactive ILP systems, on the other hand, are mostly adopting the idea of intensional coverage.

[8] It may be algorithmically impossible, if the used language is not decidable.

[9] See e.g. [30] for details.

In the following, ways of generalisation in the ILP framework will be discussed. For that purpose, a few more technical terms are defined.

Definition 1. Given a language L, a **generalisation operator** ρ maps a clause c to a set of (more general) clauses $\rho(c)$ which are generalizations of c. That is,

$$\rho(c) = \{c' | c' \in L, c' \succ c\},$$

where $\succ$ is the 'more general than' relation.

There are basically two possible ways to generalise a given clause:

- Substitute the involved terms by more general terms. For example, replace a constant by a variable.
- Remove a literal from the body of a clause.

Relative Least General Generalisation

Plotkin [27] developed his notion of *least general generalisation*. The least general generalisation considers two clauses c_1, c_2 and produces a third clause c_3 which as little more general as possible than both of the clauses c_1 and c_2 together.

For defining exactly what this means, we define a least general generalisation *(lgg)* for the parts of a clause first. That is, we define *(lgg)* for terms, atoms, and literals.

Definition 2. The **least general generalisation** $lgg(e_1, e_2)$ of two syntactical expressions e_1, e_2 is defined as follows. For e_1 and e_2 being

Terms: $lgg(t_1, t_2)$ is given by

1. $lgg(t, t) = t$,
2. $lgg(f(s_1, ..., s_n), f(t_1, .., t_n)) = f(lgg(s_1, t_1), ..., lgg(s_n, t_n))$,
3. $lgg(f(s_1, ..., s_m), g(t_1, ..., t_n)) = V$, where $f \neq g$, and $V_{f(s_1,...,s_m),g(t_1,...,t_n))}$ is a variable,
4. $lgg(s, t) = V_{s,t}$, where $s \neq t$ and at least one of s and t is a variable.

Atoms:

1. $lgg(p(s_1, ..., s_n), p(t_1, ..., t_n)) = p(lgg(s_1, t_1), ..., lgg(s_n, t_n))$, if atoms have the same predicate symbol p,
2. $lgg(p(s_1, ..., s_m), q(t_1, ..., t_n))$ is undefined if $p \neq q$.

Literals:

1. if L_1 and L_2 are atoms, then $lgg(L_1, L_2)$ is computed as defined above,
2. if both L_1 and L_2 are negative literals, $L_1 = \overline{A_1}$ and $L_2 = \overline{A_2}$, then $lgg(L_1, L_2) = lgg(\overline{A_1}, \overline{A_2}) = \overline{lgg(A_1, A_2)}$,
3. if L_1 is a positive and L_2 a negative literal, or vice versa, $lgg(L_1, L_2)$ is undefined.

Clauses: Let $c_1 = \{L_1, ..., L_m\}$ and $c_2 = \{K_1, ..., K_n\}$. Then $lgg(c_1, c_2) = \{L_{ij} = lgg(L_i, K_j) | L_i \in c_1, K_j \in c_2$ and $lgg(L_i, K_j)$ is defined$\}$.

Example. Examples for each of the given types of syntactical expressions are given below:

terms: $lgg(a, a) = a$. $lgg(a, b) = V_{a,b}$. It is important to note that if a variable is introduced for a pair of terms, then for every occurrence of this pair, the *same* variable is introduced. E.g. $lgg(f(a, a), f(b, b)) = f(lgg(a, b), lgg(a, b)) = f(V_{a,b}, V_{a,b})$.

atoms: $lgg(p(a), p(b)) = p(lgg(a, b)) = p(V_{a,b})$. $lgg(q(f(a, b), c), q(a, c)) = q(lgg(f(a, b), a), lgg(c, c)) = q(V_{f(a,b),a}, c)$. $lgg(p(a), q(a, a))$ is undefined.

literals: $lgg(q(a, b), \overline{q(a, a)})$ is undefined.

clauses: $lgg((p(a, b) \leftarrow q(a, c), r(c, b)), (p(d, e) \leftarrow q(d, f), r(f, e))) = p(V_{a,d}, V_{b,e}) \leftarrow q(V_{a,d}, V_{c,f}), r(V_{c,f}, V_{b,e})$.

Now, we are also ready for the definition of *relative least general generalisation* (rlgg) as follows:

Definition 3. The **relative least general generalisation** of two clauses c_1 and c_2 is their least general generalisation $lgg(c_1, c_2)$ *relative* to background knowledge $\mathcal{B}$.

That means for $\mathcal{B}$ being a collection K of ground literals and c_1, c_2 being two atoms A_1, A_2 respectively:

$$rlgg(A_1, A_2) = lgg((A_1 \leftarrow K), (A_2 \leftarrow K)).$$

Inverse Resolution

Muggleton and Buntine introduced *inverse resolution* as a general technique to ILP in [31]. The basic idea is to invert the normal resolution step of the resolution proof procedure.[10]

Resolution of predicate logical clauses involves the substitution of general terms by less general terms. Usually, variables are turned into constants of functions of variables. For inverse resolution, this process needs to be inverted. For example, constants are replaced by variables. The inverse process, however, involves a slight complication compared to the usual substitution process. It requires that the substitutions of constants by variables keeps a record of which occurrence of a constant is replaced by which variable. See the following example:

Example. Let $A_1 = p(f(V_1), V_2)$ be an atom being substituted by $\theta\{V_1/a, V_2/a\}$. That is, $A_1\theta = p(f(a), a)$. Clearly, there should be an inverse substitution θ^{-1} which applied to $A_1\theta$ results in A_1.

[10] See e.g. [30] or [32] for details on the resolution proof procedure.

Therefore, $\theta^{-1} = \{(a, \langle 1 \rangle)/V_1, (a, \langle 2, 1 \rangle/V_2)$ where the index after the term to be replaced indicates which occurrence of the term is to be replaced by the following term.

In the following, inverse resolution is less formally treated and rather examples are given than the lengthy formal definitions.

Example. Assume the following background knowledge:
$\mathcal{B} = \{b_1 = father(paul, peter), b_2 = youngster(peter)\}$.

Assume the current hypothesis $h = \emptyset$ and the following positive example: $e_1 = childof(peter, paul)$ is presented.

Inverse resolution attempts to generalise from the given knowledge by assuming the given example had been derived from the background knowledge. That is, by assuming the example e_1 is the result of a resolution step. If e_1 is the result of a resolution step, there must be a clause c as follows: $c = (e_1 \leftarrow b_1)$ or $c = (e_1 \leftarrow b_2)$. This, however, would not yet constitute a generalisation. The mentioned clauses can rather be deductively derived from $\mathcal{B}$ and e_1. In order to introduce a generalisation process, inverse substitution is used.

That is, as shown in Fig. 10, in the positive example e_1 a constant is turned into a variable by choosing an inverse substitution like $\theta^{-1} = \{paul/x\}$. This results in the clause $c_1 = (e1\theta^{-1} \leftarrow b_1\theta^{-1}) = (childof(peter, x) \leftarrow father(x, peter)$.

This first generalisation step resulting in clause c_1 may be followed by a second one – again taking the available background knowledge $\mathcal{B}$ into account: To derive c_1 by a resolution step involving the second part of the background knowledge, a clause like $c = \{e_1\theta^{-1} \leftarrow (b_1\theta^{-1}, b_2)$ or a more general one is needed. A more general clause than c_2 can be obtained by applying another inverse substitution θ_1^{-1} to c. Thus, for $\theta_1^{-1} = \{peter/y\}$ we obtain: $c_2 = c\theta_1^{-1} = childof(y, x) \leftarrow (father(x, y), youngster(y))$. c_2 is a fairly strong generalisation of the initially given facts in $\mathcal{B}$ and $e1$. This demonstrates the generalisation power of inverse resolution. However, is should also be noted, that the same generalisation power poses the

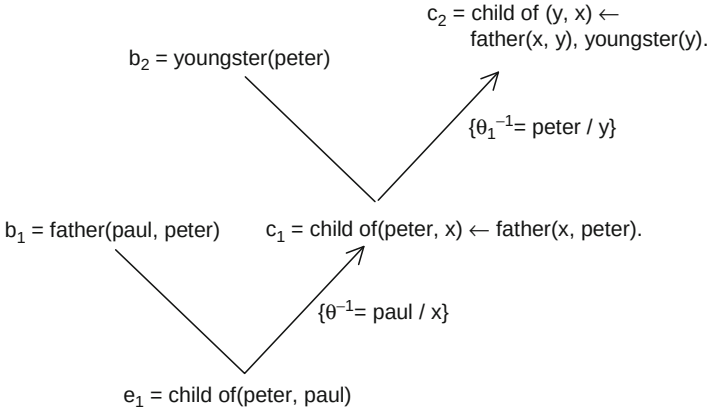


Fig. 10 V operators

practical problem to contain the actually taken generalisation steps to those which are desired. That is, by using these syntactical operations, many clauses can be generated which do not make any sense.

From the given example it can be seen, that in the process of inverse resolution, there were two operations involved which are inherently *non-deterministic*.

- The choice of the emerging clause, i.e. which part of the background knowledge is used to derive the example or an intermediately generated clause, such as c_1 in the example above. In general, there are always many options to choose from.
- The other choice is which particular inverse substitution θ^{-1} is used. The most conservative choice for θ_{con}^{-1} in the earlier example had been the empty substitution. This, however, had led to no generalisation but rather to a mere deductive derivation of the clauses
  $childof(peter, paul) \leftarrow father(paul, peter)$ and
  $childof(peter, paul) \leftarrow (father(paul, peter), youngster(peter))$.
  On the other hand, the most general inverse substitution for obtaining the first clause had been $\theta_g^{-1} = \{peter/x, paul/y\}$, which replaces both involved constants by variables.

The above used generalisation operator for generating a new clause is called the *absorption operator* (the **V** operator). It was already used in the early ILP systems MARVIN [28] and CIGOL [31]. CIGOL uses also several other operators. One important operator among them is *intra-construction*, which is an instance of the **W** operator. A **W** operator combines two **V** operators and introduces thereby a new predicate. The process of introducing new predicates which are not available in the background knowledge or in the examples is called *predicate invention*.

A unified framework, introduced in Muggleton [33], for relative least general generalisation and inverse resolution proposes to use the *most specific inverse resolution* for each step in the generalisation process.

Example. Considering the example for the inverse resolution the most specific inverse resolution would choose the empty set for every inverse substitution step. That is, $\theta_1^{-1} = \emptyset$ and $\theta_2^{-1} = \emptyset$ which results in the clauses
$(childof(peter, paul) \leftarrow father(paul, peter))$ and
$(childof(peter, paul) \leftarrow father(paul, peter), youngster(peter))$
respectively.

The relative least general generalisation can be considered as bottom-up generalisation, i.e. starting from the ground facts and becoming more general step by step. Opposed to that also a top-down approach can be taken. That is, starting with general hypotheses and becoming more specific step by step [34]. For that purpose, we have to consider ways of systematically specialising hypotheses in the ILP framework.

Specialisation Techniques

For specialisation, we are interested in operators which can be applied step by step to a given hypothesis. Usually, the operators for specialisation are called *refinement operators*.

Definition 4. Given a language bias L, a **refinement operator** ρ maps a clause c to a set of clauses $\rho(c)$ which are specialisations (or refinements) of c:

$$\rho(c) = \{c' | c' \in L, c \succ c'\}.$$

where $\succ$ is the 'more general than' relation.

A refinement operator determines normally only the set of minimal specialisations (the most general specialisations) of a clause. For doing that, a specialisation operator can make use of the following two types of syntactical operations:

- Apply a substitution to a clause. This makes the terms occurring in the clause less general, hence the corresponding predicates apply to less objects or tuples of objects respectively.
- Add a literal to the body of a clause. This introduces an additional condition to the body of the clause and hence makes it less general.

After considering the processes of both, the generalisation as well as the specialisation of a given clause, the general structure of most of the ILP systems can be sketched by giving a simplified view of the MIS (Model inference system) [26]; see Fig. 11.

In principle, in the *specialisation loop* as well as in the *generalisation loop*, different stopping criteria are possible. This may be of interest for the following two reasons:

```
Initialize hypothesis h to a (possibly empty) set of clauses in ℒ.
repeat
   Read the next example e
      /* specialisation loop */
      if e is a negative example and e is covered by some clause in h
      then delete all incorrect clauses in h
      endif
      /* generalisation loop */
      if e is a positive example and e is not covered by h
      then generate a clause c which covers e,
         but none of the negative examples presented so far
      endif
   until h is complete and consistent.
   output hypothesis h as result.
forever
```

Fig. 11 A simplified skeleton of the model inference system (MIS)
.

Initialize hypothesis $h := \emptyset$ *to the empty set.*
Let E *be the set of all (positive and negative) examples.*
Initialise $c := T(x_1, ..., x_n) \leftarrow$.
while *there is a positive example* $e_{pos} \in E$ *which is not covered by* h **do**
  while *there is a negative example* $e_{neg} \in E$
      which is covered by a clause $c \in h$ **do**
    remove c *from* h: $h := h \setminus \{c\}$.
    repeat
      determine (heuristically) the best refinement $c_{best} \in \rho(c)$.
      $c := c_{best}$.
    until c *does not cover* e_{neg}.
  Add c *to* h: $h := h \cup \{c\}$.
  endwhile
  Remove all positive examples from E *which are covered by* h.
endwhile

Fig. 12 The general structure of an ILP learning algorithm which learns by stepwise specialisation (a top-down ILP algorithm)

- For handling noisy data; i.e. not all positive examples have necessarily to be covered and there may be negative examples which could possibly be covered by the hypothesis.
- To control the hypothesis generation process beyond the requirement of being complete and consistent with the given examples. This can be considered to represent a search bias.

Figure 12 outlines the general structure often used during the hypothesis generation process in ILP (a top-down ILP algorithm).

As discussed earlier, the problem of inducing H from $\mathcal{B}$, P and N is underconstrained (see Sect. 6). In the worst case, it is possible to find an infinite number of hypotheses all of which can satisfy the required relationships. One of the ways to resolve this problem is by restricting the choice of H to be the most general hypothesis relative to $\mathcal{B}$ (as in [26]) or the least general relative to $\mathcal{B}$ (as in [27]) or such that maximum information compression of *examples* relative to $\mathcal{B}$ can be obtained (as in [35]) or apply some other criteria. The selection of a constraint will influence the types of new theories we can learn. Hence, the choice of which constraint to apply is related to the types of theories we want to learn.

Section 6.1 discusses some of the techniques used by ILP systems that can reduce the search space while inducing new hypotheses.

6.1 Biases in ILP Systems

The hypothesis language used in ILP is normally Horn-clauses. The expressive power of Horn-clauses is far more than other representations used in Machine Learning (like decision trees, propositional learners, case based reasoning, etc.) In

a Horn-clause representation, we can more easily express the domain knowledge and new possible hypotheses can also succinctly capture the required relationships in the domain. However, this comes at the cost of a large search space. In general, a number of possible hypotheses we can generate can be extremely large and in the worst case infinite. Under this circumstance, it becomes extremely important to restrict the search space so that we can use ILP systems for real world problems. All the ILP systems impose some kind of restrictions on the possible new hypotheses that they can learn. The aim is to generate all possible hypotheses that might be of interest and at the same time avoid generating hypotheses that might not prove useful. This is achieved by using different kinds of biases. A bias is a criteria used by a learner at the time of inducing new hypotheses. It may include some of the following (from [36]):

- The language used to describe hypotheses
- The space of hypotheses that the learner should consider
- The order in which hypotheses should be considered
- The acceptance criteria that define whether a search procedure may stop with a given hypothesis or should continue searching for a better choice

In general, we can categorise biases used in ILP systems as follows (from [37]).

Language Bias

Language bias specifies the syntax of the candidate hypotheses. The aim is to induce only those new hypotheses that match a given syntax. For example, we may want to induce hypotheses that are Horn-clauses and do not contain any function symbols. Alternatively, we may want to induce Horn-clauses where a number of literals in the body (negative literals) should not be greater than some fixed value n. We can also specify the types of literals that are allowed in the body and so on. For example, in MOBAL [38], we may specify the following syntax (called *rule model or* rule schema) for possible new hypotheses:

$$C(X1) \leftarrow P(X2), R(X1, X2),$$

where C, P and R can be replaced by any predicate symbols from the current theory. That means, we want to induce definite clauses such that there are two literals in the body. The above syntax also restricts the possible number of arguments we can have in each literal, and also imposes inter-dependency amongst these arguments.

GRENDEL [39] uses a special attribute description grammar to specify the types of clauses that should be considered by the learner. PROGOL [40] uses *mode declarations* that specify input/output behaviours of arguments in a predicate. It also specifies the predicate and function symbols that should be considered while inducing hypotheses. That means the learner would only generate hypotheses for a given set of predicate and function symbols. For example, in PROGOL, the following mode declaration indicates that we want to induce a predicate named "reverse," where the first argument is instantiated when it is called, and on return the second

argument will be instantiated (hopefully representing the reverse list for the first argument).

$$modeh(*, reverse(+list, -list))$$

Other examples include the *clause models* in [41], *schemata* in [42], *ij-determinacy* in GOLEM [43], etc.

Search Bias

Search bias specifies how to traverse the search space while inducing new hypotheses. For example, many algorithms in ILP use a partial-order relation that can organise hypotheses in a general to specific framework. The aim then can be to induce hypotheses that are general with respect to the current hypothesis. This will avoid considering hypotheses that are specific with respect to the current hypotheses. In a way, every algorithm provides a specific way in which the learner would search the hypotheses space.

Another approach is to induce a new theory that minimises the size of the theory along with observations. The basic idea here is that, assuming that the theory and data are optimally encoded, the best theory is the one that minimises the total cost of transmitting the combined message (theory and data) to a receiver. The receiver then can reconstruct all the original observations from this message. This approach is known as *Minimum Message Length (MML)* [44]. A related principle called *Minimum Description Length (MDL)* [35] tries to induce a theory based on its ability to compress the observations alone. Here, the size of the theory is not considered, but the measure indicates how well the theory can compress the observations.

6.2 Discussion on Inductive Logic Programming

Inductive Logic Programming (ILP) has been covered in relative length compared to other learning techniques. Reason being, that the flexible integration of domain knowledge appears very promising. It has been argued, e.g. in [45], that effective learning approaches need to utilise available domain knowledge to improve their performance: That is to allow learning from fewer examples and to improve the reliability of the learning result. Often traditional propositional approaches are not enough to handle complex data types and domain knowledge required to address many interesting real world problems. ILP provides much reacher framework to address such problems. However, currently one of the major criticisms of ILP methods is their inability to scale up to address large problems. One of the main reasons for this is a typically large search space they need to explore to learn new models. To address this problem, special techniques like language bias, search bias, etc. need to be used based on the available domain knowledge and problem requirements.

Reference [46] provides a good overview of the early ILP systems, and [47] discusses how ILP methods could be used for many relational data mining tasks. Some examples of the ILP systems for scientific knowledge discovery include, [48–53].

7 Association Rules

The process of discovering association rules involves finding possible recurrent patterns, like interesting association or correlation relationships, among a large set of data items. Here, a data item is an attribute-value pair, typically with a boolean value true indicating that a data item is present and false indicating otherwise. A set of data items represents many such items. The most common and popular example of association rules mining is *market basket analysis*. The aim here is to discover which items sell together. For example, how many times say milk and bread appear in a same "basket".

Let us consider the set of transactions in Fig. 13. Here, A, B, C and D are items and each transaction contains 1 or more of these four items.

The first task now is to find all combinations of items that occur *frequently* together. Let us say, we consider that items that appear together more than three times should be considered as *frequent* items. In other words, we say that we want the *minimum support count* to be 3 for our analysis. In the example above, the items B and C appear together four times and therefore we would consider them as a ***frequent itemset***. However, the items A and D appear together only once, and therefore we would not consider that combination as a *frequent itemset*. Generally, combinations that are not *frequent* are not explored further in the process of discovering association rules. Normally, a user defines the value of *minimum support count* based on the domain knowledge and the problem at hand. For any reasonably large data set, finding all combinations of items that occur frequently together could prove computationally very expensive, and therefore special techniques need to be used to improve the efficiency of this step.

Transactions
A, B, C
B
A, B
B, C, D
C, D
A
B, C, D
A, B
B, C
A, D

Fig. 13 A small set of transactions

Once we have all such frequently appearing items, we need to derive possible association rules from them. Let S be a frequent itemset. The set provides $\{(S \setminus X) \Rightarrow X \mid X \in S\}$ all association rules where we have one item as the consequence and the complement to S as the condition part. We can derive more rules by having more than one item in the consequence part.

Next step is to consider only those rules that are *interesting*. Normally, two measures are used for this task: ***support*** which indicates usefulness of a rule and ***confidence*** which indicates certainty of a rule.

In the above example, let us say we discover the rule $C \Rightarrow B$. Considering, the items B and C appear four times together out of the total of 10 transactions, we say that the *support* for the rule is 40%, using the following calculations:

$$support = \frac{number\ of\ transactions\ with\ both\ B\ and\ C}{number\ of\ the\ total\ transactions} = \frac{4}{10} = 0.4.$$

Now, there are five transactions with the item C, and four transactions with both the items C and B, so we say that the rule has the *confidence* value of 80%, using the following calculations:

$$confidence = \frac{number\ of\ transactions\ with\ both\ C\ and\ B}{number\ of\ transactions\ with\ C} = \frac{4}{5} = 0.8.$$

In other words, we say that 80% of the customers who buy the item C also buy the item B. Similarly, we can calculate the confidence value of the rule $B \Rightarrow C$, which is 57.14% (that is, (number of transactions with both B and C)/(number of transactions with B) $= 4/7 = 0.5714$).

Normally, we consider association rules interesting if their *support* and *confidence* values are above specific ***thresholds***. The actual values of these thresholds depend on the domain and the problem requirements.

Below we define the earlier concepts more formally from [54].

Let I represents a set of items, that is $I = \{i_1, i_2, ..., i_m\}$. Let D represents a set of transactions where each transaction T is a set of items such that $T \subseteq I$. Let A be a set of items. A transaction T is said to contain A if and only if $A \subseteq T$. An association rule is an implication of the form $A \Rightarrow B$, where $A \subset I, B \subset T$ and $A \cap B = \phi$.

The rule $A \Rightarrow B$ holds in the transaction set D with support s, where s is the percentage of transactions in D that contain both A and B. Here s represents the probability, $P(A \cup B)$.

The rule $A \Rightarrow B$ has confidence c in the transaction set D if c is the percentage of transactions in D containing A that also contain B Here, c represents the conditional probability, $P(B|A)$.

7.1 The Apriori Algorithm

In this section, we briefly outline one of the most popular algorithms, called the Apriori algorithm [55], for generating frequent itemsets for mining association rules. A simplified view of the algorithm is provided in the Fig. 14. The table in the lower part of Fig. 14 illustrates how the algorithm derives different itemsets for a simple example (from [54]). The algorithm first finds frequent itemsets with only one item, called L_1. Later L_2 (that is, itemsets with two items) is derived from L_1, and L_3 is derived from L_2, and so on until say L_k when a new derived itemset is empty.

The algorithm uses one crucial property called the *Apriori property* that indicates that all nonempty subsets of a frequent itemset must also be frequent. This means, while deriving L_i from L_{i-1}, we only need to consider generating candidates in L_i where their possible subsets of size $i-1$ already exist in L_{i-1}. This property allows the algorithm to discard many combinations of itemsets while deriving new L_i. For example, in Fig. 15, only $\{B, C, E\}$ is the possible canditate for L_3 because its subsets of size 2 are all frequent (and present in L_2). For the other possible combinations of size 3, this is not true.

Input: A set of transactions D
L_1 = find frequent itemsets with one item from D
for($k = 2; L_{k-1} \neq \phi; k{+}{+}$)
  C_k = find candidates for frequent itemsets with k items, using L_{k-1}
  L_k = take frequent itemsets from C_k with required minimum support
endfor
Output: $L_1 \cup L_2 \cup ... \cup L_k$

Min support = 2 transactions

Transactions
A, C, D
B, C, E
A, B, C, E
B, E

C_1
{A} supp 2
{B} supp 3
{C} supp 3
~~{D} supp 1~~
{E} supp 3

L_1
{A} supp 2
{B} supp 3
{C} supp 3
{E} supp 3

C_2
~~{A, B} supp 1~~
{A, C} supp 2
~~{A, E} supp 1~~
{B, C} supp 2
{B, E} supp 3
{C, E} supp 2

L_2
{A, C} supp 2
{B, C} supp 2
{B, E} supp 3
{C, E} supp 2

C_3 and L_3
{B, C, E} supp 2

Fig. 14 A simplified skeleton of the apriori algorithm and a simple example for it

for ($i = 0; i < T; i{+}{+}$)
1. Re-sample from the existing training set of N examples by bootstrapping: i.e. generate a new training sample S_i by drawing N examples randomly from the existing training set according to the uniform probability distribution among the N training examples;
2. Generate a classifier C_i from the generated sample S_i;

endfor
Classify new examples by majority vote among the T generated classifiers;

Fig. 15 Pseudocode of bagging (boosting by resampling)

The Apriori algorithm uses generate-and-test approach and often this dose not prove very efficient. Some recent association rules mining algorithms avoid using generate-and-test approach to improve the efficiency. See [54] for more discussions on this topic.

7.2 Discussion on Association Rules

Often a process of discovering association rules results in a large amount of association rules, many of which are useless or uninteresting to the user. A user can set say confidence and support thresholds, however these values do not always capture *interestingness* of associations properly. One of the alternatives is to allow a user to provide additional constraints that could be used while discovering association rules. For example, a user might be interested in discovering associations between certain items only, or generating rules for certain items only, or may not want to explore itemsets that are larger than say three items, or may not want to include some items in the analysis, etc.

It should be noted that not all strong association rules with high confidence values are always interesting. For example, let us assume that the item X exists in every transaction of the set D. Now, say we discover the following association rule, $A \Rightarrow X$ with confidence value of 100%. If we only consider the confidence value here (that is 100%), the rule seems impressive. However, considering that the item X exists in every transaction, this rule and many other similar rules are uninteresting to the user because he or she perhaps already knows that every one buys X anyway. We can address this problem by calculating the *lift* value of a rule. The *lift* value tries to measure the strength of the implication, and can be calculated by dividing the confidence by the support. In the above example, the lift value would be 1, indicating the implication has no effect. A lift value of greater than one indicates positive influence, and a value of less than one indicates negative influence on the consequence of the rule.

The above example illustrates that often we also need to use other measures like *lift*, *correlation analysis*, etc. to search for interesting association rules. See [24, 54, 56, 57] for more recent developments.

8 Naive Bayesian Classifiers

The Naive Bayesian classifier is a simple but very effective learning technique for many practical problems. As the name suggests it is based on Bayes Theorem:

$$P(h|D) = \frac{P(D|h) * P(h)}{P(D)}$$

This Theorem allows to calculate the conditional probability for the hypothesis h given the data D on the basis of a priori probabilities for both and the inverse conditional probability $P(D|h)$. In many practical situations, it is easier to obtain estimates for the three probabilities on the right-hand side than estimating the probability on the left-hand side directly. For the problem of classification, a probabilistic perspective is taken. That is, there are certain probabilities to be determined according to which a given object belongs to a given class.

That is, we generally want to estimate the probability $P(Class(X) = c_i|X)$, where X is the object to be classified and $Class(X)$ is its class. So, we would choose the class c_i for which the estimated conditional probability is maximum among all possible classes.

For the Naive Bayesian Classifier the Theorem of Bayes is used to estimate $P(Class(X) = c|X)$ and a Naive assumption is being made to estimate the reverse conditional probability needed on the right-hand side of the Theorem's equation.

So, a Bayesian Classifier would classify an object X into the class c_i, where

$$c_i = \operatorname*{argmax}_c P(Class(X) = c|X)$$

$$= \operatorname*{argmax}_c \frac{P(X|Class(X) = c) * P(Class(X) = c)}{P(X)}$$

Since we are only interested to know for which class the fraction above becomes maximum, we don't need to know $P(X)$. So, we need to estimate the a priori probability for an object belonging to a given class and the conditional probability $P(Class(X) = c|X)$. The a priori probability is estimated by simple frequency counting of the various classes in the given sample of training data.

So, what is left to estimate is $P(Class(X) = c|X)$. This is usually impractical to estimate since for each object in the universe there are usually very few, if any, examples in the training sample resulting in useless estimates for the conditional probabilities.

Hence, the Naive Bayesian Classifier uses the following Naive assumption:

The object X has a number of attributes. For each class c the attributes have a certain probability to take certain values. This probability is assumed to be conditionally independent of all other attribute values, given the class c. This allows us to use the following formula to estimate $P(X|Class(X) = c)$:

$$P(X|Class(X) = c) = \prod_i P(Att_i(X) = a_i|Class(X) = c),$$

where i ranges over all attributes of X and the function $Att_i(X)$ produces the value of the ith attribute of X.

The multiplication of the conditional probabilities

$$\prod_i P(Att_i(X) = a_i | Class(X) = c)$$

for all attributes is only valid, if indeed the probability of an attribute having the value a_i given the object belongs to class c is conditionally independent (given the object belongs to c) of all other attributes.

This leads us to the following formula to determine the most likely class c_i for an object X:

$$c_i = \underset{c}{\operatorname{argmax}}\, P(Class(X) = c) * \prod_i P(Att_i(X) = a_i | Class(X) = c).$$

All probabilities on the left-hand side are simply estimated by the respective relative frequency in the training data set. While the mentioned conditional independence assumption above will almost always be incorrect, the results of the Naive Bayesian Classifier are usually quite useful. This can mainly be explained by the fact that the Naive Bayesian Classifier does not need to get the probability estimates right but merely to rank the various possible classes correctly. That is, even if the numbers are incorrect, the obtained ranking may still deliver the correct class as the one with the highest estimated probability for the given object.

While the Naive Bayesian Classifier is a popular and practical learning technique it has been less stressed in the context of this book as the resulting classifier is more difficult to interpret for humans. The interpretability of symbolic learning results rather than numerical ones is often higher and hence more applicable to many problems in scientific discovery.

9 Improving Classifiers by Using Committee Machines

It has been found that combining multiple classifiers which were obtained by learning from multiple different but related training data sets often leads to better accuracy than training only a single classifier. Those multiple data sets are obtained by resampling techniques, i.e. constructing from the given training data new training data, e.g. by choosing different subsets of the original training data set. The combination of multiple such classifiers has been demonstrated to substantially improve the accuracy of the combined classifier in many domains, see e.g. [58, 59].

These findings are intriguing as well as useful. It appears that, roughly speaking, a single classifier tends to have weaknesses in its classification accuracy in some area of the input space. Different classifies, even if trained on the same training data seem to exhibit weaknesses in other areas of the input space. This appears to allow

a combination of multiple classifiers to patch the weaknesses of individual ones and resulting in a reduced overall error.

A number of techniques have been devised to generate different classifiers from a given set of training data. Also a number of different ways of combining generated classifiers have been proposed and are used in today's practice.

In Sect. 9.1, we sketch two approaches, bagging and boosting to provide concrete and practical techniques in this area.

9.1 Bagging

Bagging [59] generates a number of classifiers and simply lets them all vote on the classification of a new instance by weighing all classifiers equally. The generation of the different classifiers, usually 30–50 classifiers are generated, is done using bootstrap resampling (or bootstrapping) by drawing randomly n objects from the original training set containing n objects itself. That is, the resulting new sample would contain some objects multiple times while other objects from the original training data may not occur in the newly drawn sample. In fact, this is also known as a 0.632 bootstrap sample, as it usually contains only around 63.2% of the objects in the original training data set. The following pseudo-code shows the entire procedure using a 0.632 bootstrap sample.

9.2 AdaBoost

The AdaBoost [24,60] algorithm differs from bagging in two ways. On the one hand it weighs the different classifiers differently, the higher the fraction of the training data they classify correctly the higher their weight in voting on the classification of a new instance. The generation of multiple classifiers also differs from Bagging by giving different weights to the training instances. In order to draw the "attention" of the next classifier to be generated on the weaknesses of the already generated classifiers, it gives higher weight to those training instances that are incorrectly classified by the classifiers already generated. Thus, making it more likely that the new classifier being generated will classify those instances correctly that are incorrectly classified so far. In Fig. 16, we provide more details of the procedure by way of pseudo-code.

10 Discussion on Learning

Implementations of the techniques presented in this chapter as well as others, including the ones mentioned later, can be found in the Open Source Software WEKA [61] and in the system ALEPH for an ILP system. Both can be easily found for download on the WWW.

Input: Training sample $(x_i, d_i)_{i=1}^{N}$, Distribution D over the sample, number of iterations T
Initialisation: Set $D_1(i) = \frac{1}{N}$ for all i;
Main Loop: **for** $(n = 1; n < T; n{+}{+})$
Generate hypothesis F_n based on D_n and calculate error: $\epsilon_n = \sum_{i \in \{i \mid F_n(x_i)\} \neq d_i} D_n(i)$;
Set $\beta_n = \frac{\epsilon_n}{1-\epsilon_n}$;
Set $D_{n+1}(i) = \frac{D_n(i)}{Z_n} \times a$, where $a = \beta_n$ if $F_n(x_i) = d_i$; $a = 1$ otherwise;
(Z_n being normalisation constant).
endfor
Output: $F_n(x) = \text{argmax}_d \sum_{n:F_n(x)=d} \log \frac{1}{\beta_n}$;

Fig. 16 Pseudocode of AdaBoost

Some learning techniques discover equations that fit available data, see e.g. [62] for an example where ILP is also used. Those techniques are discussed in other chapters of this book in detail. Besides that, a number of other popular learning techniques exist. These include broad classes of techniques such as Neural Networks [63], Support Vector Machines [64, 65], k-means [24, 66], EM [24, 67], and others.

Those techniques usually determine a large number of numerical parameters of some more or less involved function. This makes it normally rather difficult for a human to understand and interpret the results. For these reasons we do not provide details on these techniques but rather refer the interested reader to other literature such as the above cited ones.

As we have seen in this chapter, a large number of approaches have been developed to construct concept descriptions or classification rules or, more generally speaking, functions and relations from a number of instances of the learning target. A number of these approaches have been usefully employed, while many others are still in the realm of research and will hopefully lead to useful applications at a later stage.

One can generally say that the success and failure of using a learning technique for a given problem depends crucially on choosing a suitable representation of the training data and choosing a suitable learning technique that searches in a reasonably restricted set of potential learning targets (or hypothesis space). On the other hand, the representation of the training data as well as the choice of the learning technique may actually prevent the learner from being able to express a suitable concept or function as the learning result. Similarly, if the space of potential target concepts is too large, the available data may not suffice to rule out all those candidate concepts which do not represent acceptable solutions to the learning task. Another issue that may hamper the practical success of employing machine learning techniques is the computational cost it may take to process large amounts of training data.

There are a number of other research directions not discussed in this section. Those include model selection, i.e. the idea of splitting the hypothesis space into multiple sub-hypothesis spaces and then to apply certain techniques to pick a good

sub-hypothesis space for searching for a most consistent hypothesis, see e.g. [68,69] or [70]. Other approaches to learning include lazy learners, i.e. learners which merely store the data they receive without processing. Only when a demand of classifying or reacting in other ways occurs, lazy learners start processing their stored data to find an answer. Nearest neighbour classifiers are probably the best-known class of algorithms of this kind. See e.g. [71] for a somewhat dated but still useful survey.

Other research is devoted to problems of sequential decision making where feedback from the environment may be received only in a delayed fashion, i.e. only after a possibly large number of decisions has been made the overall sequence is evaluated. A robot on the search for a new power supply may be a scenario for that. Reinforcement learning techniques, such as Temporal difference learning [72] or Q-learning [73] and further refinements of those have been developed to deal with such problems [74, 75].

Progress on the integration of learning techniques with already available data and knowledge in a system is needed. Another related area of significant interest is *knowledge discovery and datamining* (KDD), also often called *datamining*. The objective here is to integrate techniques from the fields of database systems, machine learning, and statistics to be applied to large amounts of data.

Using machine learning techniques in automatic scientific discovery is still in its infancy. However, given the enormous amount of data already available today, machine learning techniques will play an increasingly important role as human scientists are already overwhelmed by the available data as well as by the amount of published research.

References

1. J.L. Kolodner, *Case-Based Reasoning* (Morgan Kaufmann, CA, 1993)
2. D.B. Leake (ed.), *Case-Based Reasoning – Experiences, Lessons, and Future Directions* (MIT, MA, 1996)
3. R.S. Michalski, Artif. Intell. **20**, 111–161 (1983)
4. T.M. Mitchell, Artif. Intell. **18**, 203–226 (1982)
5. P.E. Utgoff, T.M. Mitchell, Acquisition of Appropriate Bias for Inductive Concept Learning, in *Proceedings of the AAAI*, pp. 414–417, 1982
6. J.R. Quinlan, *C4.5: Programs for Machine Learning* (Morgan Kaufmann, CA, 1993)
7. R.S. Michalski, On the Quasi-Minimal Solution of the General Covering Problem, in *Proceedings of the 5th International Symposium on Information Processing (FCIP 69)*, vol. A3, Liphhok, 1969, pp. 125–128
8. R.S. Michalski, J. Larson, Incremental Generation of vl_1 Hypotheses: The Underlying Methodology and the Description of Program. Technical Report ISG 83-5, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IN, 1983
9. P. Clark, T. Niblett, Mach. Learn. **3**(4), 261–283 (1989)
10. M.R. Garey, D.S. Johnson, *Computers and Intractability* (Freeman, San Francisco, 1979)
11. J. Wojtusiak, R.S. Michalski, K. Kaufman, J. Pietrzykowski, The AQ21 Natural Induction Program for Pattern Discovery: Initial Version and Its Novel Features, in *Proceedings of The 18th IEEE International Conference on Tools with Artificial Intelligence*, Washington DC, 2006
12. E.B. Hunt, J. Mairn, P.J. Stone, *Experiments in Induction* (Academic Press, New York, 1966)

13. J.R. Quinlan, in *Discovering Rules by Induction from Large Collections of Examples*, ed. by D. Michie. Expert Systems in the Microelectronic Age (Edinburgh University Press, Edinburgh, 1979)
14. L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, *Classification and Regression Trees* (Wadsworth, 1984)
15. W. Müller, F. Wysotzki, *The Decision Tree Algorithm CAL5 Based on a Statistical Approach to Its Splitting Algorithm.* in Machine Learning and Statistics (Wiley, New York, 1996)
16. I. Bratko, *Prolog: Programming for Artificial Intelligence*, 3rd edn. (Addison-Wesley Longman Publishing, Boston, MA, 2001)
17. C.E. Shannon, W. Weaver, *The Mathematical Theory of Communication* (University of Illinois Press, Urbana, Illinois, 1949)
18. J.R. Quinlan, J Artif. Intell. Res. **4**, 77–90 (1996)
19. I.J. Good, W.I. Card, Meth. Inf. Med. **3**(10), 176–188 (1971)
20. J.R. Quinlan, Combining Instance-Based and Model-Based Learning, in *Proceedings of the 10th International Workshop on Machine Learning*, Kaufmann, 1993, pp. 236–243
21. B. Cestnik, I. Kononenko, I. Bratko, in *ASSISTANT86: A Knowledge-Elicitation Tool for Sophisticated Users*, ed. by I. Bratko, N. Lavrac. Progress in Machine Learning (Sigma Press, Wilmslow, 1987)
22. J.R. Quinlan, in *The Effect of Noise on Concept Learning*, ed. by R.S. Michalski, J.G. Carbonell, T.M. Mitchell. Machine Learning: An Artificial Intelligence Approach, vol. 2 (Morgan Kaufmann, CA, 1986)
23. J.R. Quinlan, Int. J. Man Mach. Stud. **27**, 221–234 (1987)
24. X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, Z.-H. Zhou, M. Steinbach, D.J. Hand, D. Steinberg, Knowl. Inf. Syst. **1**, 1–37 (2008)
25. R.S. Michalski, in *A Theory and Methodology Of Inductive Learning*, ed. by R.S. Michalski, J.G. Carbonell, T.M. Mitchell. Machine Learning: An Artificial Intelligence Approach, vol. I (Morgan Kaufmann Publishers, CA, 1983), pp. 83–134
26. E. Shapiro, *Algorithmic Program Debugging* (MIT, Cambridge, MA, 1983)
27. G. Plotkin, in *A Note on Inductive Generalization*, ed. by B. Meltzer, D. Michie. Machine Intelligence 5 (Edinburgh University Press, Edinburgh, 1969), pp. 153–163
28. C. Sammut, R. Banerji, in *Learning Concepts by Asking Questions*, ed. by R.S. Michalski, J.G. Carbonell, T.M. Mitchell. Machine Learning: An Artificial Intelligence Approach, vol. 2 (Morgan Kaufmann, CA, 1986), pp. 167–191
29. S. Džeroski, N. Lavrač, in *An Introduction To Inductive Logic Programming*. Relational Data Mining (Springer, New York, 2001), pp. 48–71
30. J.W. Lloyd, *Foundations of Logic Programming* (Springer, Berlin, 1987)
31. S. Muggleton, W. Buntine, Machine Invention Of First-Order Predicates By Inverting Resolution, in *Proceedings of International Workshop on Machine Learning* (Kaufmann, CA, 1988), pp. 339–352
32. N. Nilsson, M.R. Genesereth, *Logical Foundations of Artificial Intelligence* (Springer, Berlin, 1987)
33. S. Muggleton, New Generat. Comput. **4**(8), 295–318 (1991)
34. J.R. Quinlan, R.M. Cameron-Jones, Foil: A Midterm Report, in *ECML '93: Proceedings of the European Conference on Machine Learning*, Springer, London, UK, 1993, pages 3–20
35. J. Rissanen, Ann. Stat. **11**, 416–431 (1983)
36. P.E. Utgoff, in *Shift of Bias for Inductive Concept Learning*, ed. by R.S. Michalski, J.G. Carbonell, T.M. Mitchell. Machine Learning: An Artificial Intelligence Approach, vol. II (Kaufmann, Los Altos, CA, 1986)
37. C. Nedellec, C. Rouveirol, H. Ade, F. Bergadano, B. Tausend, Declarative bias in ILP, 1996
38. K. Morik, K. Causse, R. Boswell, A Common Knowledge Representation For Integrating Learning Tools, in *Proceedings of 1st International Workshop on Multistrategy Learning*, 1991, pp. 81–96
39. W.W. Cohen, Rapid Prototyping of ilp Systems Using Explicit Bias, in *Proceedings of the IJCAI-93 Workshop on Inductive Logic Programming*, 1993

40. S. Muggleton, New Generat. Comput. **13**(3–4), 245–286 (1995) (Special issue on Inductive Logic Programming)
41. H. Adé, L. De Raedt, M. Bruynooghe, Mach. Learn. **20**, 119–154 (1995)
42. B. Tausend, in *Representing Biases for Inductive Logic Programming*, ed. by F. Bergadano, L.D. Raedt. ECML, vol. 784 of Lecture Notes in Computer Science (Springer, Berlin, 1994), pp. 427–430
43. S. Muggleton, C. Feng, Efficient Induction Of Logic Programs, in *Proceedings of the 1st Conference on Algorithmic Learning Theory* (Ohmsma, Tokyo, Japan, 1990), pp. 368–381
44. C. Wallace, *Statistical and Inductive Inference by Minimum Message Length (Information Science and Statistics)* (Springer, New York, 2005)
45. A. Hoffmann, *Paradigms of Artificial Intelligence – A Methodological and Computational Analysis* (Springer, Berlin, 1998). ISBN 981-3083-97-2
46. N. Lavrac, S. Džeroski, *Inductive Logic Programming: Techniques and Applications* (Routledge, New York, 1993)
47. S. Džeroski, N. Lavrač (eds.), *Relational Data Mining* (Springer, New York, 2001)
48. A. Srinivasan, R.D. King, S. Muggleton, M.J.E. Sternberg, Carcinogenesis Predictions Using ILP, ed. by S. Džeroski, N. Lavrač. in *Proceedings of the 7th International Workshop on Inductive Logic Programming*, vol. 1297, Springer, Berlin, 1997, pp. 273–287
49. A. Srinivasan, D. Page, R. Camacho, R.D. King Mach. Learn. **64**(1–3), 65–90 (2006)
50. H. Lodhi, S. Muggleton, Modeling metabolic Pathways Using Stochastic Logic Programs-based Ensemble Methods, in *Proceedings of the 2nd International Conference on Computational Methods in System Biology*, Springer, Berlin, 2004
51. S. Muggleton, Machine Learning for Systems Biology, in *Proceedings of the 15th International Conference on Inductive Logic Programming*, LNAI 3625, Springer, Berlin, 2005, pp. 416–423
52. A. Tamaddoni-Nezhad, R. Chaleil, A. Kakas, S. Muggleton, Abduction and Induction for Learning Models of Inhibition in Metabolic Networks, in *Proceedings of the 4th International Conference on Machine Learning and Applications, ICMLA'05*, IEEE Computer Society, 2005
53. A. Mahidadia, P. Compton, Assisting Model-Discovery In Neuroendocrinology, in *DS '01: Proceedings of the 4th International Conference on Discovery Science*, Springer, London, UK, 2001, pp. 214–227
54. J. Han, M. Kamber, *Data Mining: Concepts and Techniques* (Morgan Kaufmann Publishers, San Francisco, 2001)
55. R. Agrawal, T. Imielinski, A. Swami, Mining association rules between sets of items in large databases, in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Washington DC, May 1993, pp. 207–216
56. T.Scheffer, Intell. Data Anal. **9**(4), 381–395 (2005)
57. S. Kotsiantis, D. Kanellopoulos, GESTS Int. Trans. Comput. Sci. Eng. **32**(1), 71–82 (2006)
58. J.R. Quinlan, Bagging, Boosting, and C4.5, in *Proceedings of the 14th National Conference of Artificial Intelligence*, 1996
59. L. Breiman, Mach. Learn. **2**(24), 123–140 (1996)
60. Y. Freund, R. Shapire, J. Comput. Syst. Sci. **55** (1997)
61. I. Witten, E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edn. (Morgan Kaufmann, CA, 2005)
62. S. Dzeroski, L. Todorovski, J. Intell. Inf. Syst. **4**, 89–108 (1995)
63. G.P. Zhang, IEEE Trans. Syst. Man Cybern. C **30**(4), 451–462 (2000)
64. V. Vapnik, *The Nature of Statistical Learning Theory* (Springer, Berlin, 1995). ISBN 0-387-98780-0
65. B. Schölkopf, A. Smola, *Learning with Kernels* (MIT, Cambridge, MA, 2002). ISBN 0-262-19475-9
66. J. Hartigan, M.A. Wong, Appl. Stat. **28**(1), 100–108 (1979)
67. A. Dempster, N. Laird, D. Rubin, J. Roy. Stat. Soc. B **1**(39), 1–38 (1977)
68. M. Kearns, Y. Mansour, A. Ng, D. Ron, Mach. Learn. **7**, 7–50 (1997)
69. R. Kohavi, G. John, Artif. Intell. **1–2**(97), 245–271 (1997)
70. K.P. Burnham, D.R. Anderson, *Model Selection and Multi-Model Inference – A Practical Information Theoretic Approach* (Springer, Berlin, 2006)

71. D. Aha, Artif. Intell. Rev. **11**, 7–10 (1997)
72. R.S. Sutton, Mach. Learn. **3**(1), 9–44 (1988)
73. C.J. Watkins, *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989
74. L.P. Kaelbling (ed.), *Recent Advances in Reinforcement Learning* (Kluwer Academic Publishers, Dordrecht, 1996)
75. R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction* (MIT, CA, 1998)
76. S. Weiss, N. Indurkhya, Decision Tree Pruning: Biased or Optimal? in *Proceedings of the 12th National Conference of Artificial Intelligence*, pp. 626–632, 1994
77. S. Weiss, C. Kulikowski, *Computer Systems that Learn* (Morgan Kaufmann, CA, 1991)

Statistical Inference

Shahjahan Khan

1 Introduction

Often scientific information on various data generating processes are presented in the from of numerical and categorical data. Except for some very rare occasions, generally such data represent a small part of the population, or selected outcomes of any data generating process. Although, valuable and useful information is lurking in the array of scientific data, generally, they are unavailable to the users. Appropriate statistical methods are essential to reveal the hidden "jewels" in the mess of the row data. Exploratory data analysis methods are used to uncover such valuable characteristics of the observed data.

Statistical inference provides techniques to make valid conclusions about the unknown characteristics or parameters of the population from which scientifically drawn sample data are selected. Usually, statistical inference includes estimation of population parameters as well as performing test of hypotheses on the parameters. However, prediction of future responses and determining the prediction distributions are also part of statistical inference. Both Classical or Frequentists and Bayesian approaches are used in statistical inference. The commonly used Classical approach is based on the sample data alone. In contrast, increasingly popular Beyesian approach uses prior distribution on the parameters along with the sample data to make inferences. The non-parametric and robust methods are also being used in situations where commonly used model assumptions are unsupported.

In this chapter, we cover the philosophical and methodological aspects of both the Classical and Bayesian approaches. Moreover, some aspects of predictive inference are also included. In the absence of any evidence to support assumptions regarding the distribution of the underlying population, or if the variable is measured only in ordinal scale, non-parametric methods are used. Robust methods are employed to avoid any significant changes in the results due to deviations from the model

S. Khan (✉)
Department of Mathematics and Computing, Australian Centre for Sustainable Catchments, University of Southern Queensland, Toowoomba, QLD, Australia
e-mail: khans@usq.edu.au

M.M. Gaber (ed.), *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*, DOI 10.1007/978-3-642-02788-8_3,

assumptions. The aim is to provide an overview of the scientific domain of statistical inference without going in to minute details of any particular statistical method. This is done by considering several commonly used multivariate models, following both normal and non-normal, including elliptically contoured, distributions for the responses.

2 Approaches to Statistical Inference

A specific statistical method is not appropriate for every data set and under every situation. Statisticians have developed different methods to cater for various needs. Depending on the type of information and the nature of the data appropriate method may be chosen from available alternatives. For example, if assumptions of parametric methods are not satisfied by the data the user would pursue an appropriate non-parametric method. If the data has too many extreme values or outliers influencing the results significantly a robust method is essential. If the population distribution is symmetric with heavy or flat tails than normal distribution then a Student-t model is more appropriate to represent the data from any such models.

2.1 Parametric Inference

Classical statistical models are usually based on the assumption of some specific distribution of the underlying variable. Such distributions are characterized by parameter(s) that completely specify the distribution. Often these parameters are unknown quantities and need to be estimated from available sample. In the Frequentist approach, the parameters are considered as constants, usually unknown. So, the inference on the parameters solely depends on the sample data. On the other hand, in Bayesian approach the parameters are treated as random variables, and hence follow some distribution of their own. Inferences under this approach are obtained by mixing the sample information with the prior distribution on the parameters. So, philosophically the two approached are quite different. However, as the sample size increases the impact of the prior believe on the ultimate results decreases, and hence for large samples the results produced by both approaches are very close, if not identical.

Both classical and Bayesian methods use various statistical models. Many of these models are linear in parameters and are known as linear models. Often non-linear models can be transformed to linear models, and apply the methods applicable for linear models. A generalised linear model encompassing different commonly used linear models is given below.

Generalized Linear Model

The multivariate multiple regression model is introduced here. Let $\boldsymbol{y}_j$ be a p-dimensional column vector of the values of the jth realization associated with a set of values of the explanatory variables $x_1, x_2, \ldots, x_k$ from a multivariate multiple regression model. Then, $\boldsymbol{y}_j$ can be represented by the set of linear equations

$$\boldsymbol{y}_j = \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 x_{1j} + \boldsymbol{\beta}_2 x_{2j} + \cdots + \boldsymbol{\beta}_k x_{kj} + \Gamma \boldsymbol{e}_j \quad \text{for } j = 1, 2, \ldots, n, \quad (1)$$

where $\boldsymbol{\beta}_0, \boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_k$ are the p-dimensional regression parameters, Γ is a $p \times p$ scale parameter matrix assuming values in the positive half of the real line $\mathcal{R}^+$, and $\boldsymbol{e}_j$ is the vector of error variables associated with the responses $\boldsymbol{y}_j$. Assume that each component of the error vector, $\boldsymbol{e}_j$, is identically and independently distributed (i.i.d.) as a normal variable with location 0 and scale 1, that is, $\boldsymbol{e}_j \sim N(\boldsymbol{0}, I_p)$ in which I_p is an identity matrix of order p. The equation in (2.1) can be written in a more convenient form as

$$\boldsymbol{y}_j = \boldsymbol{\beta} z_j + \Gamma \boldsymbol{e}_j, \quad (2)$$

where $\boldsymbol{\beta} = [\boldsymbol{\beta}_0, \boldsymbol{\beta}_1, \ldots \boldsymbol{\beta}_k]$, a $p \times m$ dimensional matrix of regression parameters with $m = k + 1$; and $z_j = [1, x_{1j}, x_{2j}, \ldots, x_{kj}]'$, an $m \times 1$ dimensional design matrix of known values of the regressor for $j = 1, 2, \ldots, n$. Therefore, the joint density function of the error vector $\boldsymbol{e}_j$ is

$$f(\boldsymbol{e}_j) = [2\pi]^{-\frac{p}{2}} e^{-\frac{1}{2}\boldsymbol{e}_j' \boldsymbol{e}_j}, \quad (3)$$

and that of the response vector $\boldsymbol{y}_j$ is

$$f(\boldsymbol{y}_j | \boldsymbol{\beta}, \Gamma) = [2\pi]^{-\frac{p}{2}} |\Sigma|^{-1} e^{-\frac{1}{2}(\boldsymbol{y}_j - \boldsymbol{\beta} z_j)' \Sigma^{-1} (\boldsymbol{y}_j - \boldsymbol{\beta} z_j)} \quad (4)$$

where $\Sigma = \Gamma \Gamma'$, the covariance matrix of the response vector $\boldsymbol{y}_j$. Now, a set of $n > p$ responses, $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n]$, from the earlier multivariate multiple regression model can be expressed as

$$\boldsymbol{Y} = \boldsymbol{\beta} \boldsymbol{Z} + \Gamma \boldsymbol{E}, \quad (5)$$

where $\boldsymbol{Y}$ is the response matrix of order $p \times n$; $\boldsymbol{Z} = [z_1, z_2, \ldots, z_n]$ is an $m \times n$ dimensional design matrix of known values of the regressors; and $\boldsymbol{E}$ is a $p \times n$ dimensional matrix of the random error components associated with the response matrix $\boldsymbol{Y}$. It may be noted here that the non-conventional representation of responses as row vectors, rather than column vectors, is adopted in line with [1, 2] to facilitate straightforward comparison of results.

If each of the p-dimensional response column vector follows a multivariate normal distribution, the joint density function of the $p \times n$ order response matrix $\boldsymbol{Y}$ is given by

$$f(\boldsymbol{Y} | \boldsymbol{\beta}, \Sigma) = [2\pi]^{-\frac{pn}{2}} |\Sigma|^{-\frac{n}{2}} e^{-\frac{1}{2} tr\left\{\Sigma^{-1} (\boldsymbol{Y} - \boldsymbol{\beta} \boldsymbol{Z})(\boldsymbol{Y} - \boldsymbol{\beta} \boldsymbol{Z})'\right\}}, \quad (6)$$

where $tr(\Omega)$ is the trace of the matrix Ω. This is the p-dimensional matric normal distribution. It may be noted that if $p = 1$ the response matrix becomes an n-dimensional vector, $\boldsymbol{y}$, and its distribution turns out to be a multivariate normal distribution.

Special Cases

1. If $k = 1$ the model in (2.1) becomes a multivariate simple regression model with p responses associated with a single regressor. That is, we get

$$\boldsymbol{y}_j = \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 x_{1j} + \Gamma \boldsymbol{e}_j \quad \text{for } j = 1, 2, \ldots, n. \tag{7}$$

2. When $p = 1$ the model in (2.1) becomes a multiple regression model, representing a single response corresponding to k regressors, as follows:

$$y_j = \boldsymbol{\beta}_0 + \beta_1 x_{1j} + \beta_2 x_{2j} + \cdots + \beta_k x_{kj} + \gamma e_j \quad \text{for } j = 1, 2, \ldots, n', \tag{8}$$

   where γ is a non-zero positive scale factor.
3. Finally, if $p = 1$ and $k = 1$ the model (2.1) reduces to a simple regression model with one response and one explanatory variable. Thus, we get

$$y_j = \beta_0 + \beta_1 x_{1j} + \gamma e_j \quad \text{for } j = 1, 2, \ldots, n. \tag{9}$$

Thus, the model in (2.5) is a generalization of different linear models commonly used in analysing many data sets arising from varieties of real-life problems.

Generalizing Symmetric Distribution

The density function of the response matrix $\boldsymbol{Y}$ in (2.5) for non-normal, but symmetric, distributions can also be specified when errors are not normally distributed. For example, if each of the p-dimensional response column vector follows a multivariate elliptical distribution, the joint distribution of the $p \times n$ order response matrix $\boldsymbol{Y}$ is a matrix variate elliptically symmetric distribution with density function

$$f(\boldsymbol{Y}|\boldsymbol{\beta}, \boldsymbol{\Sigma}) \propto |\boldsymbol{\Sigma}|^{-\frac{n}{2}} g\left\{tr\left[\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y} - \boldsymbol{\beta}\boldsymbol{Z})(\boldsymbol{Y} - \boldsymbol{\beta}\boldsymbol{Z})'\right]\right\}, \tag{10}$$

where $g(\cdot)$ is such that $f(\cdot)$ is a proper density function. For details on spherically and elliptically symmetric distributions, see [3–6] for instance. Some of the well-known members of the spherically/elliptically symmetric family of distributions are the normal, Kotz Type, Pearson Type VII, Student-t, Cauchy, Pearson Type II, Logistic, Bassel, Scale mixture, and Stable laws. Note that elliptically symmetric family of distributions can be defined for both multivariate and matrix-variate cases. Specific choice of the form of $g(\cdot)$ would lead to particular member of the family of elliptically symmetric distribution. For instances, if $g(\cdot) = \exp[\cdot]$, the distribution becomes matrix normal, and if $g(\cdot) = \{I + [\cdot]\}$ the distribution becomes matrix T.

2.2 Nonparametric Inference

In general, Classical and Bayesian methods are based on assumptions which are not always met in practice. Unfortunately, the errors or responses are not always known to follow any specific distribution. Also, the validity of the central limit theorem is not always guaranteed. So, if there are outliers in the data, classical methods are unreliable. Moreover, if the data is in the rank order the conventional parametric methods are inappropriate. Although robust methods are classical methods, the results are not unduly affected by outliers or other small departures from model assumptions. References on robust statistics include [7–9]. A modern treatment is given by [10].

The main technique of robust method aims at reducing the weights (and hence impact) of extreme values and outliers to reduce or eliminate their undue influence on the results.

Robust parametric statistics tends to rely on replacing the normal distribution in classical methods with the Student-t distribution with low degrees of freedom (high kurtosis; degrees of freedom between four and six have often been found to be useful in practice) or with a mixture of two or more distributions.

2.3 Prediction Analysis

The prediction methods deal with inference for the future responses, rather than the model parameters, using the data on the realized responses. The prediction distribution forms the basis of all predictive inference. Prediction distribution is a conditional distribution of the unobserved future responses, conditional on the realized responses. Such a distribution is free from any distributional parameter of the model, and depends only on the sample information. This was the oldest statistical method used by scientists before the estimation and test on parameters were introduced [11].

Predictive inference uses the realized responses from the *performed experiment* to make inference about the behavior of the unobserved future responses of the *future experiment* (cf. [12]). The outcomes of the two experiments are connected through the same structure of the model and indexed by the common set of parameters. For details on the predictive inference methods and wide range of applications of prediction distribution interested readers may refer to [11, 13]. Predictive inference for a set of future responses of a model, conditional on the realized responses from the same model, has been derived by many authors including [1, 12, 14–16]. The prediction distribution of a set of future responses from the model has been used by [17] to derive β-expectation tolerance region. References [12, 18] obtained different tolerance regions from the prediction distribution.

There are various approaches to predictive inference. In the Bayesian approach, the likelihood of the unobserved future responses are mixed with the posterior distribution from the realized responses to derive the prediction distribution.

The structural distribution replaces the posterior distribution under the structural approach (cf. [1, 2]). [19] used the structural relation approach to find the prediction distribution for linear models without going through the structural distribution. Various types of tolerance regions are derived based on the prediction distribution. Although traditional methods cover prediction for future responses, [20, 21] proposed prediction distribution for the future regression parameter and future residual sum-of-squares for linear model.

3 Classical Inference

In the Classical or Frequentist approach, inferences on the parameters are made from the sample data alone. Here, the parameters of the models are assumed to be fixed unknown quantities. Estimation of these unknown quantities or testing regarding any specific value of the parameters depend on the sample data. The likelihood principle and method are often applied to derive appropriate estimator and tests in this approach. These are popularly known as maximum likelihood estimators and likelihood ratio tests, respectively. However, other principles and methods, such as least squares, minimum distance, etc., are also available. The classical inference, often, is exclusively about parameters and solely depends on the sample data. However, non-parametric methods of estimation and tests are also available. But these are also based on the sample data only. Uncertain non-sample prior information (NSPI) on the value of the parameters along with the sample data are also used in improved estimation, particularly for preliminary test and Stein-type shrinkage estimations. Reference [22] explores varieties of techniques and studies the properties of different estimators for a wide range of models.

3.1 Inference for Multiple Regression Model

For the parametric inference most commonly used model assumes normal distribution for the errors or responses. Under this model, often it is assumed that the errors are independent and identically distributed. These assumptions make the mathematical derivation of estimators and tests easier and simpler. However, other models are also used when the normal model is inappropriate. For modeling variables that follow symmetric distributions but have heavier tails than a normal distributions, Student-t model is appropriate. A more general class of symmetric distributions is represented by the class of elliptically symmetric models. The normal and Student-t models are special cases of the class of elliptically symmetric model.

For explicit analysis, consider a multiple regression model

$$y_j = \boldsymbol{\beta_0} + \beta_1 x_{1j} + \beta_2 x_{2j} + \cdots + \beta_k x_{kj} + \sigma e_j \quad \text{for } j = 1, 2, \ldots, n; \qquad (11)$$

or equivalently,

$$\boldsymbol{y} = X\boldsymbol{\beta} + \sigma\boldsymbol{e}, \tag{12}$$

where $\boldsymbol{y}$ is the n-dimensional vector of responses, and X is the design matrix of order $n \times p$ in which $k = (p-1)$ is the number of regressors or predictors in the model.

The least-square (LS) principle requires that the estimator of the parameter vector to be $\tilde{\boldsymbol{\beta}}$ such that $(\boldsymbol{y} - \tilde{\boldsymbol{\beta}}X)'(\boldsymbol{y} - \tilde{\boldsymbol{\beta}}X)$ is minimum. The calculus method yields $\tilde{\boldsymbol{\beta}} = (X'X)^{-1}X'\boldsymbol{y}$. Similarly, the LS estimator of the covariance matrix is $\tilde{\sigma}^2 I_n$ in which $\tilde{\sigma}^2 = \frac{1}{n-p}(\boldsymbol{y} - \tilde{\boldsymbol{\beta}}X)'(\boldsymbol{y} - \tilde{\boldsymbol{\beta}}X)$. These estimators are robust as the results are not dependent on the assumption of normality of the errors or responses. Often these estimators are called *unrestricted estimator* (UE) as no restrictions on the parameters is imposed in the estimation of the parameters. By the central limit theorem (CLT), $\tilde{\boldsymbol{\beta}} \sim N\left(\boldsymbol{\beta}, [X'X]^{-1}\sigma^2\right)$, if n is large.

Assuming that the errors are identical and independently distributed as normal variables with mean 0 and standard deviation 1, the joint density function of the response vector $\boldsymbol{y}$ is

$$f(\boldsymbol{y}|\boldsymbol{\beta}, \sigma) = [2\pi\sigma^2]^{-\frac{n}{2}} e^{-\frac{1}{2\sigma^2}(\boldsymbol{y}-\boldsymbol{\beta}X)'(\boldsymbol{y}-\boldsymbol{\beta}X)}. \tag{13}$$

In conventional notations, the error vector $\boldsymbol{e} \sim N(0, I_n)$, and the response vector $\boldsymbol{y} \sim N(\boldsymbol{\beta}X, \sigma^2 I_n)$, where I_n is a an identity matrix of order n. The likelihood method would lead to the maximum likelihood (ML) estimator of the regression vector $\tilde{\boldsymbol{\beta}} = (X'X)^{-1}X'\boldsymbol{y}$ and that of the covariance matrix is $\tilde{\sigma}_n^2 I_n$ in which $\tilde{\sigma}_n^2 = \frac{1}{n}(\boldsymbol{y} - \tilde{\boldsymbol{\beta}}X)'(\boldsymbol{y} - \tilde{\boldsymbol{\beta}}X)$. Note that the ML estimator of covariance matrix is biased but the LS is unbiased. Under the normal model, without using the CLT and regardless of the sample size, $\tilde{\boldsymbol{\beta}} \sim N\left(\boldsymbol{\beta}, [X'X]^{-1}\sigma^2\right)$.

In regression analysis, often the interest centres at testing the general linear hypothesis that encompasses many hypotheses as special cases. The general linear hypothesis can be expressed as

$$H_0 : H\boldsymbol{\beta} = \boldsymbol{d} \text{ against } H_a : H\boldsymbol{\beta} \neq \boldsymbol{d}, \tag{14}$$

where H is a $r \times p$ matrix of constants with rank $r \leq p$, and $\boldsymbol{d}$ is a r vector of constants. Different choices of elements of H would lead to different hypotheses. As a special case, if H is an identity matrix and $\boldsymbol{d} = \boldsymbol{0}$ the test becomes a test of significance of the regression parameters. The alternative hypothesis H_a could be specified as one sided, if needed. The following test statistic is appropriate to test the general linear hypothesis:

$$\mathcal{L} = \frac{\text{MS Regression}}{\text{MS Error}}, \tag{15}$$

where the mean sum-of-squares regression,

$$\text{MS Regression} = \frac{\text{SS Regression}}{\text{df}_R} = [H\tilde{\boldsymbol{\beta}} - \boldsymbol{d}]'[H(X'X)^{-1}H']^{-1}[H\tilde{\boldsymbol{\beta}} - \boldsymbol{d}]/r \tag{16}$$

and the mean sum-of-squares error,

$$\text{MS Error} = \frac{\text{SS Error}}{\text{df}_E} = [\boldsymbol{y} - X\tilde{\boldsymbol{\beta}}]'[\boldsymbol{y} - X\tilde{\boldsymbol{\beta}}]/(n-p). \tag{17}$$

Note that

$$s^2 = [\boldsymbol{y} - X\tilde{\boldsymbol{\beta}}]'[\boldsymbol{y} - X\tilde{\boldsymbol{\beta}}]/(n-p) = \sum_{j=1}^{n}(y_j - \tilde{y}_j)^2/(n-p) \tag{18}$$

is an unbiased estimator of σ^2. The above test statistic follows an F distribution with r and $(n-p)$ degrees of freedom (df). Under the null hypothesis $\mathcal{L}$ follows a central F distribution and under the alternative hypothesis it follows a non-central F distribution with non-centrality parameter $\Delta = [H\boldsymbol{\beta}_0 - \boldsymbol{d}]'[H(X'X)^{-1}H]^{-1}[H\boldsymbol{\beta}_0 - \boldsymbol{d}]/[2\sigma^2]$.

Improved Estimators

Improved estimators have been suggested for the regression parameters when uncertain non-sample prior information (NSPI) on the value of the parameters in available. Ever since the publication of seminal papers of [23, 24] there has been growing interest in the search for "improved" estimator of the mean vector, $\boldsymbol{\mu}$ for the multivariate normal population. Earlier, [25] and later [26] developed the preliminary test estimator that uses uncertain prior information, in addition to the sample information. In a series of papers, Saleh and Sen [27, 28] explored the preliminary test approach to James–Stein type estimation. Many authors have contributed to this area, notably [29–31]. All the earlier developments are based on the normal model. Investigations on improved estimation for Student-t model have been rather a recent development. References [32–37] studied the preliminary test and Stein-type estimation for linear models with multivariate Student-t errors. However, [38] investigated the problem from the sampling theory approach. Reference [39] deals with the improved estimation of the parallelism problem.

The value of the parameters under the null hypothesis is often called the *restricted estimator* (RE). For the multiple regression model if we suspect $H_0 : H\boldsymbol{\beta} = \boldsymbol{d}$ but not sure, we test the earlier general liner hypothesis to remove the uncertainty. Then, the RE becomes $\hat{\boldsymbol{\beta}} = \boldsymbol{d}$. The method uses both the sample information as well as the NSPI to define *preliminary test* (PT) estimator. It combines the both in the following way:

$$\hat{\boldsymbol{\beta}}^{PT} = \tilde{\boldsymbol{\beta}} - [\tilde{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}]I\left(\mathcal{L} < F_{r,m}(\alpha)\right), \tag{19}$$

where $I(\cdot)$ is an indicator function assuming value 1 or 0 depending on if the inequality in the argument holds or not, and $F_{r,m}(\alpha)$ is the upper $(1-\alpha)$th quantile

of an F distribution with (r, m) df in which $m = (n - p)$, the error d.f. The PT estimator combines the sample information, NSPI and the test statistic appropriate in removing the uncertainty in the NSPI.

The PT estimator is a convex combination of the UE and RE. In fact, it is an extreme choice between the two, and does not allow any smooth transition. It also depends on a preselected level of significance (α). The Stein-type (see [24, 31]) shrinkage estimator (SE) also uses the sample information and the test statistic. For the regression parameter vector of the multiple regression model, the SE is defined as

$$\hat{\boldsymbol{\beta}}^S = \tilde{\boldsymbol{\beta}} - d[\tilde{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}]\mathcal{L}^{-1}, \tag{20}$$

where $d = \frac{m(r-2)}{(m+2)r}$ is the shrinkage constant that minimises the quadratic risk of $\hat{\boldsymbol{\beta}}^S$. The SE is obtained by replacing $I\ (\mathcal{L} < F_{r,m}(\alpha))$ in (10) by $d\,\mathcal{L}^{-1}$ to make the estimator independent of α. Under the quadratic loss function the SE uniformly dominates the UE if $r > 2$.

The SE can be negative, even if the variable under investigation is non-negative, and it is unstable if the value of $\mathcal{L}$ in near zero. Of course, as $\mathcal{L} \to \infty$ the SE approached to the UE. To overcome the these problems, the positive-rule shrinkage (PRS) estimator is defined as

$$\begin{aligned}\hat{\boldsymbol{\beta}}^{S+} &= \hat{\boldsymbol{\beta}} + [\tilde{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}]\{1 - d\,\mathcal{L}^{-1}\}I(\mathcal{L} > d) \\ &= \hat{\boldsymbol{\beta}}^S + [\tilde{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}]\{1 - d\,\mathcal{L}^{-1}\}I(\mathcal{L} < d)\end{aligned} \tag{21}$$

which has the same format as the PT estimator, but based on $\hat{\boldsymbol{\beta}}^S$ and $\hat{\boldsymbol{\beta}}$ with the critical value d. It is well known that the PRS estimator is admissible under quadratic risk over all other estimators when $r > 2$. For details see [22].

3.2 Estimation for Multivariate Non-Normal Models

Reference [40] discarded the normal distribution as a sole model for the distribution of errors. Reference [2] showed that the results based on the Student-t models for linear models are applicable to those of normal models, but not the vice-versa. Reference [41] critically analyzed the problems of the normal distribution and recommended the Student-t distribution as a better alternative for many problems. The failure of the normal distribution to model the fat-tailed distributions has led to the use of the Student-t model in such a situation. In addition to being robust, the Student-t distribution is a 'more typical' member of the elliptical class of distributions. Moreover, the normal distribution is a special (limiting) case of the Student-t distribution. It also covers the Cauchy distribution on the other extreme. Extensive

work on this area of non-normal models has been done in recent years. A brief summary of such literature has been given by [42], and other notable references include [4, 33, 43–48].

Let $\boldsymbol{X}$ be a p-dimensional random vector having a normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix, $\tau^2 I_p$. Assume τ follows an inverted gamma distribution with the shape parameter ν and density function

$$p(\tau; \nu, \sigma^2) = \left(\frac{2}{\Gamma(\nu/2)}\right)\left(\frac{\nu\sigma^2}{2}\right)^{\nu/2} \tau^{-(\nu+1)} e^{-\frac{\nu\sigma^2}{2\tau^2}}. \tag{22}$$

Then, the distribution of $\boldsymbol{X}$, conditional on τ, is denoted by $\boldsymbol{X}|\tau \sim N_p(\boldsymbol{\mu}, \tau^2 I_p)$, and that of τ by $\tau \sim IG(\nu, \sigma)$. In the literature, it is well known that the mixture distribution of $\boldsymbol{X}$ and τ is a multivariate Student-t, and is obtained by completing the following integral:

$$p(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma, \nu) = \int_{\tau=0}^{\infty} N_p(\boldsymbol{\mu}, \tau^2 I_p) IG(\nu, \sigma) \mathrm{d}\tau, \tag{23}$$

where $\Sigma = \sigma^2 I_p$ and ν is the number of degrees of freedom. The integration yields the unconditional density of $\boldsymbol{X}$ as

$$p(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma, \nu) = k_1(\nu, p)|\Sigma|^{-1/2}\left[\nu + (\boldsymbol{x} - \boldsymbol{\mu})'\Sigma^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right]^{-\frac{\nu+p}{2}}, \tag{24}$$

where $k_1(\nu, p) = \left\{\Gamma\left(\frac{\nu+p}{2}\right)\nu^{\nu/2}\right\}\left\{\pi^{p/2}\Gamma\left(\frac{\nu}{2}\right)\right\}^{-1}$ is the normalizing constant. The above density is the p-dimensional multivariate Student-t density with an arbitrary unknown shape parameter ν. In notation, we write $\boldsymbol{X} \sim t_p(\boldsymbol{\mu}, \Sigma, \nu)$. So, $\mathrm{E}[\boldsymbol{X}] = \boldsymbol{\mu}$ and $\mathrm{Cov}[\boldsymbol{X}] = \frac{\nu}{\nu-2}\Sigma$. A method of moment estimator for ν is given by [35]. But here we are interested in the estimation of the mean vector.

Now, consider a random sample of size n from the earlier multivariate Student-t population. The likelihood function of the parameters for the given sample is

$$\begin{aligned} &L(\boldsymbol{\mu}, \Sigma, \nu; \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \\ &\quad = k_n(\nu, p)|\Sigma|^{-\frac{n}{2}}\left[\nu + \sum_{j=1}^{n}(\boldsymbol{x}_j - \boldsymbol{\mu})'\Sigma^{-1}(\boldsymbol{x}_j - \boldsymbol{\mu})\right]^{-\frac{\nu+np}{2}}, \end{aligned} \tag{25}$$

where $k_n(\nu, p) = \left\{\Gamma\left(\frac{\nu+np}{2}\right)\nu^{\nu/2}\right\}\left\{\pi^{np/2}\Gamma\left(\frac{\nu}{2}\right)\right\}^{-1}$. Refer to [36] for details on sampling from multivariate Student-t population by using the mixture of multivariate normal and inverted gamma distributions.

Here, we wish to estimate the mean vector $\boldsymbol{\mu}$ based on the random sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ when an uncertain *non-sample prior information* on $\boldsymbol{\mu}$ is available which can be presented by the null hypothesis, $H_0 : \boldsymbol{\mu} = \boldsymbol{\mu}_0$. The estimator of $\boldsymbol{\mu}$ and σ^2 are

$$\tilde{\boldsymbol{\mu}} = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{X}_j = \overline{\boldsymbol{X}} \text{ and} \tag{26}$$

$$\tilde{\sigma}^2 = \frac{1}{np}\sum_{j=1}^{n} (\boldsymbol{x}_j - \tilde{\boldsymbol{\mu}})'(\boldsymbol{x}_j - \tilde{\boldsymbol{\mu}}) \text{ respectively.} \tag{27}$$

The above are termed as *unrestricted estimators* (UE) of $\boldsymbol{\mu}$ and σ^2 respectively. The *restricted estimator* (RE) of the mean vector becomes $\hat{\boldsymbol{\mu}} = \boldsymbol{\mu}_0$.

To test $H_0 : \boldsymbol{\mu} = \boldsymbol{\mu}_0$ against $H_a : \boldsymbol{\mu} \neq \boldsymbol{\mu}_0$, the likelihood ratio statistic is given by

$$\lambda = \left[\hat{\sigma}^2/\tilde{\sigma}^2\right]^{-np/2} = \left[\sum_{j=1}^{n} (\boldsymbol{x}_j - \boldsymbol{\mu}_0)'(\boldsymbol{x}_j - \boldsymbol{\mu}_0)/s^2\right]^{-np/2}, \tag{28}$$

where $s^2 = [(n-1)p]^{-1}\sum_{j=1}^{n}(\boldsymbol{x}_j - \bar{\boldsymbol{x}})'(\boldsymbol{x}_j - \bar{\boldsymbol{x}})$. Then, it can be easily shown that under H_0 the statistic

$$T^2 = \lambda^{-2/np} = \sum_{j=1}^{n} (\boldsymbol{x}_j - \boldsymbol{\mu}_0)'(x_j - \mu_0)/s^2 \tag{29}$$

follows a scaled F-distribution with p and $m = (n-p)$ degrees of freedom (cf. [37]), and can be used to test the H_0. As discussed by [32], the F-statistic stated earlier is robust, and it is valid for all the members of the elliptical class of distributions, not just for the normal or Student-t distributions. Thus, to test the $H_0 : \boldsymbol{\mu} = \boldsymbol{\mu}_0$, we perform the F-test based on the following monoton function of the T^2-statistic:

$$F = \frac{p}{m}T^2 = \frac{\chi_n^2(\psi)}{\chi_m^2}, \tag{30}$$

where $\psi = (\boldsymbol{\mu} - \boldsymbol{\mu}_0)'(\boldsymbol{\mu} - \boldsymbol{\mu}_0)/2\sigma^2$ is the non-centrality parameter when the H_0 is not true.

Following [25], we define the *preliminary test estimator* (PTE) of $\boldsymbol{\mu}$ as follows:

$$\hat{\boldsymbol{\mu}}^P = \hat{\boldsymbol{\mu}}\, I(T^2 \leq T_\alpha^2) + \tilde{\boldsymbol{\mu}}\, I(T^2 > T_\alpha^2), \tag{31}$$

where $\hat{\boldsymbol{\mu}} = \boldsymbol{\mu}_0$ is the *restricted estimator* (RE) of $\boldsymbol{\mu}$ under the H_0; T_α^2 is a value of the T^2-statistic such that $P_r\{T^2 \leq T_\alpha^2\} = \alpha$ when the H_0 is true, for $0 \leq \alpha \leq 1$; and $I(A)$ is an indicator function of the set A.

The SE of $\boldsymbol{\mu}$ is defined as

$$\hat{\boldsymbol{\mu}}^{S} = \hat{\boldsymbol{\mu}} + (1 - k^* T^{-2})(\tilde{\boldsymbol{\mu}} - \hat{\mu}), \tag{32}$$

where k^* is the shrinkage constant. An optimal value of k^* that minimizes the value of the quadratic risk function is found to be $k = \dfrac{p-2}{m+2}$. The *positive-rule shrinkage estimator* (PRSE) is defined as follows:

$$\hat{\boldsymbol{\mu}}^{S+} = \hat{\boldsymbol{\mu}} + (1 - kT^{-2})(\tilde{\boldsymbol{\mu}} - \hat{\boldsymbol{\mu}})I(T^2 > k). \tag{33}$$

The SE uniformly dominates the UE, and the PRSE uniformly dominates the SE when $p > 2$. For details on the improved estimation for Student-t models see [45, 46].

3.3 Prediction Distribution for Elliptic Models

The multivariate multiple regression model for n_f unrealized future responses from the *future experiment* can be expressed as

$$\boldsymbol{Y}_{\mathrm{f}} = \boldsymbol{\beta} \boldsymbol{Z}_f + \Gamma \boldsymbol{E}_{\mathrm{f}}, \tag{34}$$

where $\boldsymbol{Z}_f$ is an $m \times n_f$ dimensional matrix of the values of regressors that generate the $p \times n_f$ dimensional future response matrix $\boldsymbol{Y}_{\mathrm{f}}$, and $\boldsymbol{E}_{\mathrm{f}}$ is the matrix of future error components. The joint pdf of the combined errors, $(\boldsymbol{E}, \boldsymbol{E}_f)$, is expressed as

$$f(\boldsymbol{E}, \boldsymbol{E}_f) \propto g\left\{tr[\boldsymbol{E}\boldsymbol{E}' + \boldsymbol{E}_f \boldsymbol{E}'_f]\right\}, \tag{35}$$

where $g(\cdot)$ is such that $f(\cdot)$ is a proper density function. The joint pdf of the error regression matrix $\boldsymbol{B}_E = \boldsymbol{E}\boldsymbol{Z}'(\boldsymbol{Z}\boldsymbol{Z}')^{-1}$, the associated sum of squares of error matrix $\boldsymbol{S}_E = [\boldsymbol{E} - \boldsymbol{B}_E \boldsymbol{Z}][\boldsymbol{E} - \boldsymbol{B}_E \boldsymbol{Z}]'$, and future error matrix $\boldsymbol{E}_f$ becomes

$$p\Big(\boldsymbol{B}_E, \boldsymbol{S}_E, \boldsymbol{E}_f\Big) \propto |\boldsymbol{S}_E|^{\frac{n-p-m}{2}} g\left\{tr\left[h_1(\boldsymbol{E}, \boldsymbol{Z}) + \boldsymbol{S}_E + \boldsymbol{E}_f \boldsymbol{E}'_f\right]\right\}, \tag{36}$$

where $h_1(\boldsymbol{E}, \boldsymbol{Z}) = \boldsymbol{B}_E \boldsymbol{Z}\boldsymbol{Z}'\boldsymbol{B}'_E$. Using the transformations

$$\boldsymbol{U} = \boldsymbol{B}_E, \ \boldsymbol{V} = \boldsymbol{S}_E, \text{ and } \boldsymbol{W} = \boldsymbol{S}_E^{-\frac{1}{2}}[\boldsymbol{E}_f - \boldsymbol{B}_E \boldsymbol{Z}_f] \tag{37}$$

it can be easily shown that

$$\boldsymbol{W} = \boldsymbol{S}_E^{-\frac{1}{2}}[\boldsymbol{E}_f - \boldsymbol{B}_E \boldsymbol{Z}_f] = \boldsymbol{S}_Y^{-\frac{1}{2}}[\boldsymbol{Y}_f - \boldsymbol{B}_Y \boldsymbol{Z}_f]. \tag{38}$$

Thus, the prediction distribution of $\boldsymbol{Y}_f$ can be derived from the density of $\boldsymbol{W}$. Writing the joint density of $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$, and completing the matrix integrations the pdf of $\boldsymbol{W}$ becomes

$$p(\boldsymbol{W}) \propto \left| I_p + \boldsymbol{W}(I_{n_f} - \boldsymbol{Z}'_f A^{-1} \boldsymbol{Z}_f)\boldsymbol{W}' \right|^{-\frac{n+n_f-m}{2}}, \tag{39}$$

where $A = [\boldsymbol{Z}\boldsymbol{Z}' + \boldsymbol{Z}_f \boldsymbol{Z}'_f]$. Then, the conditional density of the future response $\boldsymbol{Y}_f$ is found to be

$$p(\boldsymbol{Y}_f|\boldsymbol{Y}) = \phi(\boldsymbol{Y}, \boldsymbol{Z}_f) \left| \boldsymbol{S}_Y + \{\boldsymbol{Y}_f - \boldsymbol{B}_Y \boldsymbol{Z}_f\} H \{\boldsymbol{Y}_f - \boldsymbol{B}_Y \boldsymbol{Z}_f\}' \right|^{-\frac{n+n_f-m}{2}}, \tag{40}$$

where $\phi(\boldsymbol{Y}, \boldsymbol{Z}_f)$ is the normalizing constant and $H = \left(I_{n_f} - \boldsymbol{Z}'_f A^{-1} \boldsymbol{Z}_f\right)$. The normalizing constant for the prediction density of $\boldsymbol{Y}_f$ is

$$\phi(\boldsymbol{Y}, \boldsymbol{Z}_f) = \frac{\Gamma_p(\frac{n-m}{2})|H|^{-\frac{p}{2}}}{[\pi]^{\frac{pn_f}{2}} \Gamma_p(\frac{n+n_f-m}{2})|\boldsymbol{S}_Y|^{\frac{n-m}{2}}}, \tag{41}$$

where $\Gamma_p(a)$ is the generalized gamma function defined as

$$\Gamma_p(a) = [\pi]^{\frac{a(a-1)}{4}} \prod_{i=1}^{a} \Gamma\left(a - \frac{1}{2}[i-1]\right). \tag{42}$$

The density of future responses, conditional on realized responses, is a $p \times n_f$-dimensional matrix T density with $(n - p - m + 1)$ degrees of freedom, location matrix $\boldsymbol{B}_Y \boldsymbol{Z}_f$ and scale matrices H and $\boldsymbol{S}_Y$. So, the prediction distribution, in conventional notation (cf. [49], p.117), can be written as $[\boldsymbol{Y}_f|\boldsymbol{Y}] \sim T_{pn_f}\left(\boldsymbol{B}_Y \boldsymbol{Z}_f, H, \boldsymbol{S}_Y, n - p - m + 1\right)$. See [21] for more on prediction distribution of multivariate model with matrix-T errors.

As a special case, if the interest is to make inference on a single future response of a multivariate multiple regression model, set $n_f = 1$, and hence $\boldsymbol{y}_f = \boldsymbol{\beta} z_f + \Gamma \boldsymbol{e}_f$, where $\boldsymbol{y}_f$ and $\boldsymbol{e}_f$ are $p \times 1$ column vectors, and z_f is $m \times 1$ column vector. Then, the prediction distribution of a p-variate single response is

$$[\boldsymbol{y}_f|\boldsymbol{Y}] \sim t_p\left(\boldsymbol{B}_{\boldsymbol{Y}} z_f, H_1 \times \boldsymbol{S}_{\boldsymbol{Y}}, n - p - m + 1\right), \tag{43}$$

where $H_1 = [1 - z'_f A^{-1} z_f]$.

In another special case, when $p = 1$, the multivariate multiple regression model becomes the multiple regression model, $\boldsymbol{y} = \boldsymbol{\beta}_1 \boldsymbol{Z} + \gamma \boldsymbol{e}$, where $\boldsymbol{y}$ and $\boldsymbol{e}$ are a n-dimensional row vectors, γ is a positive scale factor, and $\boldsymbol{\beta}_1$ is an m dimensional row vector. Then, the associated model for the n_f future responses becomes

$\boldsymbol{y}_f = \boldsymbol{\beta}_1 \boldsymbol{Z}_f + \gamma \boldsymbol{e}_f$. So, the prediction distribution is obtained as a multivariate Student-t distribution, that is,

$$[\boldsymbol{y}_f | \boldsymbol{y}] \sim t_{n_f} \left(\boldsymbol{b}_{\boldsymbol{y}} \boldsymbol{Z}_f, H \times s^2_{\boldsymbol{y}}, n - m \right), \tag{44}$$

where $s^2_{\boldsymbol{y}} = [\boldsymbol{y} - \boldsymbol{b}_{\boldsymbol{y}} \boldsymbol{Z}]'[\boldsymbol{y} - \boldsymbol{b}_{\boldsymbol{y}} \boldsymbol{Z}]$ in which $\boldsymbol{b}_{\boldsymbol{y}} = \boldsymbol{y}\boldsymbol{Z}'[\boldsymbol{Z}\boldsymbol{Z}']^{-1}$.

3.4 Tolerance Region for Elliptic Models

The prediction distribution can be used to construct β-expectation tolerance region for a set of future responses from the model. In the literature, a tolerance region $R(Y)$ is defined on a probability space $(\mathcal{X}, \mathcal{A}, \mathrm{P}_\theta)$ where $\mathcal{X}$ is the sample space of the responses in the random sample $(\boldsymbol{Y}_1, \boldsymbol{Y}_2, \cdots, \boldsymbol{Y}_n)$; $\mathcal{A}$ is a σ-field defined on the sample space; and P· is the probability measure such that $\boldsymbol{\theta} = [\boldsymbol{\beta} X, \Sigma]$ is an element of the joint parameter space Ω. Thus a tolerance region $R(Y)$ is a statistic defined on the sample space $\mathcal{X}$ and takes values in the σ-field $\mathcal{A}$. The probability content of the region $R(Y)$ is called the coverage of the tolerance region and is denoted by $C(R) = P^{\boldsymbol{\theta}}_{\boldsymbol{Y}}[R(\boldsymbol{Y})]$. Note that $C(R)$ being a function of $R(\boldsymbol{Y})$, a random variable, is itself a random variable whose probability measure is induced by the measure P_θ.

Of different kinds of tolerance regions available in the literature, here we consider a particular kind of tolerance region that has an expected probability of $0 < \beta < 1$. A tolerance region $R(\boldsymbol{Y})$ is called a β-expectation tolerance region if the expectation of its coverage probability is equal to a preassigned value β. Thus for a given set of observed $\boldsymbol{Y}$, a β-expectation tolerance region $R(\boldsymbol{Y})$ must satisfy $E[C(R)|\boldsymbol{Y}] = \beta$. If $p(\boldsymbol{Y}_f \mid \boldsymbol{Y})$ denote the prediction distribution of the set of future response $\boldsymbol{Y}_f$ for the given set of observed responses $\boldsymbol{Y}$ then we can write,

$$\int_R p(\boldsymbol{Y}_f \mid \boldsymbol{Y}) \mathrm{d}\boldsymbol{Y}_f = \int_R \int_\Omega p(\boldsymbol{Y}_f, \boldsymbol{\theta} \mid \boldsymbol{Y}) \mathrm{d}\boldsymbol{\theta}\, \mathrm{d}\boldsymbol{Y}_f \tag{45}$$

where $p(\boldsymbol{Y}_f, \boldsymbol{\theta} \mid \boldsymbol{Y})$ is the joint density function of $\boldsymbol{Y}_f$ and Θ for any given $\boldsymbol{Y}$. Thus any region $R(\boldsymbol{Y})$ that satisfies $\int_R p(\boldsymbol{Y}_f \mid \boldsymbol{Y}) \mathrm{d}\boldsymbol{Y}_f = \beta$ is called a β-expectation tolerance region. So, prediction distribution of future responses are used to construct β-expectation tolerance region. There may could be infinitely many regions that satisfy the earlier condition. So, an optimal tolerance region is the one that has the minimal enclosure among all such regions. Reference [50] proved that tolerance regions based on the prediction distributions are optimal.

To obtain the tolerance region for the multivariate multiple regression model, we need to derive the sampling distribution of an appropriate statistic involved in the prediction distribution of the future responses. Here, we use the following result (see [51], p. 115–116) to construct an optimal β-expectation tolerance region for the multivariate elliptic regression model.

Theorem. *If the distribution of a random matrix $\boldsymbol{X}\,(p\times n')$ is given by the probability element*

$$h(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X} = \left\{\frac{|H|^{\frac{p}{2}}\,\Gamma_p\left(\frac{n+n'}{2}\right)}{\pi^{\frac{n'p}{2}}\,\Gamma_p\left(\frac{n}{2}\right)|S|^{\frac{n}{2}}}\right\} \times \left|S+\{\boldsymbol{X}-M\boldsymbol{V}^*\}\,H\,\{\boldsymbol{X}-M\boldsymbol{V}^*\}'\right|^{-\frac{n+n'}{2}}\,\mathrm{d}\boldsymbol{X}, \tag{46}$$

where S and H are symmetric non-singular matrices, then the distribution of

$$U = (I+U_1)^{-1}U_1, \tag{47}$$

where $U_1 = ZZ'$ with $Z = T\{\boldsymbol{X}-M\boldsymbol{V}^*\}K$ in which T is such that $TT' = S^{-1}_{(X)}$ and K is such that $KK' = H$, is given by the probability element

$$f(U)\,\mathrm{d}U = B_p^{-1}\left(\frac{n'}{2},\frac{n}{2}\right)|U|^{\frac{n'-p-1}{2}}\,|I-U|^{\frac{n-p-1}{2}}\,\mathrm{d}U. \tag{48}$$

That is, U in (47) follows a generalized beta distribution with n' and n degrees of freedom.

Comparing (46) with the prediction density in (40), we have that $\boldsymbol{W} = (I+\boldsymbol{W}_1)^{-1}\boldsymbol{W}_1$ has a generalized beta distribution with n_f and $n-m$ degrees of freedom, where $\boldsymbol{W}_1 = ZZ'$ with $Z = T\{\boldsymbol{Y}^f - \boldsymbol{B}_{\boldsymbol{Y}}Z_f\}K$, T is such that $TT' = \boldsymbol{S}_Y^{-1}$, and K is such that $KK' = H$. It may be noted here that both $S_{\boldsymbol{Y}} = \{\boldsymbol{Y} - \boldsymbol{B}_{\boldsymbol{Y}})Z_f\}\{\boldsymbol{Y}-\boldsymbol{B}_{\boldsymbol{Y}}Z_f\}'$ and $H = [I_{n_f} - Z_f'A^{-1}Z_f]$ are symmetric non-singular matrices. [46] provides an extension of the earlier generalized beta distribution.

From the definition of a β-expectation tolerance region, $R^*(\boldsymbol{Y}) = \{\boldsymbol{W} : \boldsymbol{W} < \boldsymbol{W}^*\}$ is a β-expectation tolerance region for the central $100\beta\%$ of the multivariate elliptically contoured distribution being sampled, if $\boldsymbol{W}^*$ is the β^{th} quantile of the generalized beta distribution with n_f and $n-m$ degrees of freedom. That is, $R^*(\boldsymbol{Y})$ is a β-expectation tolerance region for the model under study if $\boldsymbol{W}^*$ is such that

$$B_p^{-1}\left(\frac{n_f}{2},\frac{n-m}{2}\right)\int_{W=0}^{W^*}|\boldsymbol{W}|^{\frac{n_f-p-1}{2}}\left|I_{n_f}-\boldsymbol{W}\right|^{\frac{n-p-m-1}{2}}\,\mathrm{d}\boldsymbol{W} = \beta. \tag{49}$$

So, the tolerance region for the future responses depends on the regression parameter of the prediction distribution $\boldsymbol{B}_{\boldsymbol{Y}}Z_f$ and the scaled scale matrix $\boldsymbol{S}_{\boldsymbol{Y}}$. So, it depends on the sample data only through the regression matrix, $\boldsymbol{B}_{\boldsymbol{Y}}Z_f$ and the scale matrix $\boldsymbol{S}_{\boldsymbol{Y}}^{\frac{1}{2}}$.

In the special case, when the interest is the tolerance region for a single response of the multivariate multiple regression model, we set $n_f = 1$, so that the model reduces to $\boldsymbol{y}_f = \boldsymbol{\beta}z_f + \Gamma\boldsymbol{e}_f$, where $\boldsymbol{y}_f$ and $\boldsymbol{e}_f$ are $p\times 1$ column vectors, and z_f

is an $m \times 1$ column vector. Since $[\boldsymbol{y}_f|\boldsymbol{Y}] \sim t_p\left(\boldsymbol{B}_{\boldsymbol{Y}} z_f, H_1 \times \boldsymbol{S}_{\boldsymbol{Y}}, n-p-m+1\right)$, to define the β-expectation tolerance region for $\boldsymbol{y}_f$, we can use the distribution of the quadratic form

$$\left(\boldsymbol{y}_f - \boldsymbol{B}_{\boldsymbol{Y}} z_f\right) H_1 S_{\boldsymbol{Y}}^{-1} \left(\boldsymbol{y}_f - \boldsymbol{B}_{\boldsymbol{Y}} z_f\right)'. \tag{50}$$

It can be easily shown that

$$\{(n-p+1)/p\}\left(\boldsymbol{y}_f - \boldsymbol{B}_{\boldsymbol{Y}} z_f\right) H_1 S_{\boldsymbol{Y}}^{-1} \left(\boldsymbol{y}_f - \boldsymbol{B}_{\boldsymbol{Y}} z_f\right)' \sim F_{p,\, n-p+1}. \tag{51}$$

Then, a β-expectation tolerance region that will enclose $100\beta\%$ of the future responses from the earlier special model is given by the ellipsoidal region:

$$R(\boldsymbol{y}) = \Bigg\{ \boldsymbol{y} : [(n-p+1)/p] \times \Big[\left(\boldsymbol{y}_f - \boldsymbol{B}_{\boldsymbol{Y}} \boldsymbol{Z}_f\right) \left(I_{n_f} - \boldsymbol{Z}'_f A^{-1} \boldsymbol{Z}_f\right) S_{\boldsymbol{Y}}^{-1} \left(\boldsymbol{y}_f - \boldsymbol{B}_{\boldsymbol{Y}} \boldsymbol{Z}_f\right)'\Big] \leq F_{n_f, n-p+1, \beta} \Bigg\}, \tag{52}$$

where $F_{n_f,\, n-p+1,\beta}$ is the $\beta \times 100\%$ point of a central F distribution with n_f and $n-p+1$ degrees of freedom such that $P(F_{n_f,\, n-p+1} < F_{n_f,\, n-p+1,\beta}) = \beta$.

4 Bayesian Inference

In the recent years, Bayesian methods are gaining increasing popularity, arguably, due the advent of high-power computers. The evaluation of marginal posterior density function involving complicated integration is now performed by routine computer program. Many statistical packages are available to perform the computations using various popular algorithms.

4.1 Bayesian Philosophy

Central to the Bayesian philosophy is the Bayes' Theorem. Unlike the Classical methods, the Bayesian methods treat the parameters of statistical models as random variables that follow some probability distribution. Such prior distributions are combined with the likelihood function of the parameters based on available sample data. The posterior distribution of the parameters is obtained from the product of the prior distribution and the likelihood function.

Consider a random sample $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)'$ from the multiple regression model with regression vector $\boldsymbol{\beta}$ and scale parameter σ. Let $\boldsymbol{\theta} = [\boldsymbol{\beta},\ \sigma^{-1}]$. Then, the Bayes' posterior density of θ is given by

$$p(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{p(\boldsymbol{\theta})\, p(\boldsymbol{y}|\boldsymbol{\theta})}{p(\boldsymbol{y})}, \tag{53}$$

where $p(\boldsymbol{y}) = \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta})p(\boldsymbol{y}|\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}$ in which $p(\boldsymbol{y}|\boldsymbol{\theta})$ is the likelihood function, and $P(\boldsymbol{\theta})$ is the prior distribution of $\boldsymbol{\theta}$. All inferences on $\boldsymbol{\theta}$ is then based on the posterior distribution of the parameters. The process essentially updates the prior belief via the likelihood function. As the sample size grows larger the impact of the prior belief on the posterior distribution reduces. However, for small samples the prior information has a greater role in the determination of the posterior distribution, and hence on the estimator of $\boldsymbol{\theta}$. For recent references on multivariate Bayesian methods see [52–54] to name a few.

4.2 Estimation

Consider the multiple regression model

$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{54}$$

where $\boldsymbol{y}$ is an n-dimensional vector of responses, and X is the design matrix of order $n \times p$ in which $k = (p-1)$ is the number of regressors or predictors, $\boldsymbol{e}$ is the scaled error vector and $\boldsymbol{\epsilon}$ is the error vector of order n. Assume that $\boldsymbol{\epsilon} \sim N(0, \sigma^2 I_n)$. Then, the inference on $\boldsymbol{\beta}$ depends on whether the variance of $\boldsymbol{\epsilon}$, that is, σ^2 is known or unknown.

When σ^2 is known the posterior distribution of $\boldsymbol{\beta}$ is given by

$$p(\boldsymbol{\beta}|\boldsymbol{y}) \propto p(\boldsymbol{\beta}) \times p(\tilde{\boldsymbol{\beta}}|\boldsymbol{\beta}, \sigma^2), \tag{55}$$

where $\tilde{\boldsymbol{\beta}}$ is the maximum likelihood estimator of $\boldsymbol{\beta}$ and $p(\boldsymbol{\beta})$ is the prior distribution. Under the non-informative prior, that is, $p(\boldsymbol{\beta}) \propto$ constant, for the regression vector, the posterior distribution is given by

$$p(\boldsymbol{\beta}|\boldsymbol{y}) = \frac{|X'X|^{1/2}}{[2\pi\sigma^2]^{k/2}} \exp\left[-\frac{1}{2\sigma^2}(\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})X'X(\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})\right]. \tag{56}$$

The Bayes' estimator of $\boldsymbol{\beta}$ is the posterior mean, that is, $\tilde{\boldsymbol{\beta}}$. All other inferences on the regression vector can be made from the fact that $\boldsymbol{\beta}|\boldsymbol{y} \sim N(\tilde{\boldsymbol{\beta}}, \sigma^2[X'X]^{-1})$. It may be noted that the marginal posterior distribution of any subset of the regression vector is a multivariate normal distribution with appropriate mean vector and covariance matrix.

If σ^2 is unknown, and information on σ^2 is available from the sample data, the posterior distribution of $\boldsymbol{\beta}$ and σ^2 is defined as

$$p(\boldsymbol{\beta}, \sigma^2|\boldsymbol{y}) \propto p(\boldsymbol{\beta}, \sigma^2) \times p(\tilde{\boldsymbol{\beta}}|\boldsymbol{\beta}, \sigma^2). \tag{57}$$

Assume that $\boldsymbol{\beta}$ and σ^2 are independent and locally uniform, so that the joint prior distribution is

$$p(\boldsymbol{\beta}, \sigma^2) = p(\boldsymbol{\beta}) \times p(\sigma^2) \propto \sigma^{-2}. \tag{58}$$

Then, the marginal posterior distribution of $\boldsymbol{\beta}$ becomes

$$p(\boldsymbol{\beta}|\boldsymbol{y}) = \frac{\Gamma\left([\nu + k]/2\right)|X'X|^{1/2}}{[\pi\nu]^{k/2}\,\Gamma(\nu/2)} \left[1 + \frac{\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})X'X(\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})}{\nu s^2}\right]^{-\frac{\nu+k}{2}}, \tag{59}$$

where $s^2 = (\boldsymbol{y} - X\hat{\boldsymbol{\beta}})'(\boldsymbol{y} - X\hat{\boldsymbol{\beta}})/\nu$ and $\nu = n - k$. So, conditional on the observed responses, the regression vector follows a multivariate Student-t distribution, that is, $\boldsymbol{\beta}|\boldsymbol{y} \sim t_p\left(\tilde{\boldsymbol{\beta}}, s^2[X'X]^{-1}, \nu\right)$. The location vector of the earlier multivariate Student-t distribution is $\tilde{\boldsymbol{\beta}}$ and the covariance matrix is $\mathrm{Cov}(\boldsymbol{\beta}|\boldsymbol{y}) = \frac{\nu}{\nu-2}s^2[X'X]^{-1}$. It may be noted that the marginal posterior distribution of any subset of the regression vector is a Student-t distribution with appropriate location vector and scale matrix.

Clearly, the posterior distribution depends on the knowledge of σ^2 as well as the prior distribution of the parameters, along with the sample data. In a particular case, when the posterior distribution is in the same family as the prior distribution, the prior distribution is called the conjugate prior. For the above model, let the errors be i.i.d. normal so that $\boldsymbol{y} \sim N_n(X\boldsymbol{\beta}, \sigma^2 I_n)$. Let the prior distribution of the regression vector $\boldsymbol{\beta}$ also be normal, that is, $\boldsymbol{\beta} \sim N_p(\boldsymbol{\xi}, \tau^2 V)$, where $\boldsymbol{\xi}$ a is p-dimensional vector, V is a $p \times p$ non-singular matrix, and τ^2 is a scalar. The hyperparameters τ^2 and $\boldsymbol{\xi}$ are either known or estimated from the sample data. Then, for a given set of responses, $\boldsymbol{y}$, the posterior distribution of $\boldsymbol{\beta}$ is normal. In the conventional notation, the posterior distribution of the regression vector becomes

$$\boldsymbol{\beta}|\boldsymbol{y} \sim N_p\left(\boldsymbol{\xi} + \left[C + \sigma^2\tau^{-2}V^{-1}\right]^{-1} X'(\boldsymbol{y} - X\boldsymbol{\xi}), \sigma^2\left[C + \sigma^2\tau^{-2}V^{-1}\right]^{-1}\right), \tag{60}$$

where $C = X'X$ is of full rank, and $V = C^{-1}$. Since both the prior and posterior distributions are of the same (normal) family, the prior distribution here is a conjugate distribution. The mean of the posterior distribution is the Bayes' estimator of the regression parameter vector. Thus

$$\hat{\boldsymbol{\beta}}^B = \mathrm{E}[\boldsymbol{\beta}|\boldsymbol{y}] = \boldsymbol{\xi} + (1 - B)(\tilde{\boldsymbol{\beta}} - \boldsymbol{\xi}), \tag{61}$$

where $B = \frac{\sigma^2}{\sigma^2+\tau^2}$, and the posterior distribution is

$$N_p\left(\boldsymbol{\xi}[1 - B](\tilde{\boldsymbol{\beta}} - \boldsymbol{\xi}), \sigma^2[1 - B]C^{-1}\right). \tag{62}$$

4.3 Prediction Distribution

Consider the following non-informative prior distribution of the regression and scale matrices of the multivariate multiple regression model defined in (5)

$$p\left(\boldsymbol{\beta}, \Sigma^{-1}\right) \propto |\boldsymbol{\Sigma}^{-1}|^{\frac{p+1}{2}}. \tag{63}$$

Assuming that the errors follow matrix variate elliptically symmetric distribution, the joint density of $\boldsymbol{Y}$ from the *performed experiment* and the future responses $\boldsymbol{Y}_f$ from the *future experiment* of the multivariate multiple regression model can be expressed as

$$f(\boldsymbol{Y}, \boldsymbol{Y}_f|\boldsymbol{\beta}, \boldsymbol{\Sigma}^{-1}) \propto |\boldsymbol{\Sigma}^{-1}|^{\frac{n+n_f}{2}} g\left\{tr\left\{\boldsymbol{\Sigma}^{-1}\left[R_1\left(\boldsymbol{Y}, \boldsymbol{Z}\right) + R_2\left(\boldsymbol{Y}_f, \boldsymbol{Z}_f\right)\right]\right\}\right\}, \tag{64}$$

where

$$\begin{aligned} R_1(\boldsymbol{Y}, \boldsymbol{Z}) &= (\boldsymbol{Y} - \boldsymbol{\beta}\boldsymbol{Z})(\boldsymbol{Y} - \boldsymbol{\beta}\boldsymbol{Z})' \text{ and} \\ R_2(\boldsymbol{Y}_f, \boldsymbol{Z}_f) &= \left(\boldsymbol{Y}_f - \boldsymbol{\beta}\boldsymbol{Z}_f\right)\left(\boldsymbol{Y}_f - \boldsymbol{\beta}\boldsymbol{Z}_f\right)'. \end{aligned} \tag{65}$$

Let $p(\boldsymbol{\beta}, \boldsymbol{\Sigma}^{-1}|\boldsymbol{Y}, \boldsymbol{Y}_f)$ be the posterior density of $\boldsymbol{\beta}$ and $\boldsymbol{\Sigma}^{-1}$. Then, the prediction distribution of the future responses matrix is obtained by solving the following integral

$$p(\boldsymbol{Y}_f|\boldsymbol{Y}) \propto \int_{\boldsymbol{\beta}} \int_{\boldsymbol{\Sigma}^{-1}} p(\boldsymbol{Y}, \boldsymbol{Y}_f|\boldsymbol{\beta}, \boldsymbol{\Sigma}) p(\boldsymbol{\beta}, \boldsymbol{\Sigma}^{-1}) \mathrm{d}\boldsymbol{\Sigma}^{-1} \mathrm{d}\boldsymbol{\beta}. \tag{66}$$

To evaluate the matrix integral, let $\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Omega}$. So, $\mathrm{d}\boldsymbol{\Sigma} = |\boldsymbol{\Omega}|^{-(p+1)} \mathrm{d}\boldsymbol{\Omega}$. Then, the prediction distribution of $\boldsymbol{Y}_f$ can be expressed as

$$p(\boldsymbol{Y}_f|\boldsymbol{Y}) \propto \int_{\boldsymbol{\beta}} \int_{\boldsymbol{\Omega}} |\boldsymbol{\Omega}|^{\frac{n_f+n-m}{2}} g\left\{tr\boldsymbol{\Omega}\left[R(\boldsymbol{\beta}, \boldsymbol{Y}_f)\right]\right\} \mathrm{d}\boldsymbol{\Omega}\, \mathrm{d}\boldsymbol{\beta}. \tag{67}$$

Now, let $\boldsymbol{D}$ be a non-singular symmetric matrix of order p such that

$$\boldsymbol{D}'\boldsymbol{D} = \boldsymbol{S}_Y + M_1(\boldsymbol{Y}_f, \boldsymbol{Z}_f) + M_2(\boldsymbol{\beta}, \hat{\boldsymbol{\beta}}), \tag{68}$$

where

$$\begin{aligned} M_1(\boldsymbol{Y}_f, \boldsymbol{Z}_f) &= (\boldsymbol{Y}_f - \hat{\boldsymbol{\beta}}\boldsymbol{Z}_f) H (\boldsymbol{Y}_f - \hat{\boldsymbol{\beta}}\boldsymbol{Z}_f)' \text{ and} \\ M_2(\boldsymbol{\beta}, \hat{\boldsymbol{\beta}}) &= (\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}) A (\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})' \end{aligned} \tag{69}$$

in which $\hat{\boldsymbol{\beta}}$ is the ordinary least-squares estimator (OLS) of $\boldsymbol{\beta}$, $H = [I_{n_f} - \boldsymbol{Z}'_f A^{-1} \boldsymbol{Z}_f]$ and $A = [\boldsymbol{Z}\boldsymbol{Z}' + \boldsymbol{Z}_f \boldsymbol{Z}'_f]$. Then, the Jacobian of the transformation $\boldsymbol{\Psi} = \boldsymbol{D}\boldsymbol{\Omega}\boldsymbol{D}'$ (cf. [55]) is $|\boldsymbol{D}'\boldsymbol{D}|^{-\frac{p+1}{2}}$. Completion of integration yields

$$p(\boldsymbol{Y}_f|\boldsymbol{Y}) = \psi(\boldsymbol{Y}, H) \left| \boldsymbol{S}_Y + (\boldsymbol{Y}_f - \hat{\boldsymbol{\beta}}\boldsymbol{Z}_f) H (\boldsymbol{Y}_f - \hat{\boldsymbol{\beta}}\boldsymbol{Z}_f)' \right|^{-\frac{n_f+n-m}{2}}, \quad (70)$$

where the normalizing constant is given by

$$\psi(\boldsymbol{Y}, H) = \frac{\Gamma_p\left(\frac{n-m}{2}\right) |H|^{-\frac{p}{2}}}{(\pi)^{\frac{pn_f}{2}} \Gamma_p\left(\frac{n+n_f-m}{2}\right) |\boldsymbol{S}_Y|^{\frac{n-m}{2}}}. \quad (71)$$

The above density is the probability density of a pn_f-dimensional matrix T distribution with location matrix $\hat{\boldsymbol{\beta}}\boldsymbol{Z}_f$, scale matrices $\boldsymbol{S}_Y$ and H and degrees of freedom $(n - p - m + 1)$. Note that the covariance matrix of the prediction distribution is given by $\text{Cov}(\boldsymbol{Y}_f|\boldsymbol{Y}) = \frac{(n-p-m+1)}{(n-p-m-1)}[\boldsymbol{S}_Y \bigotimes H]$ where $\bigotimes$ is the Kronecker product of two matrices. Thus, the above prediction density of the future responses is the same as that obtained using invariant differentials and structural relations of the model.

5 Nonparametric Methods

Often statistical data are available without any information on the distribution that may fit the data. Parametric methods can't be applied in any such data sets. Nonparametric or distribution free methods are essential for analyzing data that do not follow any known distribution. Also, if the variable is measured in the ordinal scale then the commonly used parametric methods are inappropriate to analyse any such ordinal data. However, if the data represent an independent random sample then non-parametric methods can be used to estimate the location or spread of the data as well as to perform statistical tests. Inferences based on non-parametric methods are robust as the results of any such method are not sensitive to any distributional assumption. However, statistical results from non-parametric methods, in general, are not superior to those obtained by parametric methods with respect to popular statistical properties.

5.1 Estimation

Different non-parametric methods are available in the literature. Maximum likelihood type method of estimation involves maximizing some function of the sample,

and is known as the M-Estimation. Estimators based on the rank statistics are known as the R-estimators. Linear rank statistics based estimators are called L-estimators. The commonly used estimators, such as mean or median, are often special cases of the earlier non-parametric estimators.

Here, we consider the R-estimation for the multiple regression model,

$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{72}$$

where the regression vector can be written as $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_k)'$ in which $1 \leq p = k + 1 \leq n$, and each component of $\boldsymbol{\epsilon} = (\epsilon_1, \epsilon_2, \ldots, \epsilon_n)'$ is independent and identically distributed having continuous distribution function $\{F(\epsilon_i) : i = 1, 2, \ldots, n\}$ with associated absolutely continuous density $f(\cdot)$.

For any real-valued vector $\boldsymbol{b} = (b_1, b_2, \ldots, b_p)$, let $R_i(b)$ be the rank of $Y_i - (\boldsymbol{x}_i - \bar{\boldsymbol{x}}_n)\boldsymbol{b}$ for $i = 1, 2, \ldots, n$, where $\boldsymbol{x}_i$ is the ith row of the design matrix $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)'$ and $\bar{\boldsymbol{x}}_n = n^{-1} \sum_{i=1}^{n} \boldsymbol{x}_i$. Then, consider the linear rank statistic

$$L_n(\boldsymbol{b}) = [L_{1n}(b_1), \ldots, L_{pn}(b_p)]' = \sum_{i=1}^{n} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}_i) a_n\left(R_i(\boldsymbol{b})\right), \tag{73}$$

where score function $a_n(\cdot)$ is generated by a function $\phi(u)$ that is non-constant, non-decreasing and square integrable, so that,

$$a_n(l) = \mathrm{E}[\phi(U_{l,n})] \quad \text{or } \phi\left(\frac{l}{n+1}\right), \quad l = 1, 2, \ldots, n, \tag{74}$$

in which $U_{l,n}$ is the order statistic with rank l in a random sample of size n from uniform distribution in the interval $(0, 1)$.

Define the sum of square and product matrix,

$$C_n = \sum_{i=1}^{n} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}_n)(\boldsymbol{x}_i - \bar{\boldsymbol{x}})'_n, \tag{75}$$

and assume that

1. $\lim_{n\to\infty} n^{-1} C_n = C$
2. $\max_{1\leq i\leq n} \left[(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_n)' C_n^{-1} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})'_n\right] = o(n)$
3. The "Fisher information" is finite, that is,

$$I(f) = \int_{-\infty}^{\infty} \left[f'(z)/f(z)\right]^2 f(z)\mathrm{d}z < \infty \tag{76}$$

4. $A_\phi^2 = \int_0^1 \phi^2(u)\mathrm{d}u - \left[\int_0^1 \phi(u)\mathrm{d}u\right]^2$
5. $\gamma(\psi, \phi) = \int_0^1 \psi(u)\phi(u)\mathrm{d}u$, and
6. $\int_0^1 \phi^2(u)\mathrm{d}u = I(f)$

$$\text{where } \psi(u) = -\frac{f'(F^{-1}(u))}{f(F^{-1}(u))},\ 0 < u < 1. \tag{77}$$

Now, setting $\|\boldsymbol{a}\| = \sum_{l=1}^{p} a^{(l)}$, where $\|\cdot\|$ is the p-dimensional Euclidean norm, and $\boldsymbol{a} = [a^{(1)}, \ldots, a^{(p)}]$, the unrestricted R-estimator, $\tilde{\boldsymbol{\beta}}_n$, of the regression vector is defined by any central point of the set

$$\boldsymbol{S} = \{\boldsymbol{b} : \|L_n(\boldsymbol{b})\| = \text{minimum}\}. \tag{78}$$

From [56], we have

$$\lim P\left\{\sum_{\|\boldsymbol{\omega}\|<k} \|L_n\left(\boldsymbol{\beta} + \frac{\boldsymbol{\omega}}{\sqrt{n}}\right) - L_n(\boldsymbol{\beta}) + \gamma C_\omega\| > \epsilon\right\} = 0 \tag{79}$$

for any $k > 0$ and $\epsilon > 0$, where $\boldsymbol{\omega}$ is a p-dimensional column vector. Therefore, as $n \to \infty$, the sampling distribution of the R-estimator of the regression vector, $\tilde{\boldsymbol{\beta}}$, is given by

$$\sqrt{n}\left(\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta}\right) \sim N_p\left(\boldsymbol{0}, \sigma^2 C^{-1}\right) \text{ with } \sigma^2 = A_\phi^2/\gamma^2(\psi, \phi). \tag{80}$$

5.2 Test of Hypothesis

To test the general linear hypothesis $H_0 : H\boldsymbol{\beta} = \boldsymbol{d}$ against $H_a : H\boldsymbol{\beta} \neq \boldsymbol{d}$, where H is a $r \times p$ matrix of non-zero constants, the rank-test is based on the test statistic

$$\mathcal{L}_n = A_n^{-1}\left[L_n(\hat{\boldsymbol{\beta}})\right]' C_n^{-1}\left[L_n(\hat{\boldsymbol{\beta}})\right], \tag{81}$$

where

$$\hat{\boldsymbol{\beta}} = \tilde{\boldsymbol{\beta}}_n - C_n^{-1}H(HC_n^{-1}H')(H\tilde{\boldsymbol{\beta}}_n - \boldsymbol{d}) \text{ and} \tag{82}$$

$$A_n^2 = (n-1)^{-1}\sum_{l=1}^{n}(a_n(l) - \bar{a}_n)^2 \text{ in which } \bar{a}_n = n^{-1}\sum_{l=1}^{n} a_n(l). \tag{83}$$

From the sampling distribution of the R-estimator of the regression vector, it can be shown that under the null hypothesis, as $n \to \infty$, the above test statistic follows a chi-square distribution with r degrees of freedom (d.f.). So, a level α test for the general linear hypothesis is to reject H_0 if the observed value of $\mathcal{L}_n$ is greater than or equal to the critical value $\chi^2_{r,\alpha}$, where $\chi^2_{r,\alpha}$ is the upper $(1-\alpha)$th quantile of a central chi-square distribution with r d.f.

Nonparametric M-estimation method along with the M-test can be found in [57–60, 63]. Also, see [56, 61] for robust estimation and tests.

References

1. D.A.S. Fraser, *The Structure of Inference* (Wiley, New York, 1968)
2. D.A.S. Fraser, *Inference and Linear Models* (McGraw-Hill, New York, 1979)
3. K.T. Fang, S. Kotz, K.W. Ng, *Symmetric Multivariate and Related Distributions* (Chapman and Hall, New York, 1990)
4. A.K. Gupta, T. Varga, *Elliptically Contoured Models in Statistics* (Kluwer Academic Publishers, Netherlands, 1993)
5. D. Kelker, Sankhya: Indian J. Stat. A **32**, 419–430 (1970)
6. R.J. Muirhead, *Aspects of Multivariate Statistical Theory* (Wiley, New York, 1982)
7. F.R. Hampel, *Robust Statistics: The Approach Based on Influence Functions* (Wiley, New York, 1986)
8. P.J. Huber, *Robust Statistics* (Wiley, New York, 1981)
9. P.J. Rousseeuw, A.M. Leroy, *Robust Regression and Outlier Detection* (Wiley, New York, 1987)
10. R.A. Maronna, R.D. Martin, V.J. Yohai, *Robust Statistics: Theory and Methods* (Wiley, New York, 2006)
11. S. Geisser, *Predictive Inference: An Introduction* (Chapman and Hall, London, 1993)
12. J. Aitchison, I.R. Dunsmore, *Statistical Prediction Analysis* (Cambridge University Press, Cambridge, 1975)
13. J. Aitchison, D. Sculthorpe, Biometrika **55**, 469–483 (1965)
14. M.S. Haq, S. Khan, Comm. Stat. Theor. Meth. **19**(12), 4705–4712 (1990)
15. S. Khan, Aligarh J. Stat. **15–16**, 1–17 (1996)
16. S. Khan, M.S. Haq, Sankhya B: Indian J. Stat. **56**, 95–106 (1994)
17. M.S. Haq, S. Rinco, J. Multivariate Anal. **6**, 414–421 (1976)
18. I. Guttman, *Statistical Tolerance Regions* (Griffin, London, 1970)
19. M.S. Haq, Statistische Hefte **23**, 218–228 (1982)
20. S. Khan, Comm. Stat. Theor. Meth. **33**(10), 2423–2443 (2004)
21. S. Khan, J. Multivariate Anal. **83**, 124–140 (2002)
22. A.K.Md.E. Saleh, *Theory of Preliminary Test and Stein-Type Estimation with Applications* (Wiley, New Jersey, 2006)
23. W. James, C. Stein, *Estimation with Quadratic Loss*. Proceedings of the Fourth Berkeley Symposium on Math. Statist. and Probability, vol. 1 (University of California Press, Berkeley, 1961), pp. 361–379
24. C. Stein, *Inadmissibility of the Usual Estimator for the Mean of a Multivariate Normal Distribution*. Proceedings of the Third Berkeley Symposium on Math. Statist. and Probability, vol. 1 (University of California Press, Berkeley, 1956), pp. 197–206
25. T.A. Bancroft, Ann. Math. Statist. **15**, 190–204 (1944)
26. C.-P. Han, T.A. Bancroft, J. Am. Stat. Assoc. **62**, 1333–1342 (1968)
27. A.K.Md.E. Saleh, P.K. Sen, Ann. Stat. **6** 154–168 (1978)
28. P.K. Sen, A.K.Md.E. Saleh, Ann. Stat. **13**, 272–281 (1985)
29. S.L. Sclove, C. Morris, R. Radhakrishnan, Ann. Math. Stat. **43**, 1481–1490 (1972)
30. G.G. Judge, M.E. Bock, *The Statistical Implications of Pre-test and Stein-rule Estimators in Econometrics* (North-Holland, New York, 1978)
31. C. Stein, Ann. Stat. **9**, 1135–1151 (1981)
32. T.W. Anderson, in *Nonnormal Multivariate Distributions: Inference Based on Elliptically Contoured Distributions*, ed. by C.R. Rao. Multivariate Analysis: Future Directions (North-Holland, Amsterdam, 1993), pp. 1–24
33. D. Celler, D. Fourdriunier, W.E. Strawderman, J. Multivariate Anal. **53**, 194–209 (1995)
34. J.A. Giles, J. Econometrics **50**, 377–398 (1991)
35. R.S. Singh, Econ. Lett. **27**, 47–53 (1988)
36. S. Khan, A.K.Md.E. Saleh, Biometrical J. **39**, 131–147 (1997)
37. A. Zellner, J. Am. Stat. Assoc. **60**, 601–616 (1976)
38. S. Khan, A.K.Md.E. Saleh, Comm. Stat. Theor. Meth. **27**, 193–210 (1998)

39. S. Khan, Comm. Stat. Theor. Meth. **37**, 247–260 (2008)
40. R.A. Fisher, *Statistical Methods in Scientific Inference* (Oli and Boyd, London, 1956)
41. I.R. Prucha, H.H. Kelejian, Econometrica **52**, 721–736 (1984)
42. M.A. Chmielewski, Int. Stat. Rev. **49**, 67–74 (1981)
43. K.T. Fang, T.W. Anderson, *Statistical Inference in Elliptically Contoured and Related Distributions* (Allerton Press, New York, 1990)
44. K.T. Fang, Y. Zhang, *Generalized Multivariate Analysis* (Science Press and Springer, New York, 1990)
45. S. Khan, J. Stat. Res. **39**(2), 79–94 (2005)
46. S. Khan, Comm. Stat. Theor. Meth. **29**(3), 507–527 (2000)
47. S. Khan, Far East J. Theor. Stat. **1**, 77–91 (1997)
48. S. Khan, M.S. Haq, Comm. Stat. Theor. Meth. **19**, 4705–4712 (1990)
49. G.E.P. Box, G.C. Tiao, *Bayesian Inference in Statistical Analysis* (Wiley, New York, 1992)
50. J. Bishop, Parametric tolerance regions. Unpublished Ph.D. Thesis, University of Toronto, Canada (1976)
51. S. Rinco, β-expectation tolerance regions based on the structural models. Ph.D. thesis, University of Western Ontario, London, Canada (1973)
52. J.M. Bernardo, A.F Smith, *Bayesian Theory* (Wiley, New York, 2000)
53. P. Congdon, *Bayesian Statistical Modeling*, 2nd edn. (Wiley, New York, 2006)
54. D.B. Rowe, *Multivariate Bayesian Statistics* (Chapman and Hall/CRC, London, 2003)
55. V.M. Ng, Comm. Stat. Theor. Meth. **29**(3), 477–483 (2000)
56. J. Jurečková, P.K. Sen, *Robust Statistical Procedures Asymptotics and Interrelations* (Wiley, New York, 1996)
57. J. Jurečková, P.K. Sen, J. Statist. Plan. Infer. **5**, 253–266 (1981)
58. P.K. Sen, Biometrika **69**, 245–248 (1982)
59. P.K. Sen, A.K.Md.E. Saleh, Ann. Stat. **15**, 1580–1592 (1987)
60. R.M. Yunus, S. Khan, Increasing power of the test through pre-test – a robust method. USQ Working Paper Series SC-MC-0713 *http://www.sci.usq.edu.au/research/workingpapers.php.* (2007)
61. R.R. Wilcox, *Introduction to Robust Estimation and Hypothesis Testing* (Elsevier, US, 2005)
62. J. Bernardo, R. Rueda, Int. Stat. Rev. **70**, 351–372 (2002)
63. A.K.Md.E. Saleh, P.K. Sen, *Nonparametric tests of location after a preliminary test on regression in the multivariate case*. Comm. Stat. Theor. Meth. **11**, 639–651 (1983)

The Philosophy of Science and its relation to Machine Learning

Jon Williamson

In this chapter I discuss connections between machine learning and the philosophy of science. First I consider the relationship between the two disciplines. There is a clear analogy between hypothesis choice in science and model selection in machine learning. While this analogy has been invoked to argue that the two disciplines are essentially doing the same thing and should merge, I maintain that the disciplines are distinct but related and that there is a dynamic interaction operating between the two: a series of mutually beneficial interactions that changes over time. I will introduce some particularly fruitful interactions, in particular the consequences of automated scientific discovery for the debate on inductivism versus falsificationism in the philosophy of science, and the importance of philosophical work on Bayesian epistemology and causality for contemporary machine learning. I will close by suggesting the locus of a possible future interaction: evidence integration.

1 Introduction

Since its genesis in the mid 1990s, data mining has been thought of as encompassing two tasks: using data to test some pre-determined hypothesis, or using data to determine the hypothesis in the first place. The full automation of both these tasks – hypothesising and then testing – leads to what is known as *automated discovery* or *machine learning*. When such methods are applied to science, we have what is called *automated scientific discovery* or *scientific machine learning*. In this chapter, we shall consider the relationship between the philosophy of science and machine learning, keeping automated scientific discovery particularly in mind.

Section 2 offers a brief introduction to the philosophy of science. In Sect. 3 it is suggested that the philosophy of science and machine learning admit mutually fruitful interactions because of an analogy between hypothesis choice in the philosophy

J. Williamson (✉)
University of Kent, Canterbury, UK
e-mail: j.williamson@kent.ac.uk

M.M. Gaber (ed.), *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*, DOI 10.1007/978-3-642-02788-8_4,

of science and model selection in machine learning. An example of the benefit of machine learning for the philosophy of science is provided by the importance of work on automated scientific discovery for the debate between inductivism and falsificationism in the philosophy of science (Sect. 4). On the other hand, the influence of philosophical work on Bayesianism and causality provides an example of the benefits of the philosophy of science for machine learning (Sect. 5). Section 6 hypothesises that evidence integration may become the locus of the further fruitful interaction between the two fields.

2 What is the Philosophy of Science?

In the quest to improve our understanding of science, three fields of enquiry stand out: history of science, sociology of science, and philosophy of science. Historians of science study the development of science, key scientists and key ideas. Sociologists of science study social constraints on scientific activity–e.g., how power struggles impact on the progress of science. Philosophers of science study the concepts of science and normative constraints on scientific activity. Questions of interest to philosophers of science include:

Demarcation: What demarcates science from non-science? One view is that empirical testability is a necessary condition for a theory to count as scientific.

Unity: To what extent is science a unified or unifiable field of enquiry? Some take physics to be fundamental and the elements of other sciences to be reducible to those of physics. Others argue that science is a hotch-potch of rather unrelated theories, or that high-level complexity is not reducible to low-level entities and their arrangement.

Realism: Are the claims of science true? To what extent are we justified in believing contemporary scientific theories? Realists hold that scientific theories aim to describe an independent reality and that science gradually gets better at describing that reality. On the other hand, instrumentalists hold that science is an instrument for making predictions and technological advances and that there is no reason to take its claims literally, or – if they are taken at face value – there are no grounds for believing them.

Explanation: What is it to give a scientific explanation of some phenomena? One view is that explaining is the act of pointing to the physical mechanism that is responsible for the phenomena. Another is that explanation is subsumption under some kind of regularity or law.

Confirmation: How does evidence confirm a scientific theory? Some hold that evidence confirms a hypothesis just when it raises the probability of the hypothesis. Others take confirmation to be a more complicated relation, or not a binary relation at all but rather to do with coherence with other beliefs.

Scientific Method: How are the goals of science achieved? What is the best way of discovering causal relationships? Can one justify induction? While many maintain that in principle one can automate science, others hold that scientific discovery is an essentially human, intuitive activity.

Concepts of the Sciences: How should one interpret the probabilities of quantum mechanics? Does natural selection operate at the level of the individual, the population or the gene? Each science has its particular conceptual questions; even the interpretation of many general concepts – such as probability and causality – remains unresolved.

3 Hypothesis Choice and Model Selection

There is a clear link between the philosophy of science on the one hand and the area of machine learning and data mining on the other. This link is based around an analogy between *hypothesis choice* in science and *model selection* in machine learning. The task of determining a scientific hypothesis on the basis of current evidence is much like the task of determining a model on the basis of given data. Moreover, the task of evaluating the resulting scientific hypothesis is much like the task of evaluating the chosen model. Finally, the task of deciding which evidence to collect next (which experiments and observations to perform) seems to be similar across science and machine learning. Apparently, then, scientific theorising and computational modeling are but two applications of a more general form of reasoning.

How is this general form of reasoning best characterised? It is sometimes called *abductive inference* or *abduction*, a notion introduced by C.S. Peirce. But this nomenclature is a mistake: the form of reasoning alluded to here is more general than abduction. Abduction is the particular logic of moving from observed phenomena to an explanation of those phenomena. Science and machine learning are interested in the broader, iterative process of moving from evidence to theory to new evidence to new theory and so on. (This broader process was sometimes called "induction" by Peirce, though "induction" is normally used instead to refer to the process of moving from the observation of a feature holding in each member of a sample to the conclusion that the feature holds of unobserved members of the population from which the sample is drawn.) Moreover, explanation is just one use of hypotheses in science and models in machine learning; hypotheses and models are also used for other forms of inference such as prediction. In fact, while explanation is often the principal target in science, machine learning tends to be more interested in prediction. When explanation is the focus, one is *theorising*; when prediction is the focus, the process is better described as *modeling*. Clearly, then, the general form of reasoning encompasses both theorising and modeling. This general form of reasoning is sometimes called *discovery*. But that is not right either: the form of reasoning under consideration here is narrower than discovery. "Discovery" applies to finding out new particular facts as well as to new generalities, but we are interested purely in generalities here. In want of a better name, we shall call this general form of reasoning *systematising*, and take it to encompass theorising and modeling.

Granting, then, that hypothesis choice in science and model selection in machine learning are two kinds of systematising, there are a variety of possible views as to the relationship between the philosophy of science and machine learning.

One might think that since the philosophy of science and machine learning are both concerned with systematising, they are essentially the same discipline, and hence some kind of merger seems sensible [1]. This position is problematic, though. As we saw in Sect. 2, the philosophy of science is not only concerned with the study of systematising, but also with a variety of other topics. Hence the philosophy of science can at best be said to intersect with machine learning. Moreover, even where they intersect the aims of the two fields are rather different: e.g., the philosophy of science is primarily interested in explanation and hence theorising, while the area of machine learning and data mining is primarily interested in prediction and hence modeling. Perhaps automated scientific discovery is one area where the aims of machine learning and the philosophy of science coincide. In which case automated scientific discovery is the locus of intersection between the philosophy of science and machine learning. But this rather narrow intersection falls far short of the claim that the two disciplines are the same.

More plausibly, then, the philosophy of science and machine learning are not essentially one, but nevertheless they do admit interesting connections [2, 189; 3, 4]. In [4], I argue that the two fields admit a *dynamic interaction*. There is a dynamic interaction between two fields if there is a connection between them which leads to a mutually beneficial exchange of ideas, the direction of transfer of ideas between the two fields changes over time, and the fields remain autonomous [5]. Here, we shall take a look at two beneficial points of interaction: the lessons of automated scientific discovery for the study of scientific method (Sect. 4) and the influence of work on Bayesian epistemology and probabilistic causality on machine learning (Sect. 5).

4 Inductivism Versus Falsificationism

Scientific method is an important topic in the philosophy of science. How do scientists make discoveries? How *should* they make discoveries?

One view, commonly called *inductivism* and advocated by Bacon [6], is that science should proceed by first making a large number of observations and then extracting laws via a procedure that is in principle open to automation. An opposing position, called *falsificationism* and held by popper [7], is that the scientist first conjectures in a way that cannot be automated and then tests this conjecture by observing and experimenting to see whether or not the predictions of the conjecture are borne out, rejecting the conjecture if not.

While examples from the history of science have tended to support falsificationism over inductivism, the successes of automated scientific discovery suggest that inductivism remains a plausible position [8]. The approach of machine learning is to collect large numbers of observations in a dataset and then to automatically extract a predictive model from this dataset. In automated scientific discovery this model is usually also meant to be explanatory; hence, the model plays the same role

as scientific laws. To the extent that such procedures are successful, inductivism is successful. Gillies [8], Sect. 2.6 cites the GOLEM inductive logic programming system as an example of a machine learning procedure that successfully induced scientific laws concerning protein folding; this success was achieved with the help of humans who encoded background knowledge [8, Sect. 3.4]. Journals such as Data Mining and Knowledge Discovery and the Journal of Computer-Aided Molecular Design show that the inductive approach continues to produce advances. Moreover, the investment of drug and agrochemical companies suggests that this line of research promises to pay dividends. While the hope is that one day such companies might "close the inductive loop" – i.e., automate the whole cyclic procedure of data collection, hypothesis generation, further data collection, hypothesis reformulation ... – the present reality is that machine successes are achieved in combination with human expertise. The use by Dow AgroSciences of neural networks in the development of the insecticide spinetoram offers a recent example of successful human–machine collaboration [9]; spinetoram won the US Environmental Protection Agency 2008 Designing Greener Chemicals Award.

Perhaps human scientists proceed by applying falsificationism while machine science is inductivist. Or perhaps falsificationism and inductivism are but different approximations to a third view which better explicates scientific method. What could this third view be? It is clear that human scientists base their conjectures on a wide variety of different kinds of evidence, not just on a large number of homogeneous observations. (This – together with the fact that it can be hard for scientists to pin-point all the sources of evidence for their claims and hard for them to say exactly how their evidence informs their hypotheses – makes it hard to see how hypothesis generation can be automated. It is natural to infer, with falsificationists, that hypothesis generation can't be automated, but such an inference may be too quick.) On the other hand, most machine learning algorithms do take as input a large number of homogenous observations; this supports inductivism, but with the proviso that the successes of automated scientific discovery tend to be achieved in concert with human scientists or knowledge engineers. Human input appears to be important to fully utilise the range of evidence that is available. The third view of scientific method, then, is that a theory is formulated on the basis of extensive evidence (including background knowledge), but evidence which is often qualitative and hard to elucidate. The theory is revised as new evidence is accrued, and this new evidence tends to be accrued in a targeted way, by testing the current theory. Can this third way be automated? Contemporary machine learning methods require well-articulated quantitative evidence in the form of a dataset, but, as I suggest in Sect. 6, there is scope for relaxing this requirement and taking a fuller spectrum of evidence into account.

In sum, while not everyone is convinced by the renaissance of inductivism [see, e.g., 10], it is clear that automated scientific discovery has yielded some successes and that these have sparked new life into the debate between inductivism and falsificationism in the philosophy of science.

5 Bayesian Epistemology and Causality

Having looked at one way in which machine learning has had a beneficial impact on the philosophy of science, we now turn to the other direction: the impact of the philosophy of science on machine learning.

Epistemologists are interested in a variety of questions concerning our attitudes towards the truth of propositions. Some of these questions concern propositions that we already grant or endorse: e.g., do we know that $2 + 2 = 4$? if so, why? how should a committee aggregate the judgements of its individuals? Other questions concern propositions that are somewhat speculative: should I accept that all politicians are liars? to what extent should you believe that it will rain tomorrow?

Philosophers of science have been particularly interested in the latter question: to what extent should one believe a proposition that is open to speculation? This question is clearly relevant to scientific theorising, where we are interested in the extent to which we should believe current scientific theories. In the twentieth century philosophers of science developed and applied *Bayesian epistemology* to scientific theorising. The ideas behind Bayesian epistemology are present in the writings of some of the pioneers of probability theory – e.g., Jacob Bernoulli, Thomas Bayes – but only in recent years has it widely caught on in philosophy and the sciences.

One can characterise contemporary Bayesian epistemology around the norms that it posits:

Probability: The strengths of an agent's beliefs should be representable by probabilities. For example, the strength to which you believe it will rain tomorrow should be measurable by a number $P(r)$ between 0 and 1 inclusive, and $P(r)$ should equal $1 - P(\neg r)$, where $\neg r$ is the proposition that it will not rain tomorrow.

Calibration: These degrees of belief should be calibrated with the agent's evidence. For example, if the agent knows just that between 60% and 70% of days like today have been followed by rain, she should believe it will rain tomorrow to degree within the interval $[0.6, 0.7]$.

Equivocation: Degrees of belief should otherwise be as equivocal as possible. In the above example, the agent should equivocate as far as possible between r and $\neg r$, setting $P(r) = 0.6$, the value in the interval $[0.6, 0.7]$ that is closest to total equivocation, $P_=(r) = 0.5$.

So-called *subjective Bayesianism* adopts the Probability norm and usually the Calibration norm too. This yields a relatively weak prescription, where the extent to which one should believe a proposition is largely left up to subjective choice. To limit the scope for arbitrary shifts in degrees of belief, subjectivists often invoke a further norm governing the updating of degrees of belief: the most common such norm is *Bayesian conditionalisation*, which says that the agent's new degree of belief $P'(a)$ in proposition a should be set to her old degree of belief in a conditional on the new evidence e, $P'(a) = P(a|e)$.

In contrast to subjective Bayesianism, *Objective Bayesianism* adopts all three of the earlier norms – Probability, Calibration and Equivocation. If there are finitely many basic propositions under consideration (propositions that are not composed

out of simpler propositions), these three norms are usually cashed out using the *maximum entropy principle*: an agent's degrees of belief should be representable by a probability function, from all those calibrated with evidence, that has maximum entropy $H(P) = -\sum_{\omega \in \Omega} P(\omega) \log P(\omega)$. (The maximum entropy probability function is the function that is closest to the maximally equivocal probability function $P_=$ which gives the same probability $P_=(\omega) = 1/2^n$ to each conjunction $\omega \in \Omega = \{\pm a_1 \wedge \cdots \wedge \pm a_n\}$ of the basic propositions $a_1, \ldots, a_n$ or their negations, where distance from one probability function to another is understood in terms of cross entropy $d(P, Q) = \sum_{\omega \in \Omega} P(\omega) \log P(\omega)/Q(\omega)$.) Since these three norms impose rather strong constraints on degrees of belief, no further norm for updating need be invoked [11]. The justification of these norms and the relative merits of subjective and objective Bayesian epistemology are topics of some debate in philosophy [see, e.g., 12].

The development of Bayesian epistemology has had a profound impact on machine learning [13]. The field of machine learning arose out of research on expert systems. It was quickly realised that when developing an expert system it is important to model the various uncertainties that arise on account of incomplete or inconclusive data. Thus the system MYCIN, which was developed in the 1970s to diagnose bacterial infections, incorporated numerical values called "certainty factors." Certainty factors were used to measure the extent to which one ought to believe certain propositions. Hence, one might think that Bayesian epistemology should be applied here. In fact, the MYCIN procedure for handling certainty factors was non-Bayesian, and MYCIN was criticised on account of its failing to follow the norms of Bayesian epistemology. From the late 1970s, it was common to handle uncertainty in expert systems using Bayesian methods. And, when the knowledge bases of expert systems began to be learned automatically from data rather than elicited from experts, Bayesian methods were adopted in the machine learning community.

But Bayesian methods were rather computationally intractable in the late 1970s and early 1980s, and consequently systems such as Prospector, which was developed in the second half of the 1970s for mineral prospecting, had to make certain simplifying assumptions that were themselves questionable. Indeed considerations of computational complexity were probably the single biggest limiting factor for the application of Bayesian epistemology to expert systems and machine learning.

It took a rather different stream of research to unleash the potential of Bayesian epistemology in machine learning. This was research on causality and causal reasoning. In the early twentieth century, largely under the sceptical influence of Mach, Pearson and Russell, causal talk rather fell out of favour in the sciences. But it was clear that while scientists were reluctant to talk the talk, they were still very much walking the walk: associations between variables were being interpreted causally to predict the effects of interventions and to inform policy. Consequently, philosophers of science remained interested in questions about the nature of causality and how one might best reason causally.

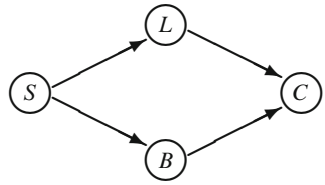


Fig. 1 Smoking (S) causes lung cancer (L) and bronchitis (B) which in turn cause chest pains (C)

Under the *probabilistic* view of causality, causal relationships are analysable in terms of probabilistic relationships – more specifically in terms of patterns of probabilistic dependence and independence [14]. Reichenbach, Good, Suppes and Pearl, pioneers of the probabilistic approach, developed the concept of a *causal net*. A causal net is a diagrammatic representation of causes and effects – such as that depicted in Fig. 1 – which has probabilistic consequences via what is now known as the *Causal Markov Condition*. This condition says that each variable in the net is probabilistically independent of its non-effects, conditional on its direct causes. If we complete a causal net by adding the probability distribution of each variable conditional on its direct causes, the net suffices to determine the joint probability distribution over all the variables in the net. Since the probabilities in the net tend to be interpreted as rational degrees of belief, and since the probabilities are often updated by Bayesian conditionalisation, a causal net is often called a *causal Bayesian net*. If we drop the causal interpretation of the arrows in the graph, we have what is known as a *Bayesian net*. The advantage of the causal interpretation is that under this interpretation the Markov Condition appears quite plausible, at least as a default constraint on degrees of belief [15].

Now, a causal Bayesian net – and more generally a Bayesian net – can permit tractable handling of Bayesian probabilities. Depending on the sparsity of the graph, it can be computationally feasible to represent and reason with Bayesian probabilities even where there are very many variables under consideration. This fact completed the Bayesian breakthrough in expert systems and machine learning. By building an expert system around a causal net, efficient representation and calculation of degrees of belief were typically achievable. From the machine learning perspective, if the space of models under consideration is the space of Bayesian nets of sufficiently sparse structure, then learning a model will permit efficient inference of appropriate degrees of belief. If the net is interpreted causally, we have what might be considered the holy grail of science: a method for the machine learning of causal relationships directly from data.

In sum, Bayesian epistemology offered a principled way of handling uncertainty in expert systems and machine learning, and Bayesian net methods overcame many of the ensuing computational hurdles. These lines of work had a huge impact: the dominance of Bayesian methods – and Bayesian net methods in particular – in the annual conferences on Uncertainty in Artificial Intelligence (UAI) from the 1980s is testament to the pervasive influence of Bayesian epistemology and work on probabilistic causality.

6 Evidence Integration

We now have some grounds for the claim that there is a dynamic interaction between machine learning and the philosophy of science: the achievements of automated scientific discovery have reinvigorated the debate between inductivists and falsificationists in the philosophy of science; on the other hand, work on Bayesian epistemology and causality has given impetus to the handling of uncertainty in machine learning. No doubt the mutually supportive relationships between philosophy of science and machine learning will continue. Here, we will briefly consider one potential point of interaction, namely the task of *evidence integration*.

The dominant paradigm in machine learning and data mining views the machine learning problem thus: given a dataset learn a (predictively accurate) model that fits (but does not overfit) the data. Clearly, this is an important problem and progress made on this problem has led to enormous practical advances. However, this problem formulation is rather over-simplistic in the increasingly evidence-rich environment of our information age. Typically, our evidence is *not* made up of a single dataset. Typically, we have a variety of datasets – of varying size and quality and with perhaps few variables in common – as well as a range of qualitative evidence concerning measured and unmeasured variables of interest – evidence of causal, logical, hierarchical and mereological relationships for instance. The earlier problem formulation just doesn't apply when our evidence is so multifarious. The Principle of Total Evidence, which holds that one should base one's beliefs and judgements on all one's available evidence, is a sound epistemological precept, and one that is breached by the dominant paradigm.

The limitations of the earlier problem formulation are increasingly becoming recognised in the machine-learning community. This recognition has led to a spate of research on what might be called *forecast aggregation*: a variety of models, each derived from a different dataset, are used to make predictions, and these separate predictions are somehow aggregated to yield an overall prediction. The aggregation operation may involve simple averaging or more sophisticated statistical meta-analysis methods.

But forecast aggregation itself has several limitations. First, it still falls foul of the Principle of Total Evidence: each model is based on a single dataset but qualitative evidence tends to be ignored. (Not always: qualitative evidence about relationships between the variables is sometimes invoked in the preprocessing of the datasets – if the knowledge engineer sees that a variable in one dataset is a subcategory of a variable in another dataset, these two variables might be unified in some way. But this data grooming is typically done by hand. As datasets involve more and more variables it is increasingly important that qualitative evidence be respected *as a part of the automation process*.) Second, it is unclear how far one should trust aggregated forecasts when they are often generated by models that not only disagree but are based on mutually inconsistent assumptions. Obviously in such cases at most one of the mutually inconsistent models is true; surely it would be better to find and use the true model (if any) than to dilute its predictions with the forecasts of false models. But this requires collecting new evidence rather than forecast aggregation. Third,

the general problem of judgement aggregation – of which forecast aggregation is but a special case – is fraught with conceptual problems; indeed the literature on judgement aggregation is replete with impossibility results, not with solutions [16].

In view of these problems, a better approach might be to construct a single model which is based on the entirety of the available evidence – quantitative and qualitative – and to use that model for predictions. Combining evidence is often called *knowledge integration*. However, available evidence may not strictly qualify as knowledge because it may, for reasons that are not evident, not all be true; hence *evidence integration* is better terminology.

Bayesian epistemology provides a very good way of creating a single model on the basis of a wide variety of evidence. As discussed in Sect. 5, the model in this case is (a representation of) the probability function that captures degrees of belief that are appropriate for an agent with the evidence in question. The evidence is integrated via the Calibration norm: each item of evidence imposes constraints that this probability function must satisfy. (Some kind of consistency maintenance procedure must of course be invoked if the evidence itself is inconsistent.) A dataset imposes the following kind of constraint: the agent's probability function, when restricted to the variables of that dataset, should match (fit but not overfit) the distribution of the dataset, as far as other evidence permits. Qualitative evidence imposes another kind of equality constraint, as follows. A relation R is an *influence relation* if learning of a new variable that does not stand in relation R to (i.e., does not influence) the current variables does not provide grounds for changing one's degrees of belief concerning the current variables. Arguably causal, logical, hierarchical and mereological relationships are influence relations. Hence, evidence of such relationships imposes equality constraints of the form: the agent's probability function, when restricted to variables that are closed under influence, should match the probability function that the agent would have adopted were she only to have had evidence concerning that subset of variables, as far as other evidence permits. Hence, both quantitative and qualitative evidence impose certain equality constraints on degrees of belief. See [15] and [17] for the details and motivation behind this kind of approach.

In sum, evidence integration has greater potential than forecast aggregation to circumvent the limited applicability of the current machine learning paradigm. Bayesian epistemology is strikingly well-suited to the problem of evidence integration. Hence, there is scope for another fruitful interaction between philosophy of science and machine learning.

Example: Cancer Prognosis

As an illustration of the kind of approach to evidence integration that Bayesian epistemology offers, we shall consider an application of objective Bayesian epistemology to integrating evidence for breast cancer prognosis. This application is described in detail in [18].

When a patient has breast cancer and has had surgery to remove the cancer it is incumbent on the relevant medical practitioners to make an appropriate onward treatment decision. Broadly speaking, more effective treatments are more aggressive in the sense that they have harsher side effects. Such treatments are only warranted to the extent that the cancer is likely to recur without them. The more strongly the medical practitioner believes the cancer will recur, the more aggressive the treatment that will be instigated. It is important, then, that the agent's degree of belief in recurrence is appropriate given the available evidence.

Evidence here – as in many realistic applications – is multifarious. There are clinical datasets detailing the clinical symptoms of past patients, genomic datasets listing the presence of various molecular markers in past patients, scientific papers supporting particular causal claims or associations, medical experts' causal knowledge, and information in medical informatics systems, including medical ontologies, and previous decision support systems such as argumentation systems.

In [18] we had the following sources of evidence available. First, we had a clinical dataset, namely the SEER study, which involves three million patients in the US from 1975–2003, including 4,731 breast cancer patients. We also had two molecular datasets, one with 502 cases and another with 119 cases; the latter dataset also measured some clinical variables. Finally, we had a published study which established a causal relationship between two variables of interest.

These evidence sources impose constraints on an agent's degrees of belief, as outlined earlier. The agent's degrees of belief should match the dataset distributions on their respective domains. Moreover, degrees of belief should respect the equality constraints imposed by knowledge of causal influence. While the Probability norm holds that the strengths of the agent's beliefs should be representable by a probability function, the Calibration norm holds that this probability function should satisfy the constraints imposed by evidence.

These two norms narrow down the choice of belief function to a set of probability functions. But objective Bayesian epistemology imposes a further norm, Equivocation. Accordingly, the agent's degrees of belief should be representable by a probability function from within this set that is maximally equivocal. On a finite domain, this turns out to be the (unique) probability function in this set that has maximum entropy. So, objective Bayesian epistemology recommends that treatment decisions be based on this maximum entropy probability function.

As discussed in Sect. 5, from a computational point of view it is natural to represent a probability function by a Bayesian net. A Bayesian net that represents a probability function that is deemed appropriate by objective Bayesian epistemology is called an *objective Bayesian net* [19]. This Bayesian net can be used for inference – in our case to calculate degree to which one ought to believe that the patient's cancer will recur. Figure 2 depicts the graph of the objective Bayesian net in our cancer application. At the top is the recurrence node, beneath which are clinical variables. These are connected to five molecular variables at the bottom of the graph. Hence, one can use both molecular markers and clinical symptoms to predict

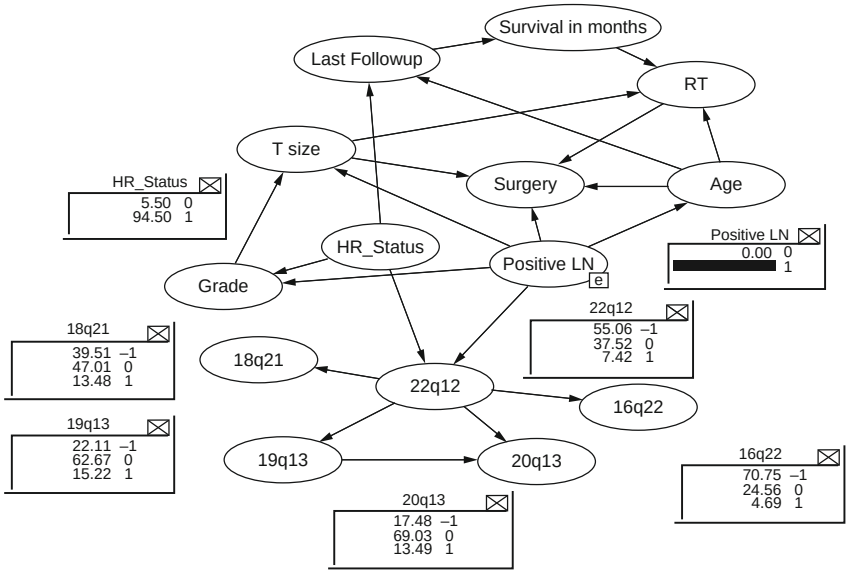


Fig. 2 Graph of an objective Bayesian net for breast cancer prognosis

the patient's survival, even though no dataset contains information about all these variables together. The objective Bayesian net model succeeds in integrating rather disparate evidence sources.

7 Conclusion

Machine learning in general and automated scientific discovery in particular have a close relationship with the philosophy of science. On the one hand, advances in automated scientific discovery have lent plausibility to inductivist philosophy of science. On the other hand, advances in probabilistic epistemology and work on causality have improved the ability of machine learning methods to handle uncertainty.

I have suggested that inductivism and falsificationism can be reconciled by viewing these positions as approximations to a third view of scientific method, one which considers the full range of evidence for a scientific hypothesis. This third way would be an intriguing avenue of research for philosophy of science. I have also suggested that the current single-dataset paradigm in machine learning is becoming increasingly inapplicable, and that research in machine learning would benefit from serious consideration of the problem of formulating a model on the basis of a broad range of evidence. Bayesian epistemology may be a fruitful avenue of research for tackling evidence integration in machine learning as well as evidence integration in the philosophy of science.

Acknowledgements This research was supported by the Leverhulme Trust. I am very grateful to Drake Eggleston and Peter Goodfellow for helpful leads.

References

1. K.B. Korb, Machine Learning as Philosophy of Science, ed. by K. Korb, H. Bensusan. in *Proceedings of the ECML-PKDD-01 Workshop on Machine Learning as Experimental Philosophy of Science*, 2001
2. P. Thagard, *Computational Philosophy of Science* (MIT, Cambridge, MA, 1988)
3. H. Bensusan, in *Is Machine Learning Experimental Philosophy of Science?* ed. by A. Aliseda, D. Pearce. ECAI2000 Workshop Notes on Scientific Reasoning in Artificial Intelligence and the Philosophy of Science (ECAI, Berlin, 2000), pp. 9–14
4. J. Williamson, Mind Mach. **14**(4), 539–549 (2004)
5. D. Gillies, Y. Zheng, Theoria **16**(3), 437–459 (2001)
6. F. Bacon, *The New Organon*, ed. by L. Jardine, M. Silverthorne (Cambridge University Press, Cambridge, 2000)
7. K.R. Popper, *Conjectures and Refutations* (Routledge and Kegan Paul, 1963)
8. D. Gillies, *Artificial Intelligence and Scientific Method* (Oxford University Press, Oxford, 1996)
9. T.C. Sparks, G.D. Crouse, J.E. Dripps, P. Anzeveno, J. Martynow, C.V. DeAmicis, J. Gifford, J. Comp. Aided Mol. Des. **22**, 393–401 (2008)
10. J.F. Allen, BioEssays **23**, 104–107 (2001)
11. J. Williamson, Synthese (2009), doi 10.1007/s11229-009-9515-y
12. J. Williamson, in *Motivating Objective Bayesianism: From Empirical Constraints to Objective Probabilities*, ed. by W.L. Harper, G.R. Wheeler. Probability and Inference: Essays in Honour of Henry E. Kyburg Jr. (College Publications, London, 2007), pp. 151–179
13. D. Gillies, in *Handling Uncertainty in Artificial Intelligence, and the Bayesian Controversy*, ed. by F. Stadler. Induction and Deduction in the Sciences (Kluwer, Dordrecht, 2003)
14. J. Williamson, *Probabilistic Theories of Causality*, ed. by H. Beebee, C. Hitchcock, P. Menzies. The Oxford Handbook of Causation (Oxford University Press, Oxford, 2008d)
15. J. Williamson, *Bayesian Nets and Causality: Philosophical and Computational Foundations* (Oxford University Press, Oxford, 2005a)
16. J. Williamson, J. Logic Comput. **19**, 461–473 (2009)
17. J. Williamson, in A. Carsetti (ed.), Causality, meaningful complexity and knowledge construction, Springer 2010
18. S. Nagl, M. Williams, J. Williamson, in *Objective Bayesian Nets for Systems Modeling and Prognosis in Breast Cancer*, ed. by D. Holmes, L. Jain. Innovations in Bayesian Networks: Theory and Applications (Springer, Berlin, 2008)
19. J. Williamson, in *Objective Bayesian Nets*, ed. by S. Artemov, H. Barringer, A.S. d'Avila Garcez, L.C. Lamb, J. Woods. We Will Show Them! Essays in Honour of Dov Gabbay, vol. 2 (College Publications, London, 2005b), pp. 713–730

“This page left intentionally blank.”

Concept Formation in Scientific Knowledge Discovery from a Constructivist View

Wei Peng and John S. Gero

1 Introduction

The central goal of scientific knowledge discovery is to learn cause–effect relationships among natural phenomena presented as variables and the consequences their interactions. Scientific knowledge is normally expressed as scientific taxonomies and qualitative and quantitative laws [1]. This type of knowledge represents intrinsic regularities of the observed phenomena that can be used to explain and predict behaviors of the phenomena. It is a generalization that is abstracted and externalized from a set of contexts and applicable to a broader scope. Scientific knowledge is a type of third-person knowledge, i.e., knowledge that independent of a specific enquirer. Artificial intelligence approaches, particularly data mining algorithms that are used to identify meaningful patterns from large data sets, are approaches that aim to facilitate the knowledge discovery process [2]. A broad spectrum of algorithms has been developed in addressing classification, associative learning, and clustering problems. However, their linkages to people who use them have not been adequately explored. Issues in relation to supporting the interpretation of the patterns, the application of prior knowledge to the data mining process and addressing user interactions remain challenges for building knowledge discovery tools [3]. As a consequence, scientists rely on their experience to formulate problems, evaluate hypotheses, reason about untraceable factors and derive new problems. This type of knowledge which they have developed during their career is called "first-person" knowledge. The formation of scientific knowledge (third-person knowledge) is highly influenced by the enquirer's first-person knowledge construct, which is a result of his or her interactions with the environment. There have been attempts to craft automatic knowledge discovery tools but these systems are limited in their capabilities to handle the dynamics of personal experience. There are now trends in developing approaches to assist scientists applying their expertise

W. Peng (✉)
Platform Technologies Research Institute, School of Electrical and Computer Engineering, RMIT University, Melbourne, VIC 3001, Australia
e-mail: w.peng@rmit.edu.au

M.M. Gaber (ed.), *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*, DOI 10.1007/978-3-642-02788-8_5,

to model formation, simulation, and prediction in various domains [4], [5]. On the other hand, first-person knowledge becomes third-person theory only if it proves general by evidence and is acknowledged by a scientific community. Researchers start to focus on building interactive cooperation platforms [1] to accommodate different views into the knowledge discovery process.

There are some fundamental questions in relation to scientific knowledge development. What are major components for knowledge construction and how do people construct their knowledge? How is this personal construct assimilated and accommodated into a scientific paradigm? How can one design a computational system to facilitate these processes? This chapter does not attempt to answer all these questions but serves as a basis to foster thinking along this line. A brief literature review about how people develop their knowledge is carried out through a constructivist view. A hydrological modeling scenario is presented to elucidate the approach.

2 Concept Formation from a Constructivist View

Cognitive science is a multi-disciplinary study with the aim to deliver a theory of intelligence. The basic assumption held by many cognitive science researchers is that there is a common set of principles underlying all instances of intelligence [6]. Aristotle attributed to perception and observation essential roles in acquiring scientific knowledge and proposed an empirical method of gathering observations followed by taxonomic classification, interpretation and inference [6].

2.1 Knowledge as Generalization

One aspect of human cognition is to develop experience and use experience to construct a judgment, in which a certain object is distinguished from other objects and is characterized by some concepts. Concepts are bearers of meanings. The enquiry for the notion of concept has long been the focus of research in philosophy. Both the notion of "concept" and our understandings of the way in which concepts are formed have evolved. John Locke [7] described that a general idea corresponds to a description of concept, which is created by abstracting and drawing commonalities from the particulars. The assumption of a concept as a consequence of induction is that the unobserved events conform to regularities in the already known facts. The assumption is not general enough to address possible exceptions in a dynamic complex world. David Hume [8] argued that discovering the "necessary connexion" between objects leads to a better understanding of *a priori* causation relationships around these objects. He also mentioned that relationships between ideas of causation can be derived from our experience [9]. David Hume's theories inspired Immanuel Kant, who later developed the notion of "a posteriori" concept. According to Kant, a concept can be further defined as an a posteriori concept or an a priori concept [10].

A posteriori or empirical concepts are abstracted/induced from specific perceived events and are applicable to them. *A priori* concepts are categories that are not abstractions from perception but are applicable to it. Although there is no unified definition of a concept, a concept is often taken to mean a mental representation of a class or a category [11]. The classical view of concept formation as abstraction or "abstract thinking," as outlined by Van Oers [12], emphasizes creating types, a process of generalizing by removing circumstantial aspects of time and place [13]. This view has been manifested in the recent research on concept formation systems in the field of artificial intelligence. Many researchers consider categorization as the essence of a concept and its formation. Concept formation has been regarded as a process of incremental unsupervised acquisition of categories and their intentional descriptions [14]. Based on this view, a broad spectrum of computational models has been developed, including inductive learning methods, explanation-based learning approaches and connectionist algorithms. However, theories of concept formation that merely focus on categorization are not able to address the complexity of the world [15]. A concept lacking an understanding of why and how the object, entity or event has its particular properties is called a protoconcept [15], [16]. The process by which people form any category knowledge biases the knowledge contents formed. Theories and models are people's approximations to the uncertain (and unknown) a priori concepts that describe the universe. They are by nature a posteriori. For example, the concept of gravitation in modern physics is cognized differently to what was taken for granted in Newton's time.

2.2 Knowledge as Construction

Knowledge is an empirical term therefore inseparable from a subject and the world external to that subject. In describing the relationship between human experience and nature, John Dewey [17] pointed out that there is a union of experience and nature in natural science. Human experience is considered as "a means of continually penetrating the hearts of the reality of nature." The enquirer must resort to his or her experience and use empirical methods if his or her findings are to be treated as genuinely scientific [17]. Referring to the genesis of knowledge, Dewey [17] mentioned that knowledge is a refined experience:

> *"The intrinsic nature of events is revealed in experience as the immediately felt qualities of things. The intimate coordination and even fusion of these qualities with regularities that form the objects of knowledge, in the proper sense of the word 'knowledge', characterizes intelligently directed experience, as distinct from mere casual and uncritical experience".*

Dewey's vision can also be found in other constructivists' works. In reviewing Piaget's conception of knowledge and reality, Von Glasersfeld [18] conceived that the cognitive organism is first and foremost an organizer who interprets experience and shapes it into a structured world. Piaget used the term "schema" to

denote the basic mental structure, which depicts how perceptual categories are organized. After examining the development of intelligence in children, Piaget [19] concluded that two intertwined processes ("assimilation" and "accommodation") and their coordination enable a child to construct knowledge from his or her experience. Assimilation tends to subordinate the environment to the organism's *a priori* or acquired schemata, whereas accommodation adapts the organism to the successive constraints of the environment by updating (or incorporating) new schemata [19]. Piaget mentioned [18], [20]:

> *"All knowledge is tied to action and knowing an object or an event is to use it by assimilating it to an action scheme ..... this is true on the most elementary sensory-motor level and all the way up to the highest logical-mathematical operations ....."*

Piaget's theory of assimilation can be interpreted as the mapping between the external world to the existing knowledge structures in the internal world of a cognitive organism. The development of new knowledge structures in the internal world of a cognitive agent is through the accommodation mechanism (Fig. 1).

Vygotsky [21] introduced the role of "activity" in knowledge construction, stating that the activities of the mind cannot be separated from overt behavior, or from the social context in which they occur. Social and mental structures interpenetrate each other [21], [22].

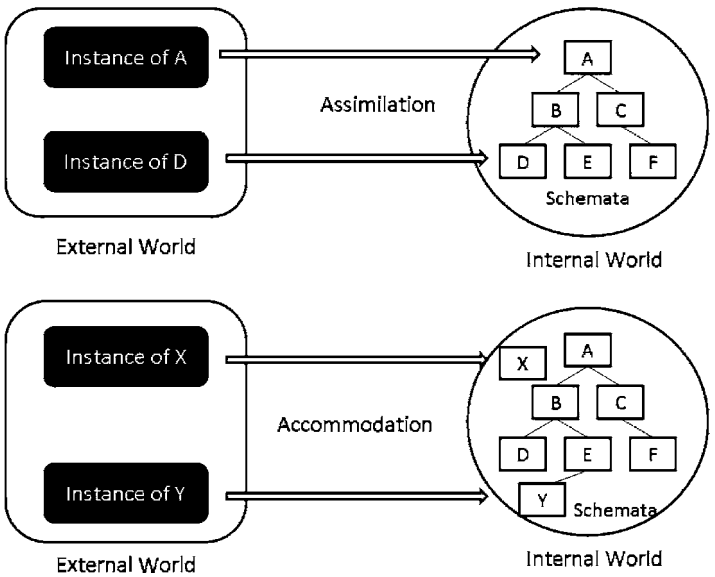


Fig. 1 An interpretation of Piaget's cognition processes of assimilation and accommodation

2.3 Experiential Learning

These theories provide insights about ingredients for knowledge construction based on the external events or context, the internal knowledge structures and the interactions between the assimilation and accommodation processes. Kolb [23] developed an experiential learning theory by drawing ideas from Dewey [24], Lewin [25], and Piaget [19]. The major components for learning knowledge consist of concrete experience, observation and reflection, the formation of abstract concepts and testing of implications of concepts in new situations. He represented this idea in the experiential learning circle, which is illustrated in Fig. 2.

Kolb and Fry [26] postulated that learning can commence from any component and operates in a continuous cycle. The observations are assimilated into a "theory" based on the immediate concrete experience. The theory is a generalization that can be used to deduce new implications for action. The implications then serve as guides for creating new experience. This model provides operational features for Piaget's notion of "mutual interaction" between assimilation and accommodation. Jarvis [27] enriched the internal process for knowledge construction by introducing flexible behaviors of an intelligent system. He put experience and memory in the loop. As illustrated in Fig. 3, these behaviors[1] include:

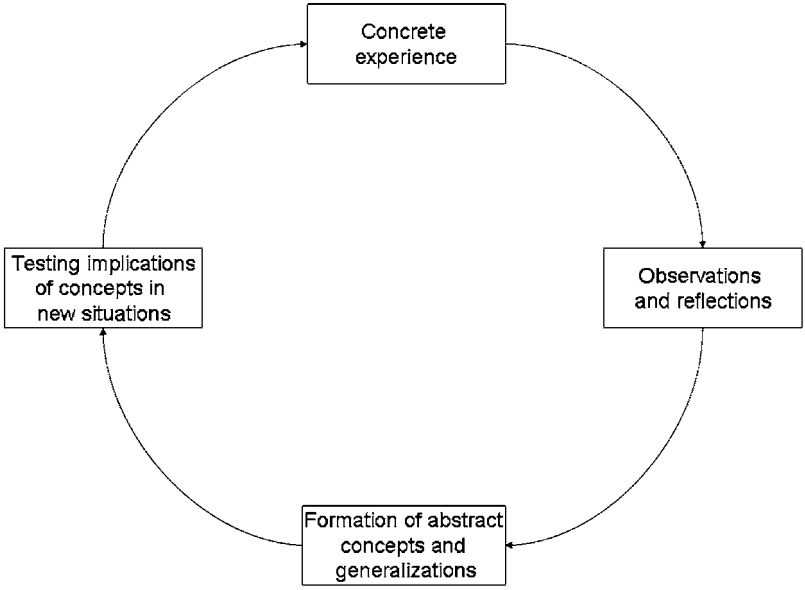


Fig. 2 Experiential learning cycle (Adapted from Fig. 2.1 of [23])

[1] The Rejection behavior where the individual refuses to learn from the situation is not list here due to its irrelevance to knowledge construction.

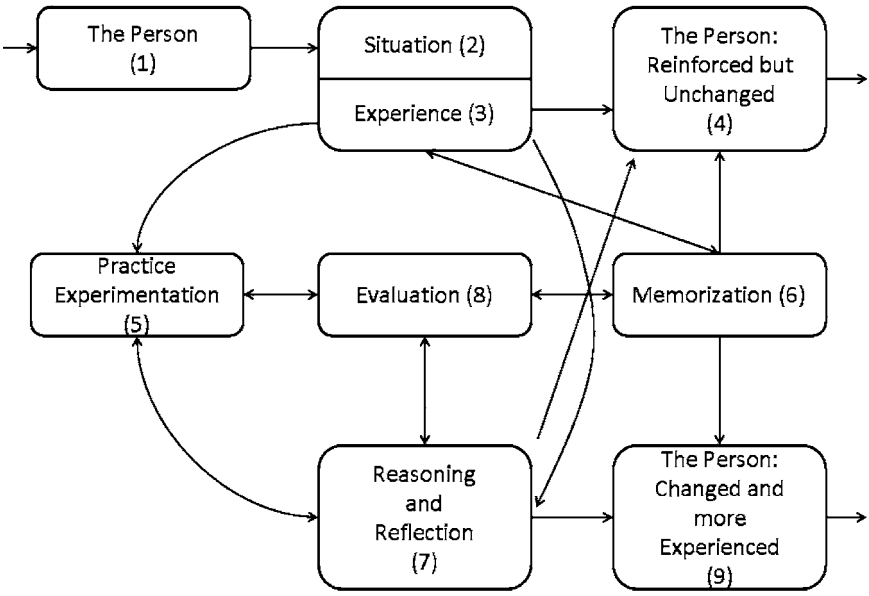


Fig. 3 Experiential learning path from Jarvis [27], [28]. Boxes represent components that consist of the processes of learning. These components are general notions that can be either entities or processes. Numbers refer to individual processes described in the text

- Non-learning presumption (or non-consideration) (boxes $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$) where people react through experience (or choose not to respond to environmental events)
- Non-reflective pre-conscious (boxes $1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 4$ or 9) where people have experience about environmental events but do not attend to it
- Non-reflective practice (boxes $1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 6 \rightarrow 4$ or 9) where people obtain basic skills
- Non-reflective memorization (boxes $1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 8 \rightarrow 4$ or 9) where people memorize new things
- Reflective contemplation (boxes $1 \rightarrow 2 \rightarrow 3 \rightarrow 7 \rightarrow 8 \rightarrow 6 \rightarrow 9$) where people reflect upon a situation and make decisions
- Reflective practice (boxes $1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 5 \rightarrow 8 \rightarrow 6 \rightarrow 9$) where the individuals reflect and then act upon a situation
- Reflective experimental learning ($1 \rightarrow 2 \rightarrow 3 \rightarrow 7 \rightarrow 5 \rightarrow 7 \rightarrow 8 \rightarrow 6 \rightarrow 9$) in which people reason about the situation and evaluate their experience.

Jarvis' experiential learning model[2] emphasizes the role of experience and cognition in developing new knowledge. Another component of Jarvis' learning model

[2] Jarvis' recent learning model [28], [29]) extends the experiential learning to lifelong learning theory and has an emotive component.

is the pragmatic unit which includes experiment and evaluation processes. A key notion which is implicit in Jarvis' theory but put forward by Kolb is "prediction" (termed as "implication for a concept" or "hypothesis" in [23]). Anticipation plays a key role in human cognition. Human activity of knowing (scientific thoughts and their epistemological interpretations) are the highest form of adaptation, which involves reflecting on past experience, abstracting specific regularities from them, and projecting these as predictions into the future [30]. This behavior of knowing is also described in the "memory prediction framework" [31] and is conjectured as the basic function of our memory system. Our memory serves as a "metaphysical linkage of times past and future" [32]. This means that our memory system constantly uses experience to form invariant representations[3] about the spatial-temporal (and feature) based events, predicts potential occurrences in the environment and biases behaviors of our sensory-motor system accordingly. Based on this understanding, this chapter will present the predominant theory of memory in constructivism and elaborate a concept formation process.

2.4 Concept Formation in Constructive Memory

Memory in cognitive science does not mean a place or device storing descriptions of actions. Neither does it refer to the information that is encoded, stored and retrieved. The view of memory as an encoding-storage-retrieval device cannot account for phenomena such as "false recognition," "intrusion" and "confabulation" [33]. Memory is predominantly conceived as a mental capability for us to make sense, to learn new skills and to compose something new [13]. The constructive view of memory can be traced back to Dewey in the *The Reflex Arc Concept in Psychology* (quoted by [34]):

> *"Sequences of acts are composed such that subsequent experiences categorize and hence give meaning to what experienced before."*

The theory of constructive memory is supported by many cognitive studies [35]–[37]. The basic functions of a constructive entity are described by Riegler [33] as:

1. The cognitive apparatus creates a structure
2. It interacts with other structures (such as the surrounding environment or the older structure of its apparatus)
3. It compares the newly created structures with those encountered in step 2
4. It adapts the structures when needed before returning to step 1.

The cognitive apparatus tends to maintain the fitness of a knowledge structure in relation to the changing environment. For example, if you have never been to

[3] Invariant representations are knowledge structures that can be used to classify an object or infer potential acts. For example, the knowledge structures you rely on to recognize a friend's face should always work even if the distance and the orientation of your friend's face vary.

Australia, your representation of "swan" is induced from your Northern Hemisphere exposure to a type of elegant white birds from the Anatidae family. Suppose someone tells you that there are black swans in the Southern Hemisphere, you then carry out a literature search. You find that there indeed exist black swans and then realize the concept of swan needs to be adapted to accommodate this new information. Such characteristics of our memory system are in concordance with the reflection and evaluation processes of experiential learning. The essence of the constructive memory is the means of adapting knowledge structures to a dynamic environment. Reigler's constructive memory functions emphasize changes of knowledge structures during a construction process. However, the lack of descriptions of macroscopic behaviors of the involved person leads to speculations about how the structures will change. It is suggested that describing the constructive memory in the context of human experiential learning behavior leads to operational features of a memory system.

The constructive memory process may be viewed as the process of transforming information during interactions. An important notion introduced to illustrate the concept formation process in a constructive memory system is "grounding." Symbolic grounding explores the means by which the semantic interpretation of a formal symbol system can be made intrinsic to that system, rather than relying on the meanings in the head of a third-person interpreter or observer [38]. The grounding problem generally refers to representation grounding [39] or grounding of a concept, in which the concept can be developed through interactive behavior in an environment [40]. A grounding process here is referred to as the evaluation of whether constructed knowledge structures correctly predict environmental changes. The basic information units are "stimuli", "anticipation", "proto-concept", "hypothesis", "concept" and "invariants". Stimuli denote environmental data prior to a constructive memory process. Anticipation is the term for responsive information based on the agent's[4] experience, predicting potential environmental changes. A concept is a result of an interaction process in which meanings are attached to environmental observations. We use the term "proto-concept" to describe the intermediate state of a concept. A proto-concept is a knowledge structure that depicts the agent's interpretations and anticipations about its external and internal environments at a particular time. The term "invariant" has been defined in the previous section as the knowledge structures that an agent uses to identify categories and predict potential acts in the environment. In the scientific knowledge discovery context, this conceptual knowledge structure is composed of sets of categorized abstractions and causal relationships between various observations in various spatial-temporal scales. The term "hypothesis" is associated with the agent's explanations for discrepancies between its prediction and environmental changes. The grounding of proto-concepts and derived anticipations (or hypotheses) produces a concept. We define concepts as the grounded invariants over the agent's experience. They are abstractions of experience that confer a predictive ability for new situations [41], [42]. On the other hand, a

[4] We use the term agent to represent cognitive apparatus.

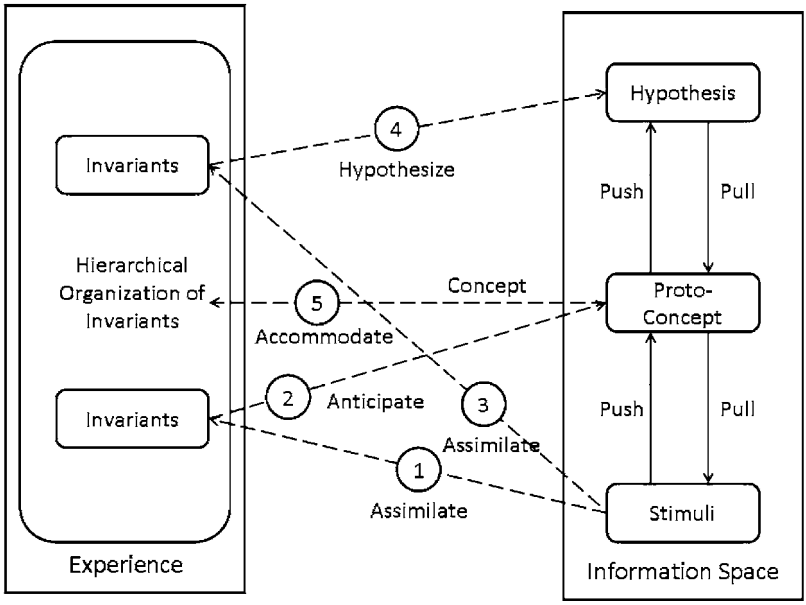


Fig. 4 A view of concept formation in constructive memory

concept contains context-dependent specifications for an abstraction, which are encapsulated in anticipations. The concept formation mechanism (described in Fig. 4) consists of the following processes:

1. Process 1 is where the agent assimilates the encountered stimuli to activate a response from its experience.
2. In Process 2, the agent generates a proto-concept which contains the categorical information and the related prediction.
3. Process 3 is where the agent assimilates the stimuli to create hypotheses for invalid predictions. This process is called reflection.
4. In Process 4, the agent produces a hypothesis based on the deduced explanations and the re-activated experience.
5. In Process 5, the agent accommodates the concept (validated proto-concept) into experience.
6. The "push" process is a data-driven process where changes in the environment (or autogenous variables of the agent) trigger the transformation of these changes into changes in the experience [43].
7. The "pull" process is an anticipation-driven process where if the process for validating a proto-concept (or a hypothesis) requires the agent to obtain a particular pattern of information from the environment, then the process is biased in the way that external variables (or autogenous variables of the agent) are filtered out or emphasized [43].
8. The validation process compares the prediction with the pulled data to determine the validity of a proto-concept.

The "push" and "pull" processes are two major components in charge of knowledge structure transformations. The push process uses Processes 1–4 to achieve its goal, which is to transform changes from the environment to the agent's experience to acquire a response. The pull process performs anticipation-driven data acquisition and data construction, which underpins the validation process.

As illustrated in Fig. 4, the agent processes the environmental data in the push process, in which data (stimuli) are transformed into categorical information and predictions based on invariants held in its experience (Processes 1 and 2). The agent subsequently tests the validity of its proto-concepts in a pull process. An ill-formed proto-concept triggers hypothesis-testing processes (Processes 3 and 4) in which the agent reasons and creates explanations for discrepancies between the proto-concepts and observations. The explanations can be used to deduce hypotheses and the associated observation requirements. A well-grounded proto-concept (obtained after processes push and pull) is called a concept and accommodated into the agent's knowledge structures as experience (in Process 5).

2.5 From First Person Construct to Third Person Knowledge

An individual's knowledge can be treated as first-person constructs, which are biased by a person's past experience in his or her interactions with the social context. Scientists build models and theories that represent their individual interpretations of natural phenomena. This type of knowledge is then transformed into a type of generalized knowledge that can apply to nonspecific events and is independent of their enquirers. This generalized knowledge once supported by scientific evidence becomes instances of a scientific paradigm that is a form of third-person knowledge. Thomas Kuhn described the notion of paradigm from his philosophical viewpoint of science in *The Structure of Scientific Revolutions* [44]. A scientific paradigm refers to scientific achievements that attract an enduring stream of a scientific community to follow the same rules and standards for their practices. A scientific paradigm centers a space for people to redefine and resolve problems. The development route of "normal science," which is built upon past paradigms, is a nonaccumulative multiple trajectories of fact-gathering, paradigm formation and paradigm revolution [44].

According to Kuhn [44], the reconstruction of prior theory and the re-evaluation of prior facts enable a new theory to be assimilated. A normal scientific paradigm tends to reject radically new theories. Normal science is to fit nature into the rigid, incomplete explanatory framework that a paradigm defined. A scientific community is likely to accept and encourage esoteric research works [44]. An individual's effort is limited and therefore a scientific revolutionary process is seldom achieved by a single person and never overnight [44].

Supporting both individuals and a scientific community to accommodate new and emergent knowledge holds the key for scientific discovery. This requires any computer-based discovery support tool to adopt constructive and interactive approaches. In Sect. 3 scientific concept formation based on a hydrological modeling scenario is presented to exemplify this idea.

3 Concept Formation in a Hydrologic Modeling Scenario

The movement of water through a permeable medium (i.e., soils) has been widely accepted as a complex phenomenon. The soil water behaviors are dynamic in the sense that precipitation, soil texture and profile, the presence of plants, land use and a variety of meteorological variables influence the spatial distribution and temporal evolution of soil moisture [45]. At the same time, water penetration into the soil changes the physical properties of the soil, which can further impact on the soil water behaviors over time. Precise soil moisture measurement to obtain ground truth in this dynamic environment is logically and economically infeasible. Therefore, approximation and simulation become dominant approaches for hydrologists. Hydrological models are simplified representation of the water system within the environment, which can assist hydrologists to understand, explain, and predict the behavior of water. Three paradigms of hydrology modeling are empirical, physical and conceptual models [46]. The aim of an empirical modeling approach is to unveil relationships between hydrologic variables that have been observed from the environment without considering the complex hydrologic processes. It is a black box approach based on data. Many data mining and analytic approaches can be used to obtain empirical models. For example, various artificial neural networks have been used in soil classification and rainfall runoff mapping. Physical hydrologic models are based on the knowledge of physical laws for water behavior in different media. They are presented in parameterized equations, which state the understanding of relationships of hydrologic variables. Conceptual models are developed based on hydrologists' endeavors to leverage the data-driven models and their physical interpretations. Hydrologists harness their knowledge of physical laws and apply data mining/analytic tools on observations to learn concepts, which are further formalized as new or validated models.

In this section, a hydrologic modeling scenario based on the slope discharge model (developed by Michel [47] and cited by Baird [48]) is described as an example of scientific knowledge discovery. It is reasonable to consider hill-slope as a single quadratic reservoir rather than attempt to model dynamic flows within the slope. This is because hydrologic flows at a small scale are strongly heterogeneous and currently not possible to model (and validate). However, small-scale complexity can lead to simplicity at a larger scale if certain principal properties exist [48]. As a result, hydrologists always resort to empirical relations obtained from empirical approaches. A simple empirical relation between total volume of water and discharge from the soil-filled trough can be formulated as a quadratic function [47]:

$$Q = \frac{V^2}{\tau \times V_0} \tag{1}$$

where Q is the total discharge from the slope (with physics dimensions L^3T^{-1}), V is the volume of drainable water remaining in the hill-slope, V_0 is the total volume of water that can drain from the slope (L^3), and τ is the system parameter (T). Empirical lumped reservoir models like (1) can be effective if coupled with

real-time observations and calibrations. However, empirical models are criticized as not physically meaningful [49]. In another camp, hydrologists use their understanding of physical constraints to formulate soil water behavior observed in the laboratory settings. For example, Darcy's Law (cited by Baird [48]) introduces physical relationship between water flux and the hydraulic energy that drives the water movement, describing the flow in saturated soils:

$$Q_f = -K\frac{\mathrm{d}h}{\mathrm{d}x} \tag{2}$$

where Q_f[5] is the discharge *per unit area* (with physics dimensions $L^3T^{-1}L^{-2}$), K is the hydraulic conductivity (with physics dimensions LT^{-1}), denoting intrinsic permeability of the soil, h is hydraulic pressure head, and x is the distance in the direction of flow (L). The unsaturated version of Darcy's law is Richards' equation [50]. Representative models obtained from the laboratory environment (like (1)–(2)) are of limited usefulness for real heterogeneous slopes [51]. The applicability of a laboratory-induced model to real-time situations depends on how well one can address uncertainties in initial conditions, time-varying inputs and time-invariant parameters. Applying these laboratory-based models to real-time environment is extremely expertise demanding. The deterministic approach is to use observations to inversely calibrate the parameters of a model (like the Darcy–Richards equation) [52], [53]. However, it is argued that deterministic models are not accurate representations of the real world. There is a growing interest in the inductive paradigm in the hydrologic literatures [54], focusing on learning regularities from large volumes of near realtime data obtained from observations. In the mean time, scientific expertise in terms of domain knowledge remains a critical factor in providing physically meaningful interpretations (i.e., causalities) to empirical relationships learned from data.

A hydrologist learns a relationship between the target variable and other variables by investigating their dependencies, Fig. 5. The State Dependent Parameter [55] analytic approach may be used to this end. An assumption for this scenario is that water discharge (Q) appears to be more strongly dependent on reservoir water (V) than any other variables. The hydrologist uses statistical approaches to identify the structure and order of the model. For example, the Akaike Information Criterion [55] (cited by Young, et al. [54]) tends to identify over-parameterized models. The model structure is identified as quadratic in this scenario.

The parameters that characterize the model are estimated by optimization techniques such as Maximum Likelihood (cited by Young et al. [54]). In this way, the hydrologist applies his or her experience and domain theories to construct a proto-concept that is formulated into (1) by assimilating the environmental variables (Process 1). A slope discharge prediction is generated afterward (Process 2). The model is subsequently evaluated using the data captured by various sensors in a

[5] We use Q_f to depict the finite element of the discharge therefore the sum-up of Q_f at the bottom of the slope makes Q in (1) that is the total discharge from the slope during a specific time.

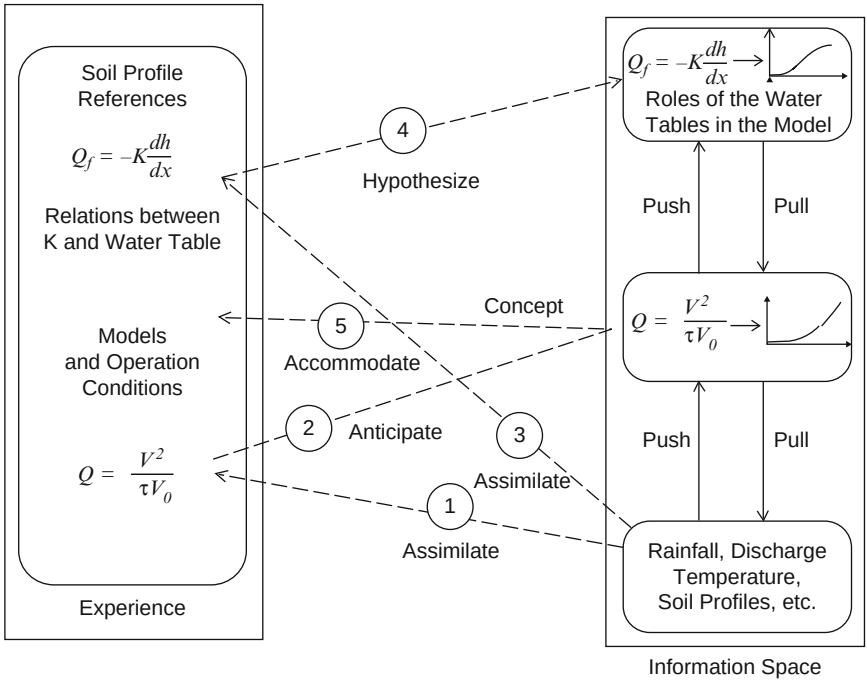


Fig. 5 A scenario of hydrologic concept formation

pull process.[6] Since an empirical model represents a coarse approximation of the related natural phenomenon, contextual information including operational conditions should be attached to the model.

When complexity cannot be ignored, the modeler relies on domain knowledge to extend this model by introducing more variables in the model. For example, he or she may add new data like soil moisture index and soil profile distribution to compensate for nonlinear errors. After he or she submitted the result to a community for reviewing, he or she was criticized for not using alternative models to validate the result. So, the modeler resorted to alternative physical-based models like the Darcy–Richards Equation (Equation 2) in a hypothesizing process (Processes 3 and 4) to provide cross-validation and physically meaningful explanations. If there is no reasonable model suited to the observations, scientists need to give explanations and re-define model application conditions. For example, as mentioned by Baird [48], the misfit between the Darcy-Richards model and observations can be explained by the heterogeneous water behavior in different media. The initial high rate of discharge from the slope is due to the rapid flow of water in macropores in the unsaturated zone, when these empty, Darcian flow processes in the saturated zone

[6] The data used for model validation should be different to those used for model estimation.

become dominant. However, the phenomenon can also be explained by the roles of the water table. The water table rises after a rainstorm. This causes a general increase in hydraulic conductivity of the soil and as a consequence, outflow from the base of the slope increases rapidly [48]. Scientists need to observe the environment and evaluate the validity of these explanations.

In this way, a grounded model is developed from the convergence of data and domain theory.

4 Challenges for Designing Scientific Knowledge Discovery Tools

A number of specific research challenges are discussed here to foster awareness for data mining approach designers:

- There is an emerging interest in model construction in the scientific knowledge discovery and Knowledge Discovery in Databases (KDD) area [56], [57]. Identifying patterns from data, selecting appropriate models, calibrating and optimizing models have consumed considerable research effort. Scientists use various data analysis and optimization tools and rely on their experience to produce solutions. There are tools which address some aspects of these activities, for example tools for principal component analysis and algorithms for optimizations. Scientists do not yet have available comprehensive platforms aiming at addressing the knowledge discovery process from a holistic point of view. Developments in scientific work-flow processes can be used to model the scientific discovery process. These approaches apply business work-flow techniques (such as control flow modeling and large-scale collaboration) to the science domain to allow scientists to share, analyze, and synthesize data [58], [59]. These approaches have not yet attempted to address the sophistication of scientific model construction. Lack of adaptivity becomes a bottleneck for these tools to overcome. Many models may provide alternative results under different conditions. Without a model of what models (and algorithms) should be applied to what problems and how they should be used, an individual's effort is constrained by their expertise. The suitability of various models and their conditional performances should be captured and should be reusable later. How to learn and use this type of knowledge both effectively and efficiently remains a research question for model construction. Investigating approaches for inclusion of personal discovery experience and the related impacts on science outcome may lead to experience-based scientific discovery-support tools that learn [60];
- Another research issue associated with model construction is that many models may generate quite heterogenous results, how to assist scientists to identify an appropriate model and improve its performance is still a research question. Scientists' causal reasoning processes play important roles in this issue. Foundations for explanatory hypothesis generation have been laid by some researchers,

for example Simon [61], Langley et al. [62], [63] and Thagard [64]–[67]. Data mining researchers may draw upon these ideas to develop computer-aided scientific discovery tools.

- Observation provides the means for scientists to understand previous unexplainable phenomena. People learn cause–effect relationships based on their ability in coordinating data-driven inductive and knowledge-driven deductive processes. Data-driven induction has been the major approach for KDD in recent years. Machine learning research has not gained much from human learning theories although there exists an expected synergical effect between them [68]. The functionalities of many KDD programs have not targeted assisting scientists in sophisticated reasoning and other intuitive tasks like analogy, mental imagery, hypothesis building, and explanation. As we discussed in previous sections, KDD research works have been clustered around quantitative methods. Research ideas drawn from cognitive processes involved in human knowledge construction are not well known in the KDD community. To assist scientists' interpretation and evaluation, one needs to understand cognitive processes involved in scientific knowledge discovery. He or she also needs to determine how to build computational models to facilitate these processes?
- Large volumes of domain knowledge are available but they are not structured in a standardized and interoperable way. For example, there has been much effort in building domain-specific ontologies. Comparably fewer resources have been allocated to tackle the interoperability of multi-disciplinary (and inter-disciplinary) knowledge. How to capture and represent this domain knowledge in a cohesive way provides a research topic for knowledge engineers and KDD researchers. How to evolve knowledge structures over time and enhance their effectiveness is another challenge.
- A scientific community provides the means to judge new theories. As mentioned in earlier sections, a science paradigm tends to subordinate a discovery. Creativity becomes a concern when a genuine innovation is suppressed in this environment. Developing frameworks and methodologies to improve scientific cooperation and communication and to ensure unbiased scientific judgments may be a viable route to a scientific revolution.

In summary, scientific knowledge discovery is an interactive process in which scientists construct their first-person and then third-person knowledge. To support this dynamic process, a computer-based scientific knowledge discovery support system not only should facilitate the first-person knowledge construction but also needs to address the transition from first-person knowledge to third-person knowledge. A system could be built on a constructivist view that accommodates both a priori (existing domain knowledge) and a posteriori (learning) approaches. A system should:

- Have a model of an individual's experience that contains personalized domain knowledge, applications and operational knowledge of how to apply the knowledge to various problems

- Provide approaches for data-driven induction that can learn Patterns and predictions from data
- Provide approaches for model-driven deductive reasoning that can assist scientists' interpretation and explanation of data patterns
- Have mechanisms to coordinate data-driven induction and expectation-driven deduction to evaluate hypotheses
- Have mechanisms to learn from each discovery activity
- Consider multiple interpretations and their impacts on the adoption of the new theory and
- Facilitate the formation of a common ground in the community.

5 Conclusion

In this chapter, scientific knowledge discovery was discussed from a constructivist view. We reviewed cognitive theories about human knowledge construction and related them to the scientific knowledge discovery process. A hydrologic modeling scenario has been presented to exemplify our view. We argue that there is a need to build scientific discovery support tools based on constructive principles. Challenges for designing such a tool have been identified as:

- Including discovery experience and providing adaptive scientific work-flow platforms that enable scientists to construct and optimize models
- Providing means for assisting scientific explorative activities such as hypothesis building and testing
- Understanding cognitive processes involved in scientific model construction and building computational models to facilitate these processes
- Producing interoperable multi-disciplinary and inter-disciplinary domain ontologies and addressing the evolution of these knowledge structures; and
- Facilitating communication and cooperation in scientific communities to enhance creativity.

References

1. P. Langley, Int. J. Hum. Comput. Stud. **53**, 393–410 (2000)
2. S. Muggleton, Commun. ACM **42**(11), 42–46 (1999)
3. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, AI Magazine **17**, 37–54 (1996)
4. J.N. Sanchez, P. Langley, An Interactive Environment for Scientific Model Construction. in: *Proceedings of the Second International Conference on Knowledge Capture*, ACM Press, Sanibel Island, FL, 2003, pp. 138–145
5. W. Bridewell, J.N. Sanchez, P. Langley, D. Billman, Int. J. Hum. Comput. Stud. **64**(11), 1099–1114 (2006)
6. G.F. Luger, P. Johnson, C. Stern, J.E Newman, R. Yeo, Cognitive Science: The Science of Intelligent Systems (Academic Press, New York, 1994)

7. J. Locke, An Essay Concerning Human Understanding (Collins, London, 1690) reprinted in 1964
8. D. Hume, An Enquiry Concerning Human Understanding, e-book. http://ebooks.adelaide.edu.au/h/hume/david/h92e/chapter7.html
9. D. Hume, A Treatise of Human Nature, e-book. http://ebooks.adelaide.edu.au/h/hume/david/h92t/B1.3.6.html
10. I Kant, *Critique of Pure Reason*, Trans. by N.K. Smith (Macmillan Education Limited, London, 1781) reprinted in 1956
11. D.L. Medin, E.E. Smith, Ann. Rev. Psychol. **35**, 113–138 (1984)
12. B. Van Oers, Cognit. Sci. Q. 1, 279–305 (2001)
13. W. Clancey, Quarterly 1, 389–421 (2001)
14. D.H. Fisher, M. Pizzani, in *Computational Models of Concept Learning*, ed. by D.H. Fisher, M. Pazzani, P. Langley. Concept Formation: Knowledge and Experience in Unsupervised Learning (Morgan Kaufmann, San Mateo, CA, 1991), pp. 3–43
15. P.R. Bisbey, G.P. Trajkovski, *Rethinking Concept Formation for Cognitive Agents* (Towson University, 2005)
16. L.S. Vygotsky, *Thought and Language* (MIT, Cambridge, MA, 1986)
17. J. Dewey, *Experience and Nature* (Dover Publications, New York, 1929) reprinted in 1958
18. E. Von Glasersfeld, Revue Internationale de Philosophie **36**(4), 612–635 (1982)
19. J. Piaget, *The Construction of Reality in the Child*, Trans. by M. Cook (Basic Books, New York, 1954)
20. J. Piaget, *Biologie et connaissance* (Gallimard, Paris, 1967)
21. L.S. Vygotsky, *Mind in Society: The Development of Higher Psychological Processes* (Harvard University Press, Cambridge, MA, 1934) reprinted in 1978
22. W. Clancey, A Tutorial on Situated Learning, ed. by J. Self. in *Proceedings of the International Conference on Computers and Education*, AACE, Charlottesville, VA, 1995, pp. 49–70
23. D.A Kolb, *Experiential Learning: Experience as the Source of Learning and Development* (Prentice Hall, Englewood Cliffs, NJ, 1984)
24. J. Dewey, Experience and Education (Kappa Delta Pi, Indiana, 1938) reprint in 1998
25. K. Lewin, *Resolving Social Conflicts: Selected Papers on Group Dynamics* (Harper and Row, New York, 1948)
26. D.A. Kolb, R. Fry, in *Toward an Applied Theory of Experiential Learning* ed. by C. Cooper. Theories of Group Process (Wiley, London 1975)
27. P. Jarvis *Adult Learning in the Social Context* (Croom Helm, London 1987)
28. P. Jarvis, *Towards a Comprehensive Theory of Human Learning: Lifelong Learning and the Learning Society*, vol. I (Routledge, London, 2006)
29. P. Jarvis, in *Towards a Philosophy of Human Learning: An Existentialist Perspective*, ed. by P. Jarvis, S. Parker. Human Learning: An Holistic Approach (Routledge, London, 2005), pp. 1–15
30. E. Von Glasersfeld, in *Anticipation in the Constructivist Theory of Cognition* ed. by D.M. Dubois. Computing Anticipatory Systems (American Institute of Physics, Woodbury, NY, 1998), pp. 38–47
31. J. Hawkings, S. Blakeslee, *On Intelligence: How a New Understanding of the Brain will Lead to the Creation of Truly Intelligent Machines* (An Owl Book, Henry Holt and Company, New York, 2004)
32. Y. Dudai, M. Carruthers, Nature **434**, 567 (2005)
33. A. Riegler, Kybernetes **34**(1/2), 89–104 (2005)
34. W. Clancey, *Situated Cognition: On Human Knowledge and Computer Representations* (Cambridge University Press, Cambridge, 1997)
35. F.C Bartlett, *Remembering: A Study in Experimental and Social Psychology* (Cambridge University Press, Cambridge, 1932) reprinted in 1977
36. H. Von Foerster, in *Thoughts and Notes on Cognition* Ed. By H. Von Foerster, Understanding Understanding, (Springer, New York, 1970), pp. 169–190, reprinted in 2003

37. A. Riegler, in *Memory Ain't No Fridge: A Constructivist Interpretation of Constructive Memory*, ed. by B. Kokinov, W. Hirst. Constructive Memory, (NBU Series in Cognitive Science, Sofia, 2003), pp. 277–289
38. S. Harnad, Physica D **42**, 335–346 (1990)
39. D.J. Chalmers, in *Subsymbolic Computation and the Chinese Room*, ed. by J. Dinsmore. The Symbolic and Connectionist Paradigms: Closing the Gap (Lawrence Erlbaum, Hillsdale, NJ, 1992), pp. 25–48
40. G. Dorffner, E. Prem, Connectionism, Symbol Grounding, and Autonomous Agents. in *Proceedings of the 15th Annual Meeting of the Cognitive Science Society*, Lawrence Erlbaum, Hillsdale, NJ, 1993, pp. 144–148
41. M.T. Rosenstein, P.R. Cohen, Concepts from Time Series. in *Proceedings of 15th National Conference on Artificial Intelligence*, AAAI Press, Menlo Park, CA, 1998, pp. 739–745
42. G. Smith, J.S. Gero, The Autonomous, Rational Design Agent. in *Workshop on Situatedness in Design, Artificial Intelligence in Design'00* (Worcester, MA 2000), pp. 19–23
43. J.S. Gero, H. Fujii, Knowl. Based Syst. **13**(6), 361–368 (2000)
44. T.S. Kuhn, *The Structure of Scientific Revolutions* (The University of Chicago Press, Chicago, 1962) reprinted in 1996
45. S.A. Margulis, D. McLaughlin, D. Entekhabi, S. Dunne, Water Res. Res. **38**(12), 1299–1316 (2002)
46. CRC Catchment Hydrology Toolkit. http://www.toolkit.net.au/pdfs/MC-1.pdf
47. C. Michel, Water Resource Res. **35**, 3573 (1999)
48. A. Baird in *Soil and Hillslope Hydrology*, ed. by J. Wainwright, M. Mulligan. Environmental Modeling: Finding Simplicity in Complexity (Wiley, London, 2004), pp. 93–106
49. T.S. Steenhuis, J.Y. Parlange, W.E. Sanford, A. Heilig, F. Stagnitti, M.F. Walter, Water Resource Res. 35, 3575–3576 (1999)
50. L.A. Richards, Physics **1**, 318–333 (1931)
51. A.M. Binley, K.J. Beven, J. Elgy, Water Resources Res. **25**(6), 1227–1233 (1989)
52. S. Bitterlich, W. Durner, S.C. Iden, P. Knabner, Vadose Zone J. **3**, 971–981 (2004)
53. J. Simunek, R. Angulo-Jaramillo, M.G. Schaap, J.P. Vandervaere, M.T. Van Genuchten, Geoderma **86**, 61–81 (1998)
54. P.C. Young, A. Chotai, K.J. Beven, in *Data-based Mechanistic Modeling and the Simplification of Environmental Systems*, ed. by J. Wainwright, M. Mulligan. Environmental Modeling: Finding Simplicity in Complexity (Wiley, London, 2004), pp. 371–388
55. P.C. Young, in *Data-based Mechanistic Modeling and Validation of Rainfall-flow Processes*, ed. by M.G. Anderson, P.D. Bates. Model Validation: Perspectives in Hydrological Science (Wiley, Chichester, 2001), pp. 117–161
56. Langley, P.: Lessons for the Computational Discovery of Scientific Knowledge. in: *Proceedings of the 1st International Workshop on Data Mining Lessons Learned*, Sydney, 2002, pp. 9–12
57. P. Domingos, Data Min. Knowl. Disc. **15**, 21–28 (2007)
58. A.M. Ellison, L.J. Osterweil, J.L. Hadley, A. Wise, E. Boose, L.A. Clarke, D.R. Foster, A. Hanson, D. Jensen, P. Kuzeja, E. Riseman, H. Schultz. Ecology **87**(6), 1345–1358 (2006)
59. L.J. Osterweil, A. Wise, L. Clarke, A.M. Ellison, J.L. Hadley, E. Boose, R. David, R. Foster, *Process Technology to Facilitate the Conduct of Science*, in Software Process Workshop (SPW2005). LNCS, vol. 3840, Springer, Beijing, 2005, pp. 403–415
60. J.S. Gero, in *Design Tools that Learn: A Possible CAD Future*, ed. by B. Kumar. Information Processing in Civil and Structural Design (Civil-Comp Press, Edinburgh, 1996), pp. 17–22
61. D. Kulkarni, H.A. Simon, Cognit. Sci. **12**, 139–175 (1988)
62. P. Langley, O. Shiran, J. Shrager, L. Todorovski, A. Pohorille, AI Medicine **37**, 191–201 (2006)
63. P. Langley, W. Bridewell, Processes and Constraints in Explanatory Scientific Discovery. in *Proceedings of the 13th Annual Meeting of the Cognitive Science Society*, Washingon, DC, 2008, in press
64. P. Thagard, Behav. Brain Sci. **12**, 435–467 (1989)
65. P. Thagard, Cognit. Sci. Q. **1**, 93–116 (2000)
66. P. Thagard, Appl. Artif. Intell. **18**(3), 231–249 (2004)

67. P. Thagard, A. Litt, in *Models of Scientific Explanation*, ed. by R. Sun. The Cambridge Handbook of Computational Psychology (Cambridge University Press, Cambridge, 2008), pp. 549–564
68. T.M. Mitchell, The Discipline of Machine Learning, Machine Learning Department technical report CMU-ML-06-108, Carnegie Mellon University, 2006

“This page left intentionally blank.”

Knowledge Representation and Ontologies

Stephan Grimm

1 Knowledge Representation

Knowledge representation and reasoning aims at designing computer systems that reason about a machine-interpretable representation of the world. Knowledge-based systems have a computational model of some domain of interest in which symbols serve as surrogates for real world domain artefacts, such as physical objects, events, relationships, etc. [1]. The *domain* of interest can cover any part of the real world or any hypothetical system about which one desires to represent knowledge for computational purposes.

A knowledge-based system maintains a *knowledge base*, which stores the symbols of the computational model in the form of statements about the domain, and it performs reasoning by manipulating these symbols. Applications can base their decisions on answers to domain-relevant questions posed to a knowledge base.

1.1 Principles of Representing Knowledge

Various forms of knowledge representation have been proposed, which affect the flavor of interaction of a knowledge-based system with its knowledge base for reasoning about domain knowledge.

Forms of Representing Knowledge

The most prevalent forms of knowledge representation appearing in computer systems are those of semantic networks, rules and logic. While semantic networks use

S. Grimm (✉)
FZI Research Center for Information Technologies, University of Karlsruhe, Germany
e-mail: grimm@fzi.de

M.M. Gaber (ed.), *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*, DOI 10.1007/978-3-642-02788-8_6,

the metaphor of a graph to visualize conceptual structures, rules exhibit some if-then-reading to express statements about a domain. Logic is used to implement a precise formal semantics for both semantic networks and rules.

Semantic Networks

Originally, semantic networks stem from the "existential graphs" introduced by Charles Peirce in 1896 to express logical sentences as graphical node-and-link diagrams [2]. Later on, similar notations have been introduced, such as conceptual graphs [1], all differing slightly in syntax and semantics.

Figure 1 shows an example of a semantic network that captures some knowledge about the domain of academia. The nodes in the network represent either classes of things, such as *Professor*, or individual entities, such as *John*, which are referred to as *concepts* and *individuals*, respectively. The arcs between the nodes interrelate concepts and individuals, and are referred to as *relations*. The network states, for example, that "*professors lecture courses attended by students,*" or that "*John is a professor who lectures the course CalculusI attended by the undergraduate student Ben*".

Semantic networks are especially suitable for capturing the taxonomic structure of concept hierarchies and for expressing general statements about the domain of interest. Inheritance (*kindOf*) and other relations between such categories can be represented in and derived from subsumption hierarchies. On the other hand, the representation of large amounts of concrete individuals or even data values, like numbers or strings, does not fit well the idea of semantic networks. Modern descendants of semantic networks are description logics, which give a rigour formal semantics to the network constructs.

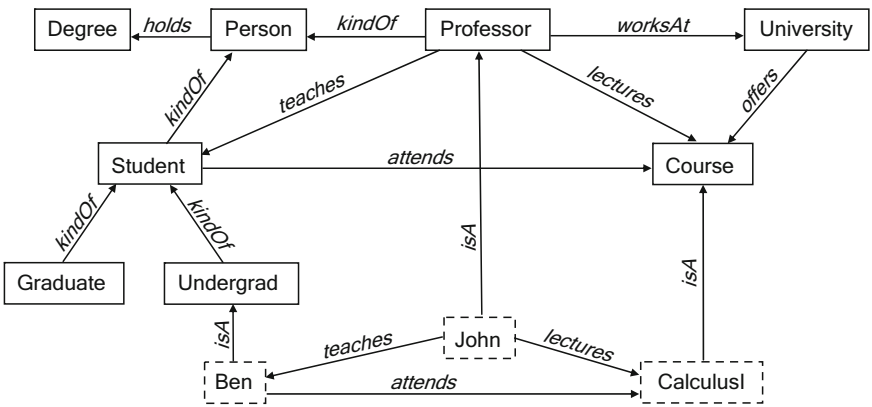


Fig. 1 A semantic network for academia

Rules

Another natural form of expressing knowledge in some domain of interest are *rules* that reflect the notion of consequence. Rules come in the form of if-then-constructs and allow to express complex derivation statements. Rules can be found in logic programming systems, like the language Prolog [3], in deductive databases [4] or in business rules systems.

An example for a set of rules with an if-then-reading, also capturing knowledge about academia, is the following:

1. **IF** *somebody is a graduate student* **THEN** *he is also a student*
2. **IF** *some professor lectures a course attended by a particular student*
   THEN *the professor teaches this student*
3. **FACT** *John is a professor who lectures a course that the graduate student Ben attends*

The basic principle behind rules is that of *derivation*: knowledge is expressed in terms of what is derived from a particular situation of facts. For example, starting from the fact that "*John, the professor, lectures a course attended by Ben, the graduate student*" (3), it can be derived by the rules (1) and (2) that "*John teaches Ben*". Such derived information can then, in turn, trigger other rules, further stimulating the derivation process.

The rule-based form of knowledge representation is particularly suitable for reasoning about concrete instance data, i.e., simple ground facts of the form "*John is a Professor*". Complex sets of rules can efficiently derive implicit facts from explicitly given ones. They are problematic, however, if more complex and general statements about the domain shall be derived which do not fit a rule's shape. Rule-based approaches are implemented in deductive databases and rule engines, and are given a precise formal semantics by means of logic programming formalisms.

Logic

Both forms, semantic networks as well as rules, have been formalised using logic to give them a precise semantics. Without such a precise formalisation they are vague and ambiguous, and thus problematic for computational purposes. From just the graphical representation of the fragment $\boxed{\textit{Professor}} \xrightarrow{\textit{lectures}} \boxed{\textit{Course}}$ in the semantic network shown in Figure 1, for example, it is not clear whether every professor lectures a course or only some do. Also for rules, despite their much more formal appearance, the exact meaning remains unclear, for example, when forms of negation are introduced that allow for potential conflicts between rules. Depending on the choice of procedural evaluation or flavour of formal semantics, different derivation results are being produced.

The most prominent and fundamental logical formalism classically used for knowledge representation is the "first-order predicate calculus," or *first-order logic* for short, and we choose this formalism to present logic as a form of knowledge representation here. First-order logic allows one to describe the domain of interest

as consisting of objects, i.e., things that have individual identity, and to construct logical formulas around these objects formed by predicates, functions, variables and logical connectives [2].

Similar to semantic networks, many natural language statements can be expressed in terms of logical sentences about objects of the domain of interest with an appropriate choice of predicate and function symbols. Concepts are mapped to unary, relations to binary predicates. We illustrate the use of logic for knowledge representation by axiomatising the above fragment of our example semantic network more precisely, as follows.

$$\boxed{\textit{Professor}} \xrightarrow{\textit{lectures}} \boxed{\textit{Course}} \quad \begin{array}{l} \forall x, y : (\textit{lectures}(x, y) \rightarrow \textit{Professor}(x) \wedge \textit{Course}(y)) \\ \forall x : \exists y : (\textit{Professor}(x) \rightarrow \textit{Course}(y) \wedge \textit{lectures}(x, y)) \end{array}$$

The first formula states that "*the lectures-relation holds between professors and courses*", while the second formula assures that "*every professor does actually lecture some course*".

Rules can also be formalised with logic. An **IF**–**THEN**–rule can be represented as a logical implication with universally quantified variables. The typical formalisation of a rule is illustrated in the following example.

IF *some professor lectures a course attended by a particular student*
THEN *the professor teaches this student*

$$\forall x, y, z : (\textit{Professor}(x) \wedge \textit{Course}(y) \wedge \textit{Student}(z) \wedge \textit{lectures}(x, y) \wedge \textit{attends}(z, y) \rightarrow \textit{teaches}(x, z)).$$

The *body* of the rule is represented by the premises of a logical implication and captures a situation of "*a course lectured by a professor and attended by a student*", while the *head* of the rule is represented by the consequence of the implication and derives that "*the professor teaches the student*". By universal quantification of all variables involved, the rule expresses general knowledge and is applied to any individual situation that matches the body.

Reasoning about Knowledge

The way in which we, as humans, process knowledge is by reasoning, i.e., the process of reaching conclusions. Analogously, a computer processes the knowledge stored in a knowledge base by drawing conclusions from it, i.e., by deriving new statements that follow from the given ones.

The basic operations a knowledge-based system can perform on its knowledge base are typically denoted by `tell` and `ask` [2]. The `tell`-operation adds a new statement to the knowledge base, whereas the `ask`-operation is used to query what is known. The statements that have been added to a knowledge base via the `tell`-operation constitute the *explicit knowledge* a system has about the domain of interest. The ability to process explicit knowledge computationally allows a knowledge-based system to reason over the domain by deriving *implicit* knowledge that follows from what has been told explicitly.

This leads to the notion of logical consequence or *entailment*. A knowledge base KB is said to entail a statement α if α "follows" from the knowledge stored in KB, which is written as $KB \models \alpha$. A knowledge base entails all the statements that have been added via the `tell`-operation plus those that are their logical consequences. As an example, consider the following knowledge base with sentences in first-order logic.

$$
\begin{aligned}
KB = \{\ & Professor(John),\ \forall x : lectures(John, x) \rightarrow MathCourse(x), \\
& Undergraduate(Ben),\ \forall x : MathCourse(x) \rightarrow attends(Ben, x), \\
& \forall x \exists y : Professor(x) \rightarrow Course(y) \wedge lectures(x, y), \\
& \forall x, y : lectures(x, y) \rightarrow Person(x) \wedge Course(y) \wedge holds(x, MasterDegree), \\
& \forall x, y, z : Professor(x) \wedge lectures(x, y) \wedge attends(z, y) \wedge Student(z) \\
& \quad \rightarrow teaches(x, z), \\
& \forall x : Undergraduate(x) \rightarrow \neg holds(x, MasterDegree) \qquad \}
\end{aligned}
$$

The knowledge base KB explicitly states that "*John is a professor who lectures only math courses*", that "*Ben is a graduate student who attends all math courses,*" that "*each professor lectures some course,*" that "*lecturing requires at least a master degree*", that "*professors teach those students who attend the courses they lecture*" and that "*undergraduate students do not have master degrees.*" If we ask the question "*Does John teach Ben?*" by saying

$$\texttt{ask}(KB, teaches(John, Ben))$$

the answer will be yes. The knowledge base KB entails the fact that "*John teaches Ben,*" i.e., $KB \models teaches(John, Ben)$, although it was not "told" so explicitly. Since Ben attends all math courses, some of which is clearly lectured by John, Ben attends a course that John lectures, and thus, John teaches Ben. A more general consequence of KB is that "*no undergraduate student lectures a course,*" i.e., $KB \models \neg\exists x, y : Undergraduate(x) \wedge Course(y) \wedge lectures(x, y)$, which is reflected by a positive answer to the question

$$\texttt{ask}(KB, \neg\exists x, y : Undergraduate(x) \wedge Course(y) \wedge lectures(x, y)).$$

This follows since "*lecturing requires a master degree*" but "*undergraduate students do not have one.*"

Another important notion related to entailment is that of consistency or *satisfiability*. Intuitively, a knowledge base is consistent or satisfiable if it does not contain contradictory facts. If we would add the information that "*Ben teaches some tutorial course*" to KB by saying

$$\texttt{tell}(KB, Course(Tutorial) \wedge lectures(Ben, Tutorial)),$$

it would become unsatisfiable because this would conflict with the formerly derived consequence about undergraduate students not lecturing courses. In general, an unsatisfiable knowledge base is not very useful, since in logical formalisms it would entail any arbitrary statement. The `ask`-operation would always return a positive result independent from its input, which is clearly not desirable for a knowledge-based system.

The inference procedures implemented in computational reasoners aim at realising the entailment relation between logical statements [2]. They derive implicit statements from a given knowledge base or check whether a knowledge base is satisfiable.

An inference procedure that only derives entailed statements is called *sound*. Soundness is a desirable feature of an inference procedure, since an unsound inference procedure would potentially draw wrong conclusions. If an inference procedure is able to derive every statement that is entailed by a knowledge base then it is called *complete*. Completeness is also a desirable property, since a complex chain of conclusions might break down if only a single statement in it is missing. Hence, for reasoning in knowledge-based systems we desire sound and complete inference procedures.

1.2 Logical Knowledge Representation Formalisms

Logical formalisms are the theoretical underpinning of symbolic knowledge representation as laid out above. Besides first-order logic with its classical model-theoretic semantics, other formalisms like description logics or logic programming have evolved in Artificial Intelligence research.

Classical Model-Theoretic Semantics

First-order logic is the prevalent and single most important knowledge representation formalism. Its importance stems from the fact that basically all current symbolic knowledge representation formalisms can be understood in their relation to first-order logic. First-order logic captures some of the essence of human reasoning by providing a notion of logical consequence that is realised in terms of a model-theoretic semantics.

A knowledge base $\mathcal{KB}$ is viewed as a set of first-order formulas that constitute a logical theory T. In model theory, the semantics of T is defined in terms of *interpretations*. Formally, an interpretation $\mathcal{I}$ is a mapping from the elements in the formulas in T into a set $\Delta^{\mathcal{I}}$, called the *interpretation domain*. Constant symbols, for example, directly map to objects in the interpretation domain, while predicate symbols are interpreted as subsets of or relations over $\Delta^{\mathcal{I}}$, which are called their *extensions*. Intuitively, an interpretation specifies a particular arrangement of objects in the interpretation domain in terms of membership in predicate

extensions. For example, in one interpretation a constant symbol like *John* can be in the extension of the predicate *Professor*, in another interpretation it can be in the extension of the concept *Student*, and in yet another one it can be in the extensions of both. If the arrangement of objects in an interpretation $\mathcal{I}$ is in accordance with the formulas in T, then $\mathcal{I}$ is called a *model* of T. For example, if we state that John is a professor and that professors are disjoint from students in a theory $T = \{Professor(John), \forall x : Professor(x) \rightarrow \neg Student(x)\}$, then interpretations in which *John* is in the extension of the predicate *Student* cannot be models of T as they do not satisfy its formulas.

Even after filtering out those interpretations that do not satisfy the formulas in a logical theory T, it has in general a multitude of models in which things are interpreted differently. For example, if we do not say whether John lectures a math course or not then there are models in which he does and such in which he does not. In this case, we say that we have *incomplete knowledge* about John's lecturing, which is captured by the situation of multiple models. Contrarily, if we explicitly list all professors together with the courses they lecture and also state that there are no other lecturing relations then we say that we have complete knowledge about lecturing, which is reflected by T having only models in which the courses lectured by professors are exactly those from our explicit list.

Reasoning about a theory T is now defined based on its models. The reasoning task of *validation*, concerned with the theory's consistency, is to check whether T has a model. The reasoning task of *deduction*, concerned with entailment of logical consequences, tests whether statements to be derived are true in *all* models of T.

Description Logics

Description Logics (DLs) [5] are a family of class (concept)-based knowledge representation formalisms with well-studied representational and computational properties. They are the modern descendants of early knowledge representation systems such as KL-One [6] or CLASSIC [7] and were developed to give a precise formalisation to semantic networks. As such, they typically form decidable fragments of the first-order logic predicate calculus restricted to unary and binary predicates to capture the nodes and arcs in a network graph.

The basic elements used to represent knowledge in the description logic formalism are *concepts*, *individuals*, and *roles*. Intuitively, concepts denote classes of things, such as *Student* or *Course*. Individuals denote instances, such as *Ben* or *CalculusI*. Roles denote relationships between things, such as *attends*. Moreover, DLs are often augmented by so called *concrete domains* [8] to also capture datatypes such as integer or string.

DLs are characterized by an intensional description of concepts using concept constructors to build complex concepts out of simple ones. Starting from a set of concept names, a set of role names, and a set of individual names, complex concept expressions can be formed using such constructors in a nested way. The choice

of constructors determines the particular description logic named according to a scheme of letters indicating specific language constructs, which is shown in Table 1.

The basic DL $\mathcal{AL}$, which stands for Attributive Language, is often referred to as a minimal language of practical interest [5], forming the basis to build more complex DLs on top. $\mathcal{AL}$ provides the top ($\top$) and bottom ($\bot$) concepts, concept conjunction ($C \sqcap D$), negation of atomic concepts ($\neg A$), universal restriction ($\forall\, r.C$) and limited existential restriction ($\exists\, r.\top$). Another important description logic is $\mathcal{ALC}$, which augments $\mathcal{AL}$ by general concept negation ($\neg C$) and unlimited existential restriction ($\exists\, r.C$), and is thus the most basic DL closed under Boolean operators. The upper part of Table 2 shows some examples of DL concept expressions with their intuitive meaning.

Table 1 Various constructors to form different description logics

Name	Symbol	Syntax	Description
Top concept	$\mathcal{AL}$	$\top$	Class of all objects
Bottom concept		$\bot$	Empty class
Atomic concept negation		$\neg A$	Complement of named class
Concept conjunction		$C \sqcap D$	Intersection of classes
Universal restriction		$\forall\, r.C$	Objects whose values for r all are in C
Limited existential restr.		$\exists r$	Objects with some value for r
Concept disjunction	$\mathcal{U}$	$C \sqcup D$	Union of classes
Existential restriction	$\mathcal{E}$	$\exists\, r.C$	Objects with some value for r in C
Concept negation	$\mathcal{C}$	$\neg C$	Complement of complex class
Min cardinality restriction	$\mathcal{N}$	$\geq n\ r$	Objects with at least n values for r
Max cardinality restriction		$\leq n\ r$	Objects with at most n values for r
Qualified min card. restr.	$\mathcal{Q}$	$\geq n\ r.C$	Objects w. at least n values for r in C
Qualified max card. restr.		$\leq n\ r.C$	Objects w. at most n values for r in C
Nominal	$\mathcal{O}$	$\{o\}$	Singleton class with individual o
Inverse role	$\mathcal{I}$	r^-	Relation with inverse direction
	$\mathcal{S}$		$\mathcal{ALC}$ with transitive roles
Role inclusion	$\mathcal{H}$	$r \sqsubseteq s$	Role hierarchies
Concrete domains	(**D**)		Datatype properties and predicates

Table 2 Examples of DL language constructs

DL concept descriptions	
Student $\sqcap\, \exists$ attends.MathCourse	Students who attend some math course
Course $\sqcap\, \forall$ *lectures*$^-$.{*John*}	Courses that are only lectured by John
Professor $\sqcap \geq 5$ lectures.Course	Professors who lecture five or more courses
DL axioms	
BusyProfessor $\equiv$ Professor $\sqcap \geq 5$ lectures.Course	Busy professors are just those who lecture five or more courses
Undergraduate $\sqsubseteq \neg\exists$ holds.{*MasterDegree*}	Undergraduates do not hold a master degree
$\forall$ lectures.MathCourse(*John*)	John lectures only math courses
teaches(*John*, *Ben*)	John teaches Ben

A DL knowledge base contains statements about the domain of interest in form of DL axioms and is composed of a T-Box and an A-Box. The T-Box captures so called terminological knowledge, which comprises general statements within a domain, and T-Box axioms are concept inclusions of the form $C \sqsubseteq D$ or concept equivalences of the form $C \equiv D$. The A-Box, on the other hand, captures so called assertional knowledge, which comprises statements about particular individuals and situations. A-Box axioms comprise concept assertions of the form $C(a)$ or role assertions of the form $r(a, b)$. Examples of DL axioms are given in the lower part of Table 2.

The semantics of description logics is defined in the classical model-theoretic way and directly builds on the semantics of first-order logic. A DL knowledge base is interpreted as the first-order logical theory that results from a respective transformation of its axioms. We refer to [5] for a thorough treatment of description logics.

Besides entailment of subsumption and assertion axioms, which underly T-Box classification and instance retrieval, typical DL reasoning tasks are *knowledge base satisfiability* to check the consistency of a knowledge base, and *concept satisfiability* to test if a concept can potentially have instances. For example, if in Table 2 professors were also restricted to lecture at most four courses then the concept *BusyProfessor* would be unsatisfiable.

Logic Programming

Logic programming (LP) was originally conceived as a way to use first-order logic as a programming language, e.g., in case of the language Prolog [3]. To ensure computability of the language, statements are syntactically restricted to so-called Horn clauses and only certain kinds of logical consequences are being considered.

Syntactically, Horn clauses can be understood as rules. For example, the expression $\mathit{Student}(x) \vee \neg \mathit{Undergrad}(x)$ is a Horn clause, which is semantically equivalent to $\forall x : \mathit{Student}(x) \leftarrow \mathit{Undergrad}(x)$. This can also be interpreted as the rule (1) from page 113 written in the typical Prolog-style notation `Student(?`x`) :- Undergrad(?`x`)`. However, the semantics of the Horn clause is given by means of first-order logic semantics, whereas logic programming rules are usually understood in a different, non-classical sense. One of the differences stems from the fact that in a logic programming system only certain types of logical consequences are being considered, namely ground instances of predicates.[1] Applying the above rule to a fact `Undergrad(Ben)` would allow to conclude `Student(Ben)` both in first-order logic and in logic programming. A conclusion such as `Graduate(Ben)` $\vee$ `Undergrad(Ben)`, however, would only be possible in first-order logic, and not derivable in a LP system.

Moreover, negative information is handled differently within the LP paradigm than it is in classical logic. In the example above we could be interested in whether

[1] A ground (instance of a) predicate is an atomic formula which does not contain any variable symbols.

the statement Student(Susan) holds. In classical logics, neither truth nor falsity of this statement is derivable. In logic programming, however, the statement would be considered false, as LP systems typically treat the lack of information as a form of default negation called *negation-as-failure*. Since no information on Susan is available, she is considered to be *not* a student. The semantics of LP formalisms is based on *minimal models*, i.e., those models of a knowledge base that have the least extensions of predicates, while other models are not considered for reasoning. Due to the lack of evidence for Susan being a student no model in which Student(Susan) holds is minimal.

In the LP paradigm, a knowledge base is understood as a *logic program*, which is a set of rules H :− B where H is called the head and B is called the body of the rule. Facts are rules with an empty body, i.e., what is stated in their head holds regardless of any conditions. The various logic programming dialects allow different forms for the heads and the bodies of rules. The most basic LP variants restrict the head to atomic predicates, while they allow for conjunction ($\wedge$) and negation-as-failure in the body. Examples for such systems are Prolog [3] and Datalog [9], a language that has its origin in deductive databases. More advanced systems, such as disjunctive Datalog [10] or Answer Set Programming [11], allow for disjunctions of predicates in the head and provide a more sophisticated treatment of negation. Another feature common to most LP-systems are built-ins for the handling of strings and numbers within a logic program, similar to concrete domains in DLs. The basic reasoning task for logic programs is retrieval of instances that match a particular query pattern. For example, the query $\texttt{Student}(?x) \wedge \neg\texttt{holds}(?x, \texttt{MasterDegree})$ asks for all students that do not have a master degree.

We refer to [12] for a formal treatment of LP formalisms and for an overview on various styles of minimal model semantics, such as stable model or well-founded semantics.

1.3 Knowledge Representation Paradigms

Besides the various logical formalisms, different modeling paradigms have evolved that characterise the representation of knowledge.

Open-World vs. Closed-World View

An essential choice that has to be made when selecting a formalism for representing knowledge is how situations of incomplete information are handled. One option is to take an open-world view, assuming that at any time new knowledge can be learned that resolves previous ambiguity. Another option is to take a closed-world view, assuming to have full knowledge about a particular situation by what has been observed so far.

Open-World View

Knowledge representation based on logics with classical semantics operates under the *open-world assumption*, which allows for a distinction between negative knowledge and the lack of knowledge. An example of negative knowledge is an entry in the complete course enrolment plan for the student Ben and a particular course, having the form of a negated assertion $\neg enrolled(Ben, CalculusI)$. From this, the professor teaching the respective course can safely conclude that Ben is not enrolled. In an open-world view, this is different from the lack of an assertion about Ben's enrolment, from which a knowledge-based system would not draw any conclusion. The question as to whether Ben is enrolled would be answered with "unknown," since neither $\neg enrolled(Ben, CalculusI)$ nor $enrolled(Ben, CalculusI)$ could be derived.

Depending on the particular use case, an open-world view can either be beneficial or hindering. While in some situations it is preferable for the professor to quickly take the action of contacting all enrolled students concerning assignments, Ben might just not have completed his enrolment process yet, such that the indication of his enrollment being "unknown" can be a valuable information, worth to be distinguished from certain absence.

The open-world view in classical logics is technically realised by the connection of logical consequence to the underlying mechanism of the model-theoretic semantics. In a situation of incomplete information, unspecified issues can be resolved in different ways, each represented by a different model of the respective knowledge base. If there is no information about Ben's enrolment in a particular course, there are models in which he is enrolled and such in which he is not. Neither conclusion can be drawn, since conclusions need to hold in all models. The open-world view has been argued to be particularly suitable for knowledge-based applications used in the web due to its open and volatile nature.

Closed-World View

When making the *closed-world assumption*, negative knowledge coincides with the lack of knowledge, and what cannot be proven true is assumed to be false. Thus, a knowledge-based system that takes a closed-world view assumes to have complete information when reasoning about a particular situation. In consequence, such a system never produces "unknown" as an answer but always takes a decision as to whether a particular statement holds or does not hold. The lack of information about the enrolment of Ben in a particular course, for example, is simply interpreted as non-attendance, taking the form of $\neg enrolled(Ben, CalculusI)$.

Similar to the open-world view, it is sometimes beneficial and pragmatical and sometimes inadequate to take the closed-world view, depending on the actual use case. A closed-world perspective is particularly natural from a database point of view. A student is assumed to be not enrolled in a course unless an enrolment record can be found in the database.

The closed-world assumption is intrinsically made in logic programming and deductive database systems, where it is realised by the underlying minimal model semantics that takes only models with minimal predicate extensions into account for reasoning. In absence of the fact *enrolled*(*Ben*, *CalculusI*), only models in which Ben is not enrolled in the course CalculusI are minimal, such that him not being enrolled becomes a logical consequence due to negation-as-failure.

Clear-Cut Predication vs. Metamodeling

In knowledge representation based on first-order logic, there is an intrinsic distinction between an *intensional* part of a knowledge base, which captures general statements about classes of objects and their properties, and an *extensional* part, which captures statements about particular situations of individual objects in the interpretation domain. This distinction is due to the "first-order" nature of the formalism, which induces a clear separation between domain objects and the predicates used to express their properties.[2]

One particular paradigm for knowledge representation is to impose a clear-cut separation between the intensional and the extensional part of a knowledge base, such that no symbol serves as a property in the intensional part and as an individual object in the extensional part at the same time. Although this separation is intuitive, it is sometimes inadequate for certain domains. For example, if we want to state that John is a professor and that professor is one particular profession among others by *Professor*(*John*) and *Profession*(*Professor*) then we break the separation between intensional and extensional statements, since *Professor* is used both as a concrete object in the domain and as a property for another object *John* in form of a unary predicate.

This alternative paradigm of not imposing a clear-cut separation between intensional and extensional knowledge is also referred to as *metamodeling* [13]. In terms of semantic networks, metamodeling amounts to using concepts also in place of individuals.

Conceptual Modeling vs. Rules

The description logic and logic programming formalisms form the two major strands for research on knowledge representation in symbolic Artificial Intelligence, standing for antithetic paradigms. As we have seen in the previous sections, both represent different forms of knowledge representation that give the modeling of domain knowledge a specific flavour. While description logics build on conceptual modeling by means of intensional descriptions of concepts and their interrelation, logic programming provides the basic construct of a rule for the derivation of facts. Both formalisms can be seen as deviations from the first-order predicate calculus in

[2] In higher-order logics this separation is abrogated and variables can range over predicates.

two different directions: while description logics limit the arity and use of predicates to fit semantic network graphs under classical semantics, logic programming formalisms restrict the shape of statements that can be derived to ground facts under certain forms of minimal model semantics to yield efficient derivation rule systems. Hence, the features they exhibit as forms of knowledge representation are those of conceptual modeling versus rules.

The modeling of knowledge for knowledge-based systems in these two paradigms has different applications. Description logics are rather used in applications that require schema-intensive reasoning and classification of concepts according to their intensional descriptions, whereas logic programming systems are more used for data-intensive reasoning tasks and for the retrieval of instance data from large extensional knowledge bases. Related to this, the two types of formalisms also differ in their computational properties concerning efficiency. While schema-intensive reasoning is rather intricate and DL reasoning problems typically have exponential runtime complexity, the reduction of LP consequences to ground facts makes reasoning with logic programs more tractable.

Both these paradigms have different expressivity and allow for complementary features, and thus, it is desirable to combine the formalisms of DL and LP to yield a more powerful tool for knowledge representation. Despite their inherent semantic incompatibility, there have recently been attempts to a seamless integration of description logics and rules at a semantic level on formal grounds, as reported in [14, 15].

2 Ontologies

Ontologies are conceptual models that make the knowledge of a particular domain of interest available to computer systems. As such, they build on the notions of knowledge representation we have presented before.

2.1 Notion of an Ontology

Ontology

In its original meaning in philosophy, *ontology* is a branch of metaphysics and denotes the philosophical investigation of existence. It is concerned with the fundamental questions of “what is being?” and “what kinds of things are there?” [16]. Dating back to Aristotle, the question of “what exists?” lead to studying general categories for all things that exist. Ontological categories provide a means to classify all existing things, and the systematic organisation of such categories allows to analyse the world that is made up by these things in a structured way. In ontology, categories are also referred to as *universals*, and the concrete things that they serve to classify are referred to as *particulars*.

Philosophers have mostly been concerned with general top-level hierarchies of universals that cover the entire physical world. Examples of universals occurring in such top-level hierarchies are most general and abstract concepts like “substance,” “physical object,” “intangible object,” “endurant” or “perdurant.” Transferred to knowledge representation and computer science, information systems can benefit from the idea of ontological categorisation. When applied to a limited domain of interest in the scope of a concrete application scenario, ontology can be restricted to cover a special subset of the world. Examples of ontological categories in the academic domain are “University,” “Professor,” “Student” or “Course,” whereas examples for particular individuals that are classified by these categories are concrete universities, professors, students and courses.

In general, the choice of ontological categories and particular objects in some domain of interest determines the things about which knowledge can be represented in a computer system [1]. In this sense, ontology provides the labels for nodes and arcs in a semantic network or the names for predicates and constants in rules or logical formulas, that constitute an *ontological vocabulary*. By defining “what exists” it determines the things that can be predicated about.

Ontologies

While “ontology” studies what exists in a domain of interest, “an ontology” as a computational artefact encodes knowledge about this domain in a machine-processable form to make it available to information systems. In various application contexts, and within different communities, ontologies have been explored from different points of view, and there exist several definitions of what an ontology is. Within the Semantic Web community the dominating definition of *an ontology* is the following, based on [17].

Definition 1 (ontology). An *ontology* is a formal explicit specification of a shared conceptualisation of a domain of interest.

This definition captures several characteristics of an ontology as a specification of domain knowledge, namely the aspects of formality, explicitness, being shared, conceptuality and domain-specificity, which require some explanation.

- *Formality*
  An Ontology is expressed in a knowledge representation language that provides a formal semantics. This ensures that the specification of domain knowledge in an ontology is machine-processable and is being interpreted in a well-defined way. The techniques of knowledge representation help to realize this aspect.
- *Explicitness*
  An ontology states knowledge explicitly to make it accessible for machines. Notions that are not explicitly included in the ontology are not part of the

machine-interpretable conceptualisation it captures, although humans might take them for granted by common sense.[3]

- *Being shared*
  An ontology reflects an agreement on a domain conceptualisation among people in a community. The larger the community, the more difficult it is to come to an agreement on sharing the same conceptualisation. Thus, an ontology is always limited to a particular group of people in a community, and its construction is associated with a social process of reaching consensus.
- *Conceptuality*
  An ontology specifies knowledge in a conceptual way in terms of symbols that represent concepts and their relations. The concepts and relations in an ontology can be intuitively grasped by humans, as they correspond to the elements in our mental model. (In contrast to this, the weights in a neural network or the probability measures in a Bayesian network would not fit such a conceptual and symbolic approach.) Moreover, an ontology describes a conceptualisation in general terms and does not only capture a particular state of affairs. Instead of making statements about a specific situation involving particular individuals, an ontology tries to cover as many situations as possible that can potentially occur [18].
- *Domain specificity*
  The specifications in an ontology are limited to knowledge about a particular domain of interest. The narrower the scope of the domain for the ontology, the more an ontology engineer can focus on axiomatizing the details in this domain rather than covering a broad range of related topics. In this way, the explicit specification of domain knowledge can be modularised and expressed using several different ontologies with separate domains of interest.

Technically, the principal constituents of an ontology are *concepts*, *relations* and *instances*. Concepts map to the generic nodes in semantic networks, or to unary predicates in logic, or to concepts as in description logics. They represent the ontological categories that are relevant in the domain of interest. Relations map to arcs in semantic networks, or to binary predicates in logic, or to roles in description logics. They semantically connect concepts, as well as instances, specifying their interrelations. Instances map to individual nodes in semantic networks, or to constants in logic. They represent the named and identifiable concrete objects in the domain of interest, i.e., the particular individuals which are classified by concepts. Altogether, these elements constitute an ontological vocabulary for the respective domain of interest. An ontology can be viewed as a set of statements expressed in terms of this vocabulary, which are also referred to as *axioms*.

[3] Notice that this notion of explicitness is different from the distinction between explicit and implicit knowledge, introduced earlier. Implicit knowledge that can be derived by means of automated deduction does not need to be included in an ontology for a computer system to access it. However, knowledge that is neither explicitly stated nor logically follows from what is stated, can by no means be processed within the machine, although it might be obvious to a human. Such knowledge remains implicit in the modeler's mind and is not represented in the computer.

Conceptual modeling with ontologies seems to be very similar to modeling in object-oriented software development or to designing entity-relationship diagrams for database schemas. However, there is a subtle twofold difference. First, ontology languages usually provide a richer formal semantics than object-oriented or database-related formalisms. They support encoding of complex axiomatic information due to their logic-based notations. Hence, an ontology specifies a semantically rich axiomatization of domain knowledge rather than a mere data or object model. Second, ontologies are usually developed for a different purpose than object-oriented models or entity-relationship diagrams. While the latter mostly describe components of an information system to be executed on a machine or a schema for data storage, respectively, an ontology captures domain knowledge as such and allows to reason about it at runtime.

In summary, an ontology used in an information system is a conceptual yet executable model of an application domain. It is made machine-interpretable by means of knowledge representation techniques and can therefore be used by applications to base decisions on reasoning about domain knowledge.

2.2 Ontologies in Information Systems

In information systems, ontologies are used with different forms of appearance and in different contexts of usage, while also different types of ontologies have been coined.

Appearance of Ontologies

When engineered for or processed by information systems, ontologies appear in different forms related to the forms of knowledge representation that we have discussed previously. A knowledge engineer views an ontology by means of some graphical or formal visualization, while for storage or transfer it is encoded in an ontology language with some machine-processable serialisation format. A reasoner, in turn, interprets an ontology as a set of axioms that constitute a logical theory. We illustrate these different forms of appearance in ontology engineering, machine-processing and reasoning by an example.

For our illustration we use the Semantic Web Research Community (SWRC) ontology[4] [19], which was proposed to represent knowledge about researchers, research communities, their publications and association to research institutes and universities. It was an early attempt to utilise semantic technology in academia, and it was used in semantic community web portals mainly for the handling of bibliographic meta data of scientific publications in various research projects. The

[4] http://ontoware.org/projects/swrc/

SWRC ontology covers the typical notions of research communities and speaks about universities, faculty staff, PhD students, publications and conferences, etc.

Graphical Appearance

To a knowledge engineer an ontology is often visualized as some form of semantic network. Figure 2 shows the graphical visualization of a part of the SWRC ontology.

As common to most ontology development environments, the visualization in Fig. 2 presents to the knowledge engineer a taxonomic hierarchy of the concepts in the ontology, indicated by *isa*-links.[5] Taxonomic information is modeled for different kinds of students, namely graduates, undergraduates and PhD students, as well as for academic staff, such as professors, both being special kinds of persons. The visualization also shows conceptual relations, such as for the cooperation of academic staff or for expressing the relationship between students and their universities. Moreover, the graph shows some concrete research projects and conferences, modeled as instances of their respective concepts.

Technical Appearance

Not all the information in an ontology can easily be visualized in a graph as the one shown in Fig. 2. For some more detailed information, such as complex axioms and restrictions on concepts, there does not seem to exist any appropriate visualization paradigm other than exposing such fragments of the ontology in a

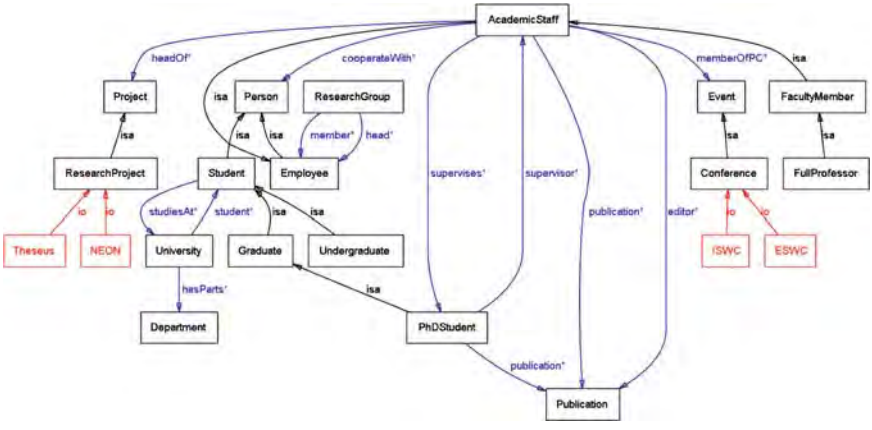


Fig. 2 A graphical visualization for parts of the SWRC ontology

[5] The ontology graph in Fig. 2 has been produced with the OntoViz-plugin for the Protégé environment (http://protege.stanford.edu/plugins/owl/).

formal language. Therefore, ontology engineering environments usually provide extra means for displaying and editing such complex axiomatic information, using a special-purpose ontology language or logical formal notation. When the environment exports the ontology for storage on a disk or for transfer over the wire, all of its information is expressed in the ontology language supported by the tool. Hence, the way an ontology appears to a developer of an ontology editor, storage facility or reasoning tool is in form of some serialization format suitable for machine processing.

For illustrating a fragment of the SWRC ontology we use the RDF [20] serialisation format of the OWL [21] ontology language in the following listing.

```
...
<owl:Class rdf:ID="Conference">
    <rdfs:subClassOf rdf:resource="#Event"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#participant"/>
            <owl:allValuesFrom rdf:resource="#Person"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#Workshop"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="cooperateWith">
    <rdf:type rdf:resource="&owl;SymmetricProperty"/>
    <rdfs:range>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Organization"/>
                <owl:Class rdf:about="#Person"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:range>
    <owl:inverseOf rdf:resource="#cooperateWith"/>
</owl:ObjectProperty>
<Conference rdf:ID="ISWC"/>
...
```

The listing shows an excerpt of the SWRC ontology as it is processed by software systems for serialisation and parsing and for transfer over a network. It exhibits the specification of OWL classes (concepts), properties (relations) and individuals (instances), all expressed by tags and attributes of a customised XML serialisation.

Formal Appearance

As ontology languages like OWL are based on logical formalisms, the formal semantics of the language precisely defines the meaning of an ontology in terms of logic. To a reasoner, therefore, an ontology appears as a set of logical formulas that express the axioms of a logical theory. It can verify whether these axioms are

consistent or derive logical consequences. This form of appearance of an ontology is free of syntactical or graphical additions or ambiguities and reflects the pure knowledge representation aspect.

We use description logic notation to exemplify some axioms of the SWRC ontology in their logical form.[6] The following DL formulas give examples of axioms for subsumption, property range specification and class disjointness.

$$
\begin{aligned}
&\ldots \\
\mathit{Employee} &\sqsubseteq \mathit{Person} \\
\mathit{AcademicStaff} &\sqsubseteq \mathit{Employee} \\
\mathit{Student} &\sqsubseteq \mathit{Person} \\
\mathit{Graduate} &\sqsubseteq \mathit{Student} \\
\mathit{PhDStudent} &\sqsubseteq \mathit{Graduate} \\
&\ldots \\
\mathit{AcademicStaff} &\sqsubseteq \forall\, \mathit{supervises}.\mathit{PhDStudent} \\
\mathit{AcademicStaff} &\sqsubseteq \forall\, \mathit{memberOfPC}.\mathit{Event} \\
\mathit{Publication} &\sqsubseteq \forall\, \mathit{contributor}.(\mathit{Person} \sqcup \mathit{Organisation}) \\
&\ldots \\
\mathit{AcademicStaff} \sqcap \mathit{TechnicalStaff} &\sqsubseteq \bot \\
\mathit{AcademicStaff} \sqcap \mathit{Manager} &\sqsubseteq \bot \\
&\ldots
\end{aligned}
$$

In this logical form, an ontology is the set of axioms that constitutes the explicit knowledge represented about its domain of interest. By means of automated deduction, implicit knowledge of the same form can be derived but is not part of the ontology's explicit specification.

Utilisation of Ontologies

Often, an ontology is distinguished from a knowledge base in that it is supposed to describe knowledge on a schema level, i.e., in terms of conceptual taxonomies and general statements, whereas the more data-intensive knowledge base is thought of containing instance information on particular situations. We take a different perspective and perceive the relation between an ontology and a knowledge base as the connection between an epistemological specification of domain knowledge and a technical means for working with knowledge. From this point of view, an ontology is a piece of knowledge that can be used by a knowledge-based application among other pieces of knowledge, e.g., other ontologies or meta data. To properly cover its domain of interest, it can make use of both schema level and instance level information. Whenever the knowledge-based system needs to consult

[6] Here we refer to an extended version of the SWRC ontology with a stronger axiomatization.

the ontology it accesses (parts of) its specification through a knowledge base, most likely together with other pieces of knowledge, via the tell–ask–interface as shown in Sect. 1.

Usage of Ontologies

The computational domain model of an ontology can be used for various purposes, and some typical types of applications have evolved that make use of ontologies in different ways. We list some of them as examples of how applications can leverage the formalized conceptual domain models that ontologies provide.

- *Information integration*
  A promising field of application for ontologies is their use for integrating heterogeneous information sources on the schema level. Often, different databases store the same kind of information but adhere to different data models. An ontology can be used to mediate between database schemas, allowing to integrate information from differently organised sources and to interpret data from one source under the schema of another.
- *Information retrieval*
  Motivated by the success and key role of Google[7] in the World Wide Web, semantically enhanced information retrieval on web documents is a widely recognised field of application, and the use of ontologies is one particular approach to improving the retrieval process. The idea behind ontology-based information retrieval is to increase the precision of retrieval results by taking into account the semantic information contained in queries and documents, lifting keywords to ontological concepts and relations.
- *Semantically enhanced content management*
  In many areas of computation the data that is actually computed is annotated with meta data for various purposes. Ontologies provide the domain-specific vocabulary for annotating data with meta data. The formality of ontology languages allows for an automated processing of this meta data and their grounding in knowledge representation facilitates machine-interpretability.
- *Knowledge management and community portals*
  In companies or other organised associations, or in communities of practice, individual knowledge can be viewed as a strategic resource that is desirable to be shared and systematically maintained, which is referred to as *knowledge management*. Ontologies provide a means to unify knowledge management efforts under a shared conceptual domain model, connecting technical systems for navigating, storing, searching and exchanging community knowledge.
- *Expert systems*
  In various domains, such as medical diagnosis or legal advice in case-law, it is desirable to simulate a domain expert who can be asked sophisticated questions.

[7] http://www.google.com

In an expert system, this is achieved by incorporating a thoroughly developed domain ontology that formalises expert knowledge. Domain-specific questions can then be answered by reasoning over such highly specialized knowledge.

Types of Ontologies

The term ontology is used for a variety of artefacts employed for diverse purposes, and a categorisation of ontologies can be made according to their subject of conceptualisation. Figure 3 shows a distinction between the following types of ontologies, according to [22].

- *Top-level ontologies*
  Top-level ontologies attempt to describe very abstract and general concepts that can be shared across many domains and applications.[8] They borrow from philosophical notions, describing top-level concepts for all things that exist, such as "physical object" or "abstract object", as well as generic notions of commonsense knowledge about phenomena like time, space, processes, etc. They are usually well thought out and extensively axiomatised. Due to their generality, they are typically not directly used for conceptual modeling in applications but reused as a basis for building more specific ontologies. Prominent examples for top-level ontologies are DOLCE [23] and SUMO [24].
- *Domain ontologies and task ontologies*
  These types of ontologies capture the knowledge within a specific domain of discourse, such as medicine or geography, or the knowledge about a particular task, such as diagnosis or configuration. In this sense, they have a much narrower and more specific scope than top-level ontologies. Prominent ontologies exist in natural sciences, such as medicine, genetics, geographic and communal efforts such

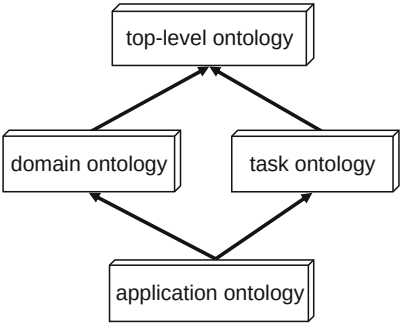


Fig. 3 Types of ontologies

[8] Top-level ontologies are sometimes also called upper ontologies or foundational ontologies.

as environment information, tourism, as well as cultural heritage and museum exhibits. Examples are GALEN[9] for the medical domain or GO[10] for the domain of bio-informatics. Task ontologies have been devised for scheduling and planning tasks, intelligent computer-based tutoring, missile tracking, execution of clinical guidelines, etc.

- *Application ontologies*
  Further narrowing the scope, application ontologies provide the specific vocabulary required to describe a certain task enactment in a particular application context. They typically make use of both domain and task ontologies, and describe e.g., the role that some domain entity plays in a specific task (see e.g., [25]).

Altogether, we can say that Fig. 3 represents an inclusion scheme: the lower ontologies inherit and specialise concepts and relations from the upper ones. The lower ontologies are more specific and have thus a narrower application scope, whereas the upper ones have a broader potential for reuse.

2.3 Semantic Web and Ontology Languages

One particular application of ontologies is their use for annotation of web content in the vision of the Semantic Web. To make ontologies available to information systems for this purpose, various ontology languages have been designed and proposed for standardisation.

The Semantic Web Vision

The World Wide Web has become a powerful tool for communication, research and commerce, however, it is limited to manual navigation of human users who interpret the content of web sites in order to access the information they provide. As stated in [26], the vision of the *Semantic Web* is to make the Web machine-understandable, allowing computers to integrate information from disparate sources to achieve the goals of end users. To this end, data in the World Wide Web is to be upgraded to a semantic level, such that it can be used by machines not just for display purposes, but for automation, integration and reuse across various applications in an automated way.

In the context of the Semantic Web, ontologies play a particularly important key role. While the content of the current web is primarily produced for human consumption, also information produced mainly for machines, such as the records in

[9] http://www.co-ode.org/galen/

[10] http://www.geneontology.org

a database, should be made available to processing over the web. The idea of the Semantic Web is to annotate both human-readable and machine-tailored web content by machine-interpretable meta data, such that computers are able to process this content on a semantic level. Ontologies provide the domain vocabulary in terms of which semantic annotation is formulated. Meta statements about web content in such annotations refer to a commonly used domain model by including the concepts, relations and instances of a domain ontology. The formality of ontology languages allows to reason about semantic annotation from different sources, connected to background knowledge in the domain of interest.

There are characteristics of the web that affect the use of ontologies for semantic annotation. One aspect is the natural distributedness of content in the web. The knowledge captured in semantic annotations and ontologies is not locally available at a single node but spread over different sites. This imposes additional constraints on the use of ontologies in the Semantic Web, taking into account distributedness of knowledge. Techniques for unique identification of ontological entities or for a modular and distributed organisation of ontologies and the reasoning process are required. Another related aspect is that content on the web is created in an evolutionary manner and maintained in a decentralised way. There is no central control over semantic annotation or ontologies that evolve in the Semantic Web, and information in one ontology can conflict with information in another one. To either avoid or deal with conflicting pieces of knowledge, modeling methodologies or techniques for treating inconsistencies are required.

Semantic Web Ontology Languages

Standardisation of ontology languages is of high importance to businesses, since it ensures that data can be exchanged between business partners and that investments made in a particular technology are safe due to broad vendor support. Various different aspects are considered for language standardisation, such as issues of the underlying knowledge representation formalism in terms of expressiveness and computational properties, web-related features like global unique identification and XML serialisation syntax, or usability add-ons like the inclusion of strings and numbers. The influence of different research and user communities with manifold requirements have resulted in a complex landscape of a multitude of languages backed by different past and ongoing standardisation efforts. It is still an open topic stimulating lively discussions in current research which languages are best suited for what purpose, how they can be efficiently implemented, realised in a user-friendly way, or technically and semantically made interoperable.

An early ontology language standard emerged as part of metadata standardisation efforts from the World Wide Web consortium (W3C) in the Semantic Web activity. This effort resulted in the Resource Description Framework **RDF** [20] and a simple ontology language **RDFS** (RDF Schema) [27], which has now become a well established and widely accepted standard for encoding meta data. The **RDF(S)** language can be used to express class-membership of resources and subsumption between

classes but its peculiar semantics does neither fit the classical nor the LP-style; however, translations to both paradigms exist that loose only few of RDF(S) functionality.

On top of RDF(S), W3C standardisation efforts have produced the **OWL** [21] family of languages for describing ontologies in the Web, which comes in several variants with increasing expressiveness. Only the most expressive language variant, namely **OWL-Full**, has a semantically proper layering on top of RDF(S), allowing for features of metamodeling and reification. The less expressive variants **OWL-Lite** and **OWL-DL** map to certain description logic dialects and fit the classical semantics as subsets of **first-order logic**. Besides the class membership and subsumption relations inherited from RDF(S), OWL offers the construction of complex classes from simpler ones by means of DL-style concepts constructors. Among ongoing standardisation efforts, OWL-DL is currently the most prominent Semantic Web ontology language following the description logic paradigm.

A current research trend is to integrate DL-style ontologies with LP-style rules to be interoperable on a semantic level. One attempt to do so is the Semantic Web Rule Language (**SWRL**[11]) that extends the set of OWL axioms to include Horn-like rules interpreted under first-order semantics. Interoperability with OWL ontologies is realised by referring to OWL classes and properties within SWRL rules, however, the combination of OWL-DL and SWRL rules results in an undecidable formalism. Another approach to amalgamate OWL ontologies and rules are the so-called **DL-safe rules** [28], which extend DL knowledge bases in a way similar to SWRL. However, DL-safe rules preserve decidability of the resulting language by imposing an additional safety restriction on SWRL rules which ensures that they are only applied to individuals explicitly known to the knowledge base.

Languages that follow the logic programming paradigm mainly stem from deductive database systems, which apply rules on the facts stored in a database to derive new facts by means of logical inferencing. A common declarative language used in deductive databases is **Datalog** [9], which is syntactically similar to Prolog [3]. In the Semantic Web context, **F-Logic** [29] is a more prominent rule language that combines logical formulas with object-oriented and frame-based description features. In its logic programming variant **F-Logic (LP)**, it adopts the semantics of Datalog rules.

Furthermore, **Topic Maps**[12] is another means to describe ontologies in the web context. In contrast to other languages mentioned here, it is rather based on a graph-oriented paradigm than on formal semantics and precise meaning.

Finally, the Web Service Modeling Language (**WSML**) [30] family is the most recent attempt to standardise ontology languages for the web, with a special focus on annotating Semantic Web Services.

[11] http://www.w3.org/Submission/SWRL/

[12] http://www.topicmaps.org

3 Outlook

In this chapter, we have presented an overview on the topics of knowledge representation and ontologies. In particular, we have given an introduction to the basic principles of representation formalisms originating in symbolic Artificial Intelligence as well as paradigms related to modeling of and reasoning with knowledge. We have also characterised the notion of an ontology as a conceptual yet computational model that makes domain knowledge available to computer systems, and we have addressed the utilisation of ontologies as well as their formalisation through formal knowledge representation languages that currently evolve in the context of the Semantic Web.

Despite being a new filed, there can be observed some achievements in ontology research. As the techniques from knowledge representation in Artificial Intelligence have been taken up again for studying ontologies, their formal foundations are well studied. The power and limitations of ontology languages are clearly determined by their underlying logical formalisms, for most of which the characteristics and complexity of reasoning problems are well known. Due to standardisation of ontology languages, in particular the W3C standards RDF(S) and OWL, the idea of using ontologies is picked up in many research areas. Tools for handling ontologies become available to researchers within and outside the WWW community, where ontologies have a major impact on the Semantic Web.

However, there are still challenges to be faced for ontologies to gain larger momentum. To achieve wide-spread use of ontologies, they have to be established as usable software artefacts that are interchanged and traded between parties, similar to computer programs or other forms of electronic content. As such, they can principally be plugged in systems that make use of knowledge-based technology. However, the logic-based notions in which ontologies are described are typically too technical and too onerous to handle to be widely accepted. To overcome this deficiency, design methodologies and higher-level descriptive languages need to be introduced that abstract from the surfeit of logical details, presenting the user a more intuitive view on domain knowledge. An analogous level of abstraction has been achieved in the field of software engineering, where more and more abstract higher-level languages have been build on top of machine codes and assembler languages.

Moreover, ontologies have not been largely taken up on the application side so far. While the usage of ontologies as computational domain models is clearly understood in principle, break-through applications with a wide acceptance and significant impact have not been identified yet. The recent trend to utilise ontologies for building intelligent web-based systems in the Semantic Web is still restricted to research prototypes that cover only limited domains of interest. For ontologies gaining greater momentum in applications, it needs to be investigated at which places web-based or other applications should rely on knowledge-based decisions and about which parts of the involved domains knowledge should be represented to find a good trade-off between modeling effort and automation.

References

1. J.F. Sowa, *Knowledge Representation* (Brooks Cole Publishing, CA, 2000)
2. S. Russel, P. Norvig, *Artificial Intelligence – A Modern Approach* (Prentice-Hall, NJ, 1995)
3. J.W. Lloyd, *Foundations of Logic Programming* (Springer, Berlin, 1988)
4. J. Minker, AI Magazine **18**(3), 21–47 (1997)
5. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider (eds.), *The Description Logic Handbook* (Cambridge University Press, Cambridge, 2003)
6. R.J. Brachmann, J.G. Schmolze, Cognit. Sci. **9**(2), 171–202 (1985)
7. R.J. Brachman, D.L. McGuiness, P.F. Patel-Schneider, L.A. Resnick, in *Living with CLASSIC: When and How to Use a KL-ONE-like Language*, ed. by J. Sowa. Principles of Semantic Networks (Morgan Kaufmann, San Mateo, 1990)
8. C. Lutz, in *Description Logics with Concrete Domains – A Survey*. Advances in Modal Logics, vol. 4 (King's College Publications, 2003)
9. J.D. Ullman, *Principles of Database and Knowledge-Base Systems*, vol. I and II (Computer Science Press, 1989)
10. T. Eiter, G. Gottlob, H. Mannila, ACM Trans. Database Syst. **22**(3), 364–418 (1997)
11. M. Gelfond, V. Lifschitz, The Stable Model Semantics For Logic Programming, ed. by R. Kowalski, K. Bowen. in *Proceedings of the 5th International Conference on Logic Programming*, Cambridge, MA, 1988, pp. 1070–1080
12. G. Antoniou, *Nonmonotonic Reasoning* (MIT, MA, 1997)
13. B. Motik, On the Properties of Metamodeling in OWL. ed. by Y. Gil, E. Motta, V.R. Benjamins, M. Musen. in *Proceedings of the 4th International Semantic Web Conference (ISWC'05)*, vol. 3729 of LNCS, Springer, Berlin, 2005, pp. 548–562
14. B. Motik, I. Horrocks, R. Rosati, U. Sattler, Can OWL and Logic Programming Live Together Happily Ever After? ed. by I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, L. Aroyo. in *Proceedings of the 5th International Semantic Web Conference (ISWC'06)*, vol. 4273 of LNCS, Springer, Berlin, 2006, pp. 501–514
15. B. Motik, R. Rosati, A faithful integration of description logics with logic programming. in *Proceedings of the Internationl Joint Conference of Artificial Intelligence (IJCAI'07)*, pp. 477–482, 2007
16. E. Craig, in *Ontology*, ed. by E. Craig. Routledge Encyclopedia of Philosophy (Routledge, New York, 1998), pp. 117–118
17. T.R. Gruber, J. Knowl. Acquis. **6**(2), 199–221 (1993)
18. N. Guarino, Formal Ontology and Information Systems, Preface. ed. by N. Guarino. in *Proceedings of the 1st International Conference on Formal Ontologies in Information Systems (FOIS'98)*, IOS Press, 1998, pp. 3–15
19. Y. Sure, S. Bloehdorn, P. Haase, J. Hartmann, D. Oberle, The SWRC ontology – Semantic Web for research communities. in *Proceedings of the 12th Portuguese Conference on Artificial Intelligence (EPIA 2005)*, vol. 3803 of Lecture Notes in Computer Science, Springer, Berlin, 2005, pp. 218–231
20. G. Klyne, J. Carroll, RDF Concepts and Abstract Syntax. http://www.w3.org/TR/rdf-primer/, 2004
21. P.F. Patel-Schneider, P. Hayes, I. Horrocks, OWL Web Ontology Language; Semantics and Abstract Syntax. http://www.w3.org/TR/owl-semantics/, November 2002
22. N. Guarino, in *Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration*, ed. by M.T. Pazienza. Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology, 1299 in LNCS, Springer, Berlin, 1997, pp. 139–170
23. A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, L. Schneider, Sweetening Ontologies with DOLCE. in *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW'02). Ontologies and the Semantic Web*, Springer, Berlin, 2002, pp. 166–181

24. I. Niles, A. Pease, Towards a Standard Upper Ontology. ed. by C. Welty, B. Smith. in *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS'01)*, 2001
25. R. Studer, R. Benjamins, D. Fensel, J. Data Knowl. Eng. **25**(1–2), 161–197 (1998)
26. T. Berners-Lee, J. Hendler, O. Lassila, *The Semantic Web* (Scientific American, 2001), pp. 28–37
27. D. Brickley, R.V. Guha, RDF Vocabulary Description Language – RDF Schema. http://www.w3.org/TR/rdf-schema/, 2004
28. B. Motik, U. Sattler, R. Studer, Query Answering for OWL-DL with Rules. ed. by S.A. McIlraith, D. Plexousakis, F. van Harmelen. in *Proceedings of the 3rd International Semantic Web Conference (ISWC'04)*, vol. 3298 of LNCS, Springer, Berlin, 2004, pp. 549–563
29. M. Kifer, G. Lausen, J. Wu, J. ACM **42**(4), 741–843 (1995)
30. J. de Bruijn, H. Lausen, A. Polleres, D. Fensel, The Web Service Modeling Language WSML: An Overview. in *Proceedings of the 3rd European Semantic Web Conference (ESWC'06)*, vol. 4011 of Lecture Notes in Computer Science, Springer, Berlin, June 2006

“This page left intentionally blank.”

Part II
Computational Science

“This page left intentionally blank.”

Spatial Techniques

Nafaa Jabeur and Nabil Sahli

1 Introduction

The environment, including the Earth and the immense space, is recognized to be the main source of useful information for human beings. During several decades, the acquisition of data from this environment was constrained by tools and techniques with limited capabilities. However, thanks to continuous technological advances, spatial data are available in huge quantities for different applications. The technological advances have been achieved in terms of hardware and software as well. They are allowing for better accuracy and availability, which in turn improves the quality and quantity of useful knowledge that can be extracted from the environment. They have been applied to geography, resulting in geospatial techniques. Applied to both science and technology, geospatial techniques resulted in areas of expertise, such as land surveying, cartography, navigation, remote sensing, Geographic Information Systems (GISs), and Global Positioning Systems (GPSs). They had evolved quickly with advances in computing, satellite technology and a growing demand to understand our global environment. In this chapter, we will discuss three important techniques that are widely used in spatial data acquisition and analysis: GPS and remote sensing techniques that are used to collect spatial data and a GIS that is used to store, manipulate, analyze, and visualize spatial data. Later in this book, we will discuss the techniques that are currently available for spatial knowledge discovery.

2 Global Positioning System

2.1 What is GPS?

The GPS is a satellite-based navigation system. It uses a constellation of at least 24 (32 by March 2008) satellites orbiting the Earth and transmitting precise microwave

N. Jabeur (✉)
Department of Computer Science, Dhofar University, Salalah, Sultanate of Oman
e-mail: nafaa_jabeur@du.edu.om

M.M. Gaber (ed.), *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*, DOI 10.1007/978-3-642-02788-8_7,

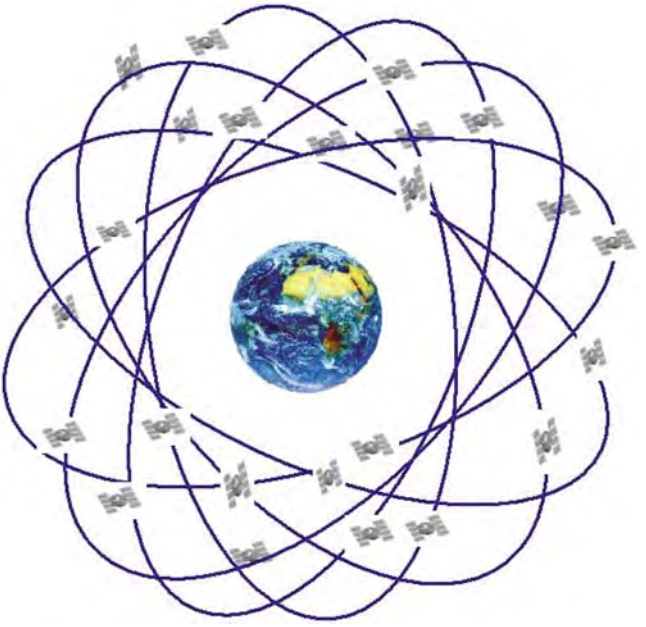


Fig. 1 GPS constellation of 24 satellites

signals, that enable GPS receivers to determine their location, speed, direction, and time (Fig. 1). At its simplest level, a GPS receiver may just tell you your longitude and latitude co-ordinates. But these days, most GPS devices come with a map and mark the device's location on the map as its owner moves. GPS works in any weather conditions, anywhere in the world, 24 h a day. It was placed into orbit by the US Department of Defense and originally intended for military applications. However, in the 1980s the government made the system available for civilians. Neither subscription fees nor setup charges are required. The official name of GPS is NAVSTAR-GPS. Similar satellite navigation systems include the Russian GLONASS (incomplete as of 2008), the upcoming European Galileo positioning system, the proposed COMPASS navigation system of China, and IRNSS of India [1]. In addition to military applications, GPS became widely used in worldwide navigation, map-making, land-surveying, commerce, and scientific research among others. In the military field, GPS allows soldiers to track potential ground and air targets before they are flagged as hostile and coordinate their movements and supplies accordingly. It also allows them to accurately target various military weapons including cruise missiles and precision-guided munitions. In these tasks, GPS highly contribute in map creation and rescue missions. In the civilian field, GPS enables receivers to calculate local velocity and orientation. This information is useful in vessels or observations of the Earth. GPS also enables researchers to explore the Earth environment including the atmosphere, ionosphere, and gravity field. In addition, it enables large communication and observation systems to synchronize clocks to exacting standards [1].

2.2 Method of Operation

From high above the Earth, each satellite continually transmits messages containing the time the message was sent, a precise orbit for the satellite sending the message (the ephemeris), and the general system health and rough orbits of all GPS satellites (the almanac). These messages travel at the speed of light through outer space, and slightly slower through the atmosphere. By timing them carefully, a GPS receiver is able to calculate its position. It uses the arrival time of each message to measure the distance to each satellite. This can be achieved with a technique called trilateration that determines the relative positions of objects on the basis of geometry of triangles in a similar fashion as triangulation. The resulting coordinates calculated by the receiver are converted to more user-friendly forms such as latitudes and longitudes, or location on a map. These coordinates are finally displayed to the user (Fig. 2). It might seem that three satellites would be enough for the GPS receiver in order to calculate its position, especially since space has three dimensions. However, a fourth satellite should always be used to compensate for inaccurate clock in GPS receivers, and therefore allowing for much better accuracy (Fig. 3). With four satellites, the receiver determines four measurements: altitude, latitude, longitude, and

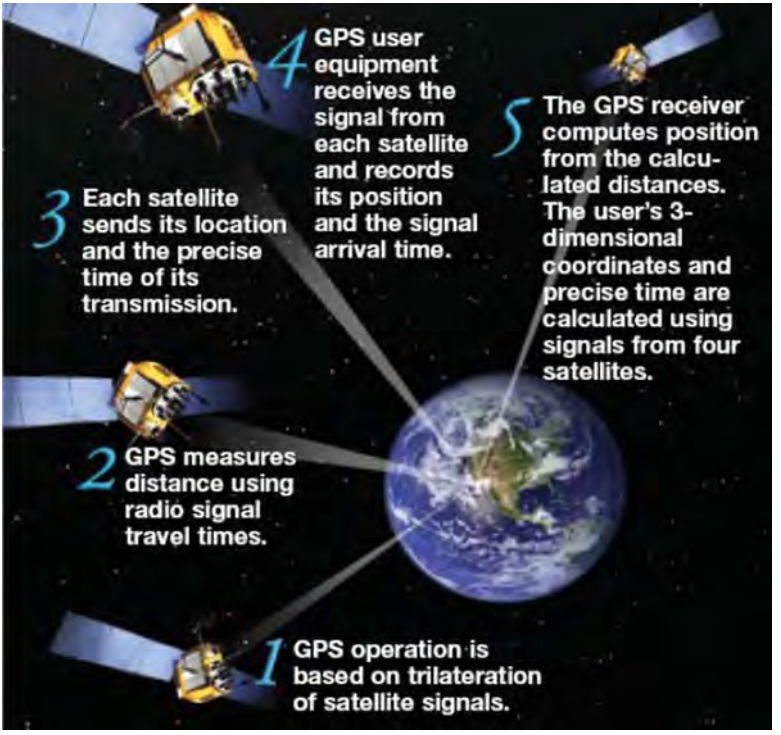


Fig. 2 GPS operational concept [2]

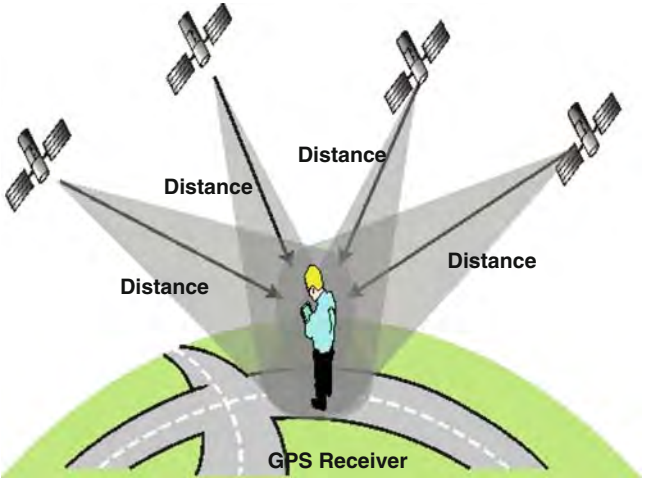


Fig. 3 Basic principle of position with GPS [3]

time. These parameters are important, but not all of them are necessary all the time. For example, knowing its altitude (which is 0), a ship needs only three satellites to compute its position at sea. Moreover, when fewer than our satellites are visible, some additional clues can be used. Examples of clues include last known position, dead reckoning, and inertial navigation [3].

2.3 Technical Description

GPS consists of three major segments (Fig. 4): Space Segment, Control segment, and User segment.

Space Segment

Space segment consists of orbiting satellites that send radio signals from space [4]. The first of the satellites was brought to its orbit as early as 1978. During the years the satellites became more and more sophisticated (Fig. 5). Currently, five different types of these satellites exist (Block I, Block II, Block IIA, Block IIR and Block IIF). Originally, the space segment comprises 24 GPS satellites orbiting the Earth every 12 h. There are six orbital plans containing four satellites each. These satellites are equally spaced 60° apart and are inclined at about 55° with respect to the equator. In this configuration at least four satellites are available from any location on the Earth's surface at all times. This configuration has been changed since September 2007. Indeed, the constellation is henceforth a non-uniform arrangement after increasing the number of satellites to 31 [1]. The additional satellites allow for

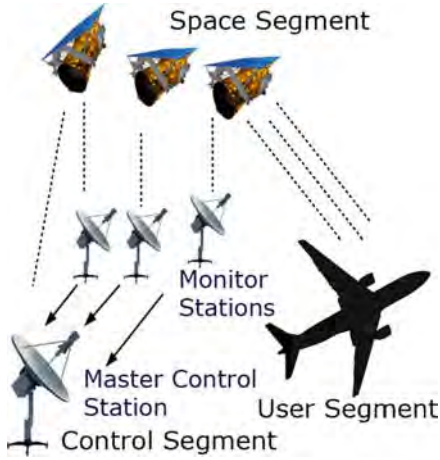


Fig. 4 The segments of GPS

Fig. 5 GPS Block I, Block IIA, and Block IIF satellites

redundant measurements that improve the precision of GPS receiver calculations. They also allow for more reliability and availability of the system, particularly when multiple satellites fail.

Control Segment

The Control segment consists of tracking stations located around the world (Fig. 6). These stations consist of Master Control Stations (MCS), several Monitor Stations (MS), and Ground Antennas (GA). The operational control segment consists of MCS near Colorado Springs (USA), three MS and GA in Kwajaleian Ascension and Diego Garcia, and two more MS at Colorado Spring and Hawai. The main functions of the Control Segment include: (1) monitoring and controlling the satellite

Fig. 6 Position of the tracking stations around the Earth

system continuously, (2) determining GPS system time, (3) predicting the satellite ephemeris and the behavior of each satellite clock, and (4) updating periodically the navigation message for each particular satellite [1].

User Segment

The User segment consists of GPS receivers that are appropriate for receiving signals from GPS satellites. These receivers convert this signal into position, velocity, and time estimates. They are mainly used in navigation (e.g., for aircraft and ships), precise positioning (e.g., for surveying and plate tectonics), and time and frequency dissemination (e.g., for astronomical observatories and telecommunications facilities). GPS receivers come in a variety of formats (Fig. 7). These formats range from devices integrated into cars, phones, and watches, to dedicated devices. The main components of GPS receivers include: (1) an antenna with pre-amplifier for the capture of frequencies transmitted by the satellites; (2) a radio-frequency section with signal identification and signal processing; (3) a micro-processor for receiver control, data sampling and data processing; (4) a receiver that is often described by its number of channels (i.e., how many satellites it can monitor simultaneously); (5) a precision oscillator; (6) a power supply; (7) a user interface, command and display panel (for providing location and speed information to the user); and (8) a memory for data storage [4].

2.4 Navigation Signals

Every GPS signal packs three bits of information: the pseudorandom code, ephemeris data, and almanac data. The pseudorandom code is the identification

Fig. 7 Examples of GPS receivers: (*left*) handheld, (*middle*) automotive, and (*right*) mobile

code of the individual satellite. The ephemeris data identifies the location of each GPS satellite at any particular time of the day. Each satellite transmits this data for the GPS receivers as well as for the other satellites in the network. The almanac data has information about the status of the satellite as well as current date and time. The almanac part of the signal is essential for determining the position. The GPS signals are transmitted at two frequencies of low power radio. These frequencies are called L1 and L2. The L1 frequency at 1575.42 MHz GPS carrier contains the C/A-Code, the encrypted P-Code (or Y-Code), and the Navigation Message [1]. It comes into play for civilian applications. Commercial GPS navigation receivers can track only the L1 carrier to make pseudo-range measurements. The L2 frequency at 1227.60 MHz GPS carrier contains only the encrypted P-Code (or Y-Code) and the Navigation Message. The Y-Code is usually encrypted and reserved for military applications. Signals at L1 and L2 frequencies can pass through clouds, glass, and plastic. However, they cannot go through more solid objects like buildings and mountains [1].

2.5 GPS Error Sources and Jamming

In order to calculate the position, the receiver needs three parameters: the current time, the position of the satellite, and the measured delay of the received signal. Due to several parameters, some errors are introduced in the GPS signal (Fig. 8). Apart from the inaccuracy of the clock in the GPS receiver, the following factors affect the quality of the GPS signal and cause calculation errors:

- *Atmospheric effects (Fig. 9)*: Ionosphere and troposphere disturbances affect the speed of the GPS signals as they pass through the Earth's atmosphere. These effects are smallest when the satellite is directly overhead and become greater for satellites nearer the horizon since the path through the atmosphere is longer. Avoiding these effects and correcting their errors is a significant challenge to improving GPS position accuracy [5].

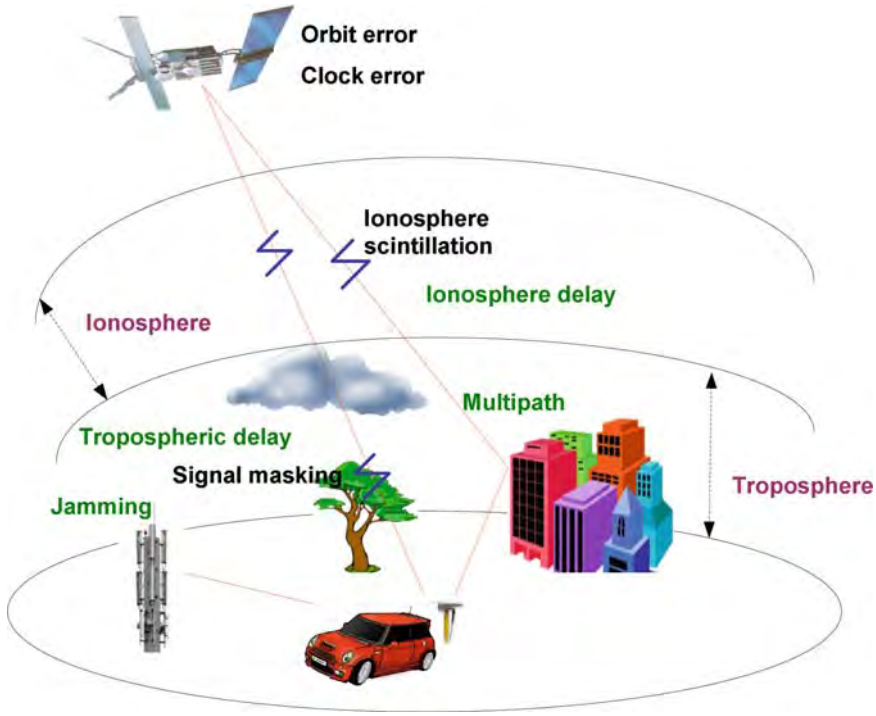


Fig. 8 Sources of GPS signal errors

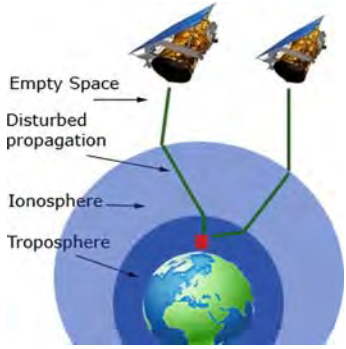


Fig. 9 Influences on radio waves propagation through the Earth's atmosphere

- *Multipath or Signal reflection (Fig. 10)*: The GPS signal may be delayed due to its reflection on objects, like tall buildings and rocks. The reflected signal takes more time to reach the receiver than the direct signal. The resulting error typically lies in the range of a few meters [5].
- *Ephemeris errors*: Ephemeris errors are also known as orbital errors. These are errors in the satellite's reported position against its actual position.

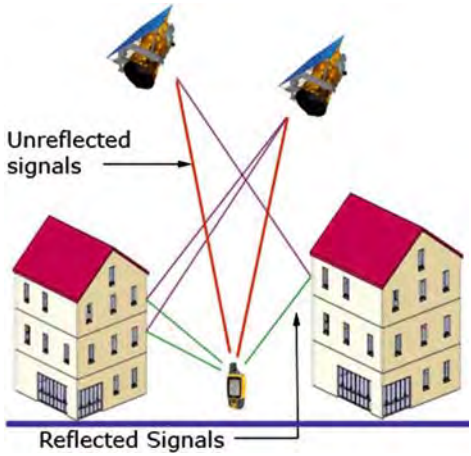


Fig. 10 Interference caused by reflection of the signal

- *Clock errors*: The satellite's atomic clocks experience noise and clock drift errors. These errors tend to be very small, but may add up to a few meters of inaccuracy [6]. Corrections of these errors and estimation of the accuracy are included in the navigation message. Moreover, the built in clock of the GPS receiver is not as accurate as the atomic clocks of the satellites and the slight timing errors leads to corresponding errors in calculations.
- *Visibility of Satellites*: The more the number of satellites a GPS receiver can lock onto, the better its accuracy. This accuracy may be reduced intentionally by switching off the Selective Availability feature included in the GPS. This feature can introduce intentional, slowly changing random errors of up to a hundred meters into the publicly available navigation signals. It is used, in order to confound the guidance of long-range missiles to precise targets. When this feature is enabled, the accuracy is still available in the signal, but in an encrypted form that is only available to the United States military, its allies and a few others, mostly government users. Even those who have managed to acquire military GPS receivers would still need to obtain the daily key, whose dissemination is tightly controlled [1]. Moreover, everything that comes in the line of sight, such as buildings, rocks and mountains, dense foliage, and electronic interference, cause position errors and sometimes make it unable to take any reading at all. GPS receivers do not work indoors, underwater, and underground.
- *Satellite Shading (Fig. 11)*: For the signals to work properly, the satellites have to be placed at wide angles from each other. Poor geometry resulting from tight grouping can result in signal interference [5].
- *Intentional degradation*: This was used till May 2000 by the US Department of Defense so that military adversaries could not use the GPS signals. This has been turned off since May 2000, which has improved the accuracy of readings in civilian equipment [1].

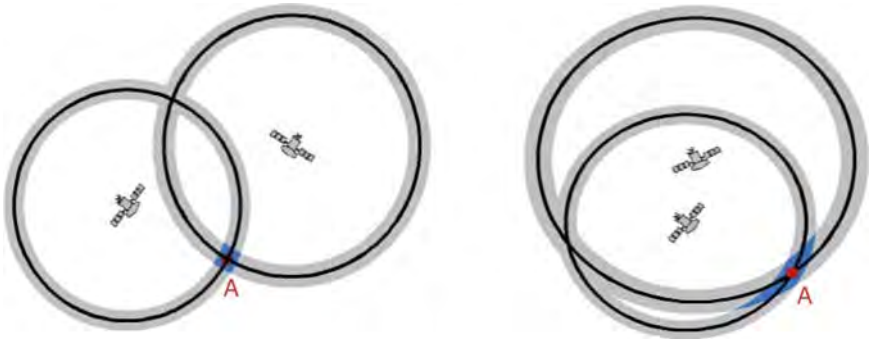


Fig. 11 Geometrical alignment of two satellites: (*left*) good, and (*right*) bad

In addition to the sources of errors mentioned above, several natural sources (e.g., solar flares and geomagnetic storms) may cause interferences and jamming that result in a difficult acquisition of GPS signals. Interferences and jamming may also be caused by artificial sources, such as defrosters, car window tinting films, and stronger signals when they are within radio range. These undesirable effects may be intentional, particularly for military purposes.

2.6 Improving the Accuracy of GPS Signals

Several techniques continue to be developed aiming to deal with errors, interferences, and jamming. Among these techniques we can mention [7–9]: *Differential GPS*. Differential GPS or DGPS brings the accuracy of readings to within 1–3 meters of the object, as compared to the 4–20 m of normal GPS. DGPS works using a network of stationary GPS receivers. The difference between their predefined position and the position as calculated by the signals from satellites gives the error factor. This error component is then transmitted as a FM signal for the local GPS receivers, enabling them to apply the necessary correction to their readings. *Wide Area Augmentation System (WAAS)*. WAAS works with ground reference stations to calculate the required correction. The correction message is then relayed to GPS receivers through additional geostationary orbits. WAAS is useful in poor visibility and precision landing situations. It is also useful in high-precision applications like GPS-based instrument approaches in aviation. WAAS is currently fully operational only in the US, however parallel developments are taking place in Europe and Japan. *Local Area Augmentation System (LAAS)*. LAAS uses similar correction data as WAAS. However LAAS uses a local source at a predefined location to transmit the data. The correction data is typically applicable within only a radius of 20–50 km around the transmitter station. *WAGE (Wide Area GPS Enhancement)*. WAGE works by getting more accurate satellite clock and ephemeris data. However this requires special receivers. *Relative Kinematic Positioning (RKP)*. RKP is another

highly precise system with accuracy levels of within 10 cm of the actual position. RKP uses a combination of DGPS data, signal phase calculations, and ambiguity resolution techniques, the entire processing work done in real time.

3 Remote Sensing

3.1 What is Remote Rensing?

Since the early use of aerial photography, remote sensing has been recognized as a valuable tool for viewing, analyzing, characterizing, and making decisions about our environment. It can be defined as the science, and to some extent the art, of measuring and acquiring information about some features of an object or phenomenon, by a recording device that is not in physical contact with the object or phenomenon being observed [1]. For example, from a spacecraft, relevant information about electromagnetic radiation, Earth's temperature, or devastating storm can be gathered by using cameras, lasers, radar systems, or magnetometers. The collected information needs a physical carrier to travel from the objects/phenomenon to the sensing devices through an intervening medium [10]. The electromagnetic radiation is normally used as an information carrier in remote sensing. The output of a remote sensing system is usually an image representing the scene being observed. A further step of image analysis and interpretation is required in order to extract useful information from the image. The process of remote sensing is familiar for humans. Indeed, as humans we rely on visual perception to provide us with much of the information about our surroundings. However, our eyes, used as sensors, are greatly limited. Actually, they are sensitive only to the visible range of electromagnetic energy. They view perspectives according to the location of our bodies. In addition, they are unable to form a lasting record of what we view. Due to these limitations, humans continuously develop technological means to increase their ability to see and record the physical properties of the environment.

3.2 Principal Process of Remote Sensing

The principal process of remote sensing can be described as follows (Fig. 12):

Energy Source

The energy source (A in Fig. 12) illuminates the object of interest. It provides this object with electromagnetic energy in the form of electromagnetic radiation. Electromagnetic radiation consists of electrical and magnetic fields traveling at the speed

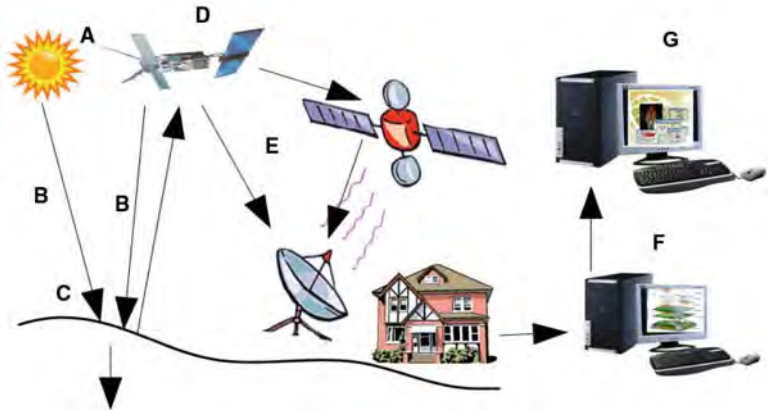


Fig. 12 Principal process in remote sensing

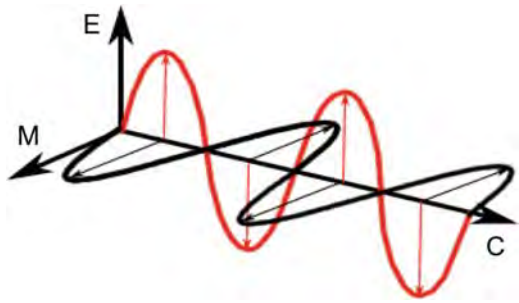


Fig. 13 Electrical and magnetic fields of electromagnetic radiation

of light (Fig. 13). The electrical field varies in magnitude in a direction perpendicular to the direction in which the radiation is traveling. The magnetic field is oriented at right angles to the electrical field. The energy source may be natural or provided by the sensors. Remote sensing systems which measure energy that is naturally available are called passive sensors. Active sensors, on the other hand, provide their own energy source for illumination [10]. In remote sensing, wavelength and frequency are the most important features of electromagnetic radiation. Measured in meters (or some of its factors, e.g., nanometers), the wavelength is the length of one wave cycle. Frequency refers to the number of cycles of a wave passing a fixed point per unit of time. Frequency is normally measured in Hertz (Hz), equivalent to one cycle per second, and various multiples of Hertz. Electromagnetic spectrum is the whole wavelengths which one meets in an electromagnetic radiation. It ranges from the shorter wavelengths (including gamma and X-rays) to the longer wavelengths (including microwaves and broadcast radio waves). There are several regions of the electromagnetic spectrum, which are useful for remote sensing.

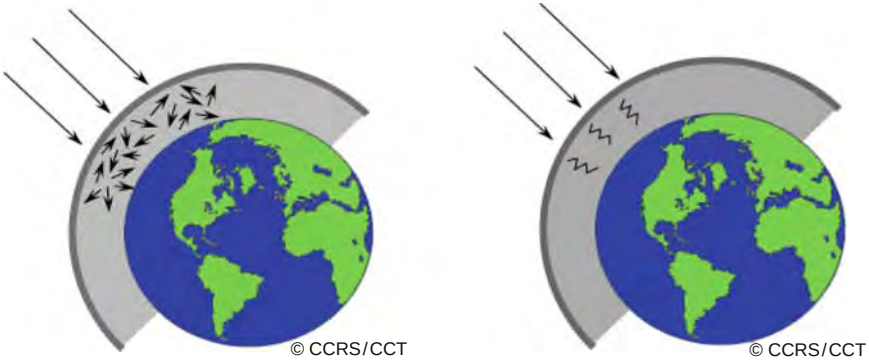


Fig. 14 (*Left*) Scattering; (*right*) absorption

Radiation and Atmosphere

As the energy travels from its source to the target, it will come in contact with and interact with the atmosphere it passes through (B in Fig. 12). This interaction may take place a second time as the energy travels from the target to the sensor. Before radiation reaches the Earth's surface it has to travel through some distance of the Earth's atmosphere. Particles and gases in the atmosphere can affect the incoming radiation. These effects are, particularly, caused by the mechanisms of scattering and absorption [10]. Scattering (Fig. 14) occurs when particles or large gas molecules present in the atmosphere interact with and cause the electromagnetic radiation to be redirected from its original path. In contrast to scattering, absorption causes molecules in the atmosphere to absorb energy at various wavelengths. Ozone, carbon dioxide, and water vapor are the three main atmospheric constituents which absorb radiation.

Interaction with the Target

The energy traveling from the source through the atmosphere interacts with the target (C in Fig. 12). This interaction depends on the properties of both the target and the radiation. Radiation that is not absorbed or scattered in the atmosphere can reach and interact with the Earth's surface [10]. There are three forms of interaction: absorption (A); transmission (T); and reflection (R) (Fig. 15).

Image

Sensors are required to collect and record the electromagnetic radiation (D in Fig. 12) after the energy has been scattered by or emitted from the target. Recording the energy is done as an array of numbers in digital format right from the start.

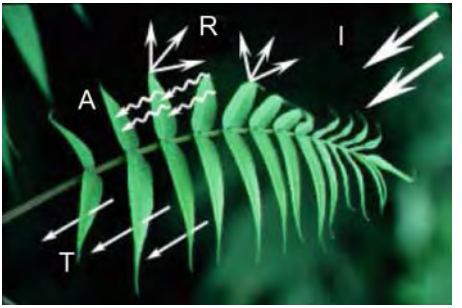


Fig. 15 Types of interactions between the energy and the target

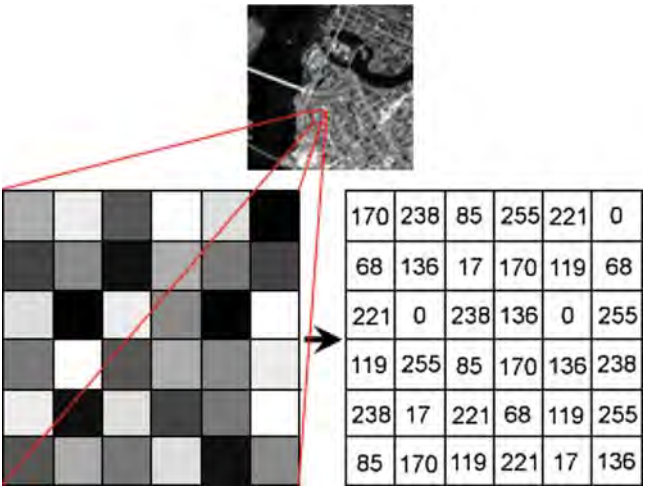


Fig. 16 Digital image representation

Image may be presented in a digital format by subdividing this array into pixels. Every pixel corresponds to a digital number, representing the brightness level of that pixel in the image (Fig. 16).

Analysis and Interpretation

The energy recorded by the sensor has to be transmitted (E in Fig. 12), often in electronic form, to a receiving and processing station where the data are processed into an image (hardcopy and/or digital). The processed image is analyzed and interpreted (F in Fig. 12), visually and/or digitally or electronically, in order to extract information about the target which was illuminated. This information will be used later in the intended applications (G in Fig. 12).

3.3 Types of Remote Sensing

Optical and Infrared Remote Sensing

In optical remote sensing, optical sensors detect solar radiation reflected or scattered from the arth, forming images resembling photographs taken by a camera high up in space. The wavelength region usually extends from the visible and near infrared to the short-wave infrared. Different materials such as water, soil, vegetation, buildings, and roads reflect visible and infrared light in different ways [11]. They have different colors and brightness when seen under the sun. The interpretation of optical images require the knowledge of the spectral reflectance signatures of the various materials (natural or man-made) covering the surface of the Earth. There are also infrared sensors measuring the thermal infrared radiation emitted from the Earth, from which the land or sea surface temperature can be derived (Figs. 17 and 18).

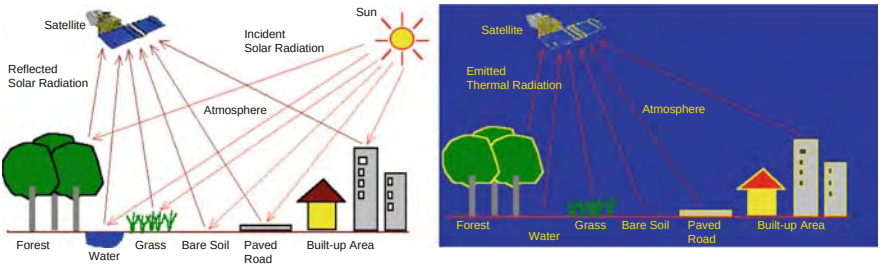


Fig. 17 Optical and infrared images

Fig. 18 Infrared image

Microwave Remote Sensing

There are some remote sensing satellites which carry passive or active microwave sensors. The active sensors emit pulses of microwave radiation to illuminate the areas to be imaged. Images of the Earth surface are formed by measuring the microwave energy scattered by the ground or sea back to the sensors. These satellites carry their own "flashlight" emitting microwaves to illuminate their targets. The images can thus be acquired day and night [11]. Microwaves have an additional advantage as they can penetrate clouds. Images can be acquired even when there are clouds covering the Earth surface. A microwave imaging system which can produce high resolution image of the Earth is the synthetic aperture radar (SAR) (Fig. 19). The intensity in a SAR image depends on the amount of microwave backscattered by the target and received by the SAR antenna. This amount depends on the roughness of the target [12].

Airborne Remote Sensing

In airborne remote sensing (Fig. 20), downward or sideward looking sensors are mounted on an aircraft to obtain images of the Earth's surface. Analog aerial photography, videography, and digital photography are commonly used in airborne remote sensing. SAR imaging is also carried out on airborne platforms. Analog photography is capable of providing high spatial resolution. The interpretation of analog aerial photographs is usually done visually by experienced analysts. The photographs may be digitized using a scanning device for computer-assisted analysis. Digital photography permits real-time transmission of the remotely sensed data to a ground station for immediate analysis [11]. In contrast to satellite remote sensing, airborne remote sensing has the capability of offering very high spatial resolution images (20 cm or less). The disadvantages are low coverage area and high cost per unit area of ground coverage. Airborne remote sensing missions are often carried out as one-time operations, whereas Earth observation satellites offer the possibility of continuous monitoring of the Earth [11].

Fig. 19 (*Left*) Basic SAR image process; (*right*) example of images taken by a SAR

Fig. 20 Airborne remote sensing concept

Spaceborne Remote Sensing

Several remote sensing satellites are currently available, providing imagery suitable for various types of applications. Each of these satellite-sensor platforms is characterized by the wavelength bands employed in image acquisition, spatial resolution of the sensor, the coverage area, and the temporal coverage, i.e., how frequent a given location on the Earth surface can be imaged by the imaging system [11]. In terms of the spatial resolution, the satellite imaging systems can be classified into: (1) low resolution systems (approx. 1 km or more), (2) medium resolution systems (approx. 100 m–1 km), (3) high resolution systems (approx. 5 m–100 m), or (4) very high resolution systems (approx. 5 m or less). In terms of the spectral regions used in data acquisition, the satellite imaging systems can be classified into: (1) optical imaging systems (include visible, near infrared, and shortwave infrared systems), (2) thermal imaging systems, or (3) synthetic aperture radar (SAR) imaging systems. Optical/thermal imaging systems can be classified according to the number of spectral bands used: (1) mono-spectral or panchromatic (single wavelength band, "black-and-white," grey-scale image) systems, (2) multi-spectral (several spectral bands) systems, (3) super-spectral (tens of spectral bands) systems, and (4) hyper-spectral (hundreds of spectral bands) systems [13].

3.4 Image Processing

Data resulting from remote sensing process can be available in an analog format. Analog images (Fig. 21), such as aerial photos, are the result of photographic imaging systems (i.e., Camera). Once the film is developed, then no more processing is required. In this case, the image data is referred to as being in an analog format. Remote sensed data can also be stored in a digital format. Using specialized tools,

Fig. 21 Example of an analog image

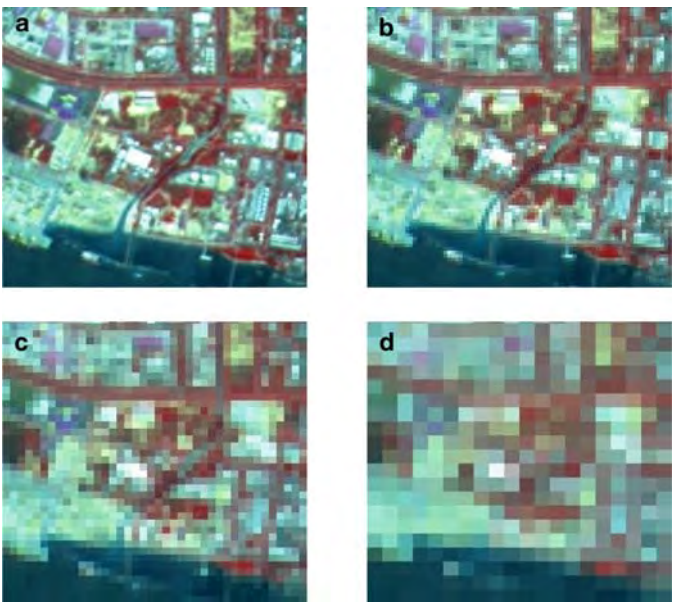


Fig. 22 Effects of the resolution on the image visibility (from left to right and from top to bottom): (**a**) 160×160 (pixel size: 10 m), (**b**) 80×80 (pixel size: 20 m), (**c**) 40×40 (pixel size: 40 m), (**d**) 20×20 (pixel size: 80 m)

the analysis and processing of this data may be carried out automatically in order to identify targets and extract information. Digital images are referred to as raster images in which the pixels are arranged in rows and columns. The number of pixels represents the image resolution. It affects the visual appearance of the area (represented by the image) and therefore, highly influences the process of analyzing and extracting information from the image [14]. For example, on the same screen size (Fig. 22), the visibility of the image is gradually damaged by reducing its resolution (which in turn affects the size of pixels). The images collected from remote sensing

Fig. 23 (*Left*) Striping (or banding) error; (*Right*) line dropouts error

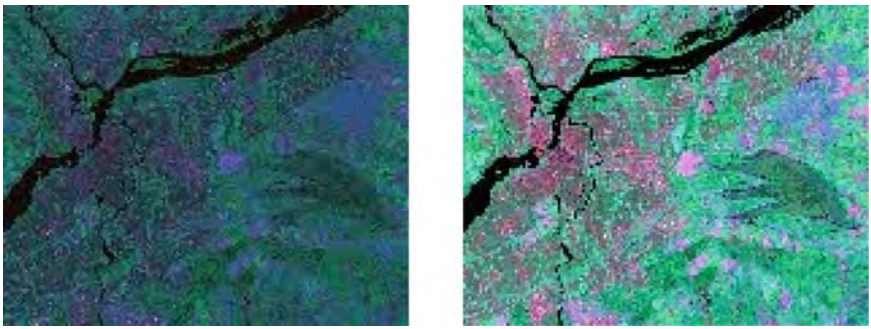


Fig. 24 Before (*left*) and after (*right*) applying the contrast stretching

devices may need some processing before starting the extraction of information. For example, since most of the recorded images are subject to distortion due to noise, some errors occur resulting in the degradation of the image quality. In order to avoid these errors, an image restoration process may be applied. Two of the more common errors that occur in multi-spectral imagery are striping (or banding) and line dropouts [15]. These errors occur in the sensor response and/or data recording and transmission and result in a shift of pixels between rows or loss of a row of pixels in the image, respectively (Fig. 23). In addition to image restoration, other techniques can be applied in order to enhance the quality of images. This can be achieved by manipulating the pixel values such that it is easier for visual interpretation. Examples of these techniques include contrast stretching and spatial filtering. Since it is common that the useful data in a digital image populates only a small portion of the available range of digital values (commonly 8 bits or 256 levels), contrast stretching involves changing the original values so that more of the available range is used. As a result, the contrast between features and their backgrounds is increased [15]. Figure 24 illustrates an example of contrast stretching where light toned areas appear

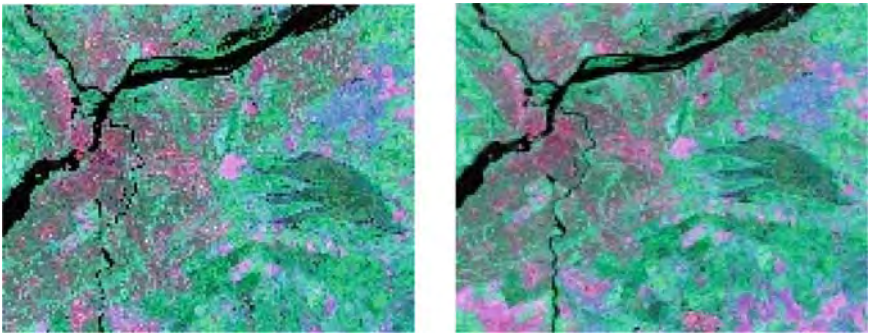


Fig. 25 (*Left*) contrast stretching; (*Right*) low-pass filter

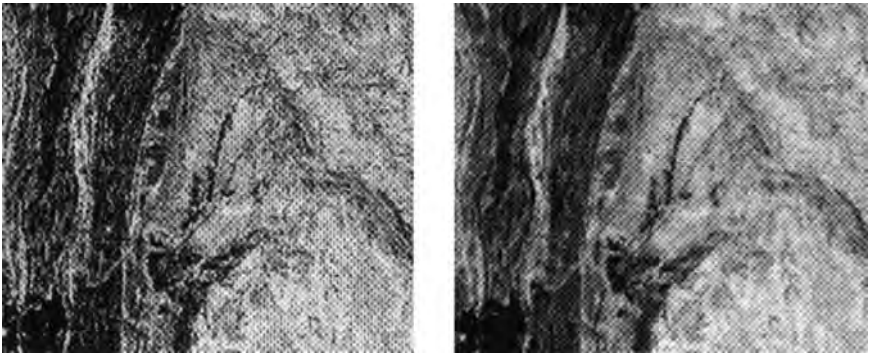


Fig. 26 (*Left*) contrast stretching, (*Right*) high-pass filter

lighter and dark areas appear darker. The quality of images can also be improved with spatial filtering techniques. Spatial filters are designed to highlight or eliminate features in an image based on their spatial frequency. For example, rapid variations in brightness levels reflect a high spatial frequency, whereas smooth areas with little variation in brightness level or tone are characterized by a low spatial frequency. There are several types of filters. Low-pass filters (Fig. 25) are used to emphasize large homogenous areas of similar tone and reduce the smaller detail. High-pass filters (Fig. 26) allow high frequency areas to pass with the resulting image having greater detail. Directional filters are designed to enhance linear features such as roads, streams, and faults. The filters can be designed to enhance features which are oriented in specific directions [14]. Enhancement can also be achieved with a density slicing technique (Fig. 27). In this technique, the grey tones in an image are divided into a number of intervals reflecting a range of digital numbers. Thus, the image is transformed into a limited number of gray or color tones. This is useful in displaying weather satellite information. Another technique for image enhancement is taking several images at different times and lighting conditions, then manipulating these images in order to produce a seamless mosaic [14]. Once the image is enhanced, the

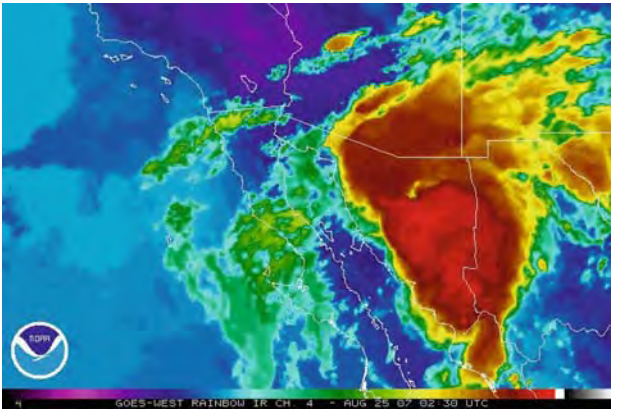


Fig. 27 Satellite infrared image with six colors: white, gray, purple, brown, red, and dark brown. Version in color available at springer.com/978-3-642-02787-1

process of extracting data can be achieved in a better way. Several techniques can be used to achieve this task. These techniques will be discussed later in this book (Chapter 12).

4 GIS: Geographic Information System

4.1 What is a GIS?

Geographic Information Systems (GISs) are being used for a wide variety of applications, such as resource management, environmental impact assessment, urban planning, cartography, criminology, history, marketing, and logistics. These systems have been described in several ways during their development and emergence as a technology. In the broadest sense, a GIS simply uses geography and technology allowing people to better understand the world and make decisions accordingly. In the strictest sense, it is a collection of computer hardware, software, and geographic data for capturing, storing, updating, manipulating, analyzing, and displaying all forms of geographically referenced information [16]. Basically, there are five key components to any functional GIS (Fig. 28): hardware, software, data, methods, and people. The hardware consists of one or more computers, printers, plotters, and networks in which a GIS operates. GIS software provides the user with the required tools and functions for the capture, storage, manipulation, and analysis of information. It includes a database software, operating system software, GIS tool (e.g., ESRI ArcGIS, ArcView, and ArcInfo), and probably network software. As the most important component, the availability of data highly influences the effectiveness. Data can be obtained from a variety of sources, such as surveying, and stored in different formats. Data is categorized and stored as a collection of thematic layers of any GDS

Fig. 28 Components of a GIS

according to its locations on the Earth. GIS takes these layers and overlays them for analysis and visualization. GIS is an extremely valuable tool for data manipulation (e.g., add, edit, or delete the attribute of data to the specification of a given project). It allows users to submit queries using the attributes associated to each layer through a database. The GIS system displays the results of queries on maps from which information can be obtained and decisions can be made accordingly [17]. An effective use of data in a given organization depends on the establishment of well-designed methods (plan and rules). The achievement of this goal remains limited unless the organization has skilled and knowledgeable people to manage, implement, and operate the system.

4.2 Data Capture and Creation

Any variable of the data source that can be located spatially can be fed into a GIS. In this case, it is important to know its location (e.g., longitude, latitude, and elevation coordinates). Data can be obtained from a variety of sources, such a GPS systems, online websites, digital images, sensors, satellites, and surveying (Fig. 29). Surveying is the technique and science of accurately determining the terrestrial or 3D space position of points and the distances and angles between them. There is a tradeoff between the quality and costs of capture and storage of data. Absolute accurate data allows for an easier interpretation of information. However, it is costly, particularly in terms of time. Data is commonly stored in a digitized format. Survey data (e.g., collected with a GPS) can be directly entered into a GIS from systems integrated to survey instruments. Once date is stored, some editing may be necessary to add attributes or remove errors. Errors may result from non-respecting topological constraints, such as representing a house at the right side of a river instead of its left side as in the real world. Errors may also appear unintentionally. For example, dirt on a scanned map might connect two lines that should not be connected [17]. The

Fig. 29 Acquisition of data with surveying

data stored in a GIS may come from different sources and concern spatial, social, or economic issues. The analysis of this data may result in the extraction of relevant knowledge to the application domain. A GIS can help in achieving this task by relating this data in spite of its heterogeneity. For example, it is able to convert existing digital information, which is not in map form, into recognizable and usable map-like form. Moreover, it allows for the application of some processing to spatial information on maps when these maps have different scales [17]. Projection is an example of processing. This fundamental component of map making allows for generating a 2D representation (on a paper medium or computer screen) from the 3D curved surface model of the Earth. The interoperability of spatial data can be better achieved by exploiting their semantics. Tools and technologies emerging from the W3C's Semantic Web Activity are useful for data integration problems in information systems. Ontologies are a key component of this semantic approach. They allow a GIS to focus on the meaning of data rather than its syntax or structure.

4.3 Types of GIS Data Models

The digital data used by a GIS represents useful real world information for human beings. This information may be of different types, such as static (e.g., buildings, lakes, mountains), dynamic (e.g., temperature, human activities in a given city), and event-based (e.g., storms, Earthquakes). GIS keeps track of this information as well the locations where they happen or exist. This goal is helped with a consistent model of the real world [18]. A data model is a set of constructs for describing and representing selected aspects of the real world in computer. Since the real world is infinitely complex, its representation requires to make difficult choices concerning what aspects to represent and how to model them. Due to the varieties of choices, several data models of the real world may exist according to users' needs

and the characteristics of the geographic space [19]. *CAD data model.* The earliest GISs were based on very simple models derived from works in the fields of image analysis, CAD, and computer cartography. In a CAD model, entities are symbolically represented as simple points, lines, and polygons. This model has three main problems: (1) it typically uses local drawing coordinates instead of real-world coordinates to represent objects, (2) the identification of individual objects is difficult since they do not have unique identifiers, and (3) it focuses on the graphical representation of objects and does not store details of any relationships between objects. *Computer cartography.* It aimed to automate the production of maps and the creation of simple thematic maps. The idea was to digitize and store paper maps on computer for subsequent printing. Like CAD systems, it is difficult with computer cartography data models to identify objects and work with object relationships. *Image model.* It is a simple model which is still used by GIS despite its limitations. It uses scanned pictures of real-world objects. It handles these pictures using rasters or grids. *Object data model.* The previous data models focus on the geometry of objects using a collection of points, lines, and polygons. They do not allow for the representation of spatial objects having large numbers of proprieties and complex relations with nearby objects. In addition, with these models, all transformations are applied to spatial objects in separate procedures making software and database development tedious and time-consuming [18]. The object data model has been proposed as a solution for these limitations. Each object contains the proprieties of a specific spatial object and several methods allowing for the handling of this object. The geometry of the spatial object is considered as an attribute of the object. Using the object model, designers are able to model relationships between objects, such as topologic relationships (mathematical relationships used to validate the geometry of vector entities, to test polygons adjacency, etc.), geographic relationships (based on geographic operators such as overlap, adjacency, and inside that determine the interaction between objects), and general relationships (rules to maintain database integrity and to enforce validation constraints). *Raster model.* It uses an array of cells (pixels) to represent the objects of the real world (Fig. 30). This array is made up of grid values with metadata about the array. This metadata typically includes the geographic coordinate of the upper-left corner of the grid, the cell size, and the number of rows and column elements. The raster model is widely used for analytical applications such as disease dispersion modeling and surface water flow analysis [19]. Raster data sets record a value for all points in the area covered which may require more storage space than representing data in a vector format that can store data only where needed. Raster data also allows easy implementation of overlay operations, which are more difficult with vector data. Non-spatial data can be represented in raster format. In this case, the cell value can store attribute information, but it can also be used as an identifier that can relate to records in another table [19]. *Vector model.* In a vector model, all lines of an area are captured as a polygon which is a series of points or vertices generally connected by straight lines (Fig. 31, left). This model needs to specify the locations of the different points that form the polygon. The vector model is widely implemented in GIS. Its popularity is due to the precise

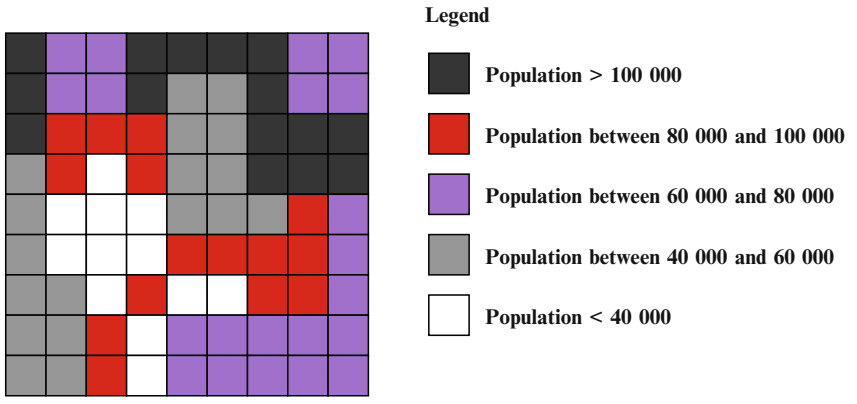


Fig. 30 Example of raster representation

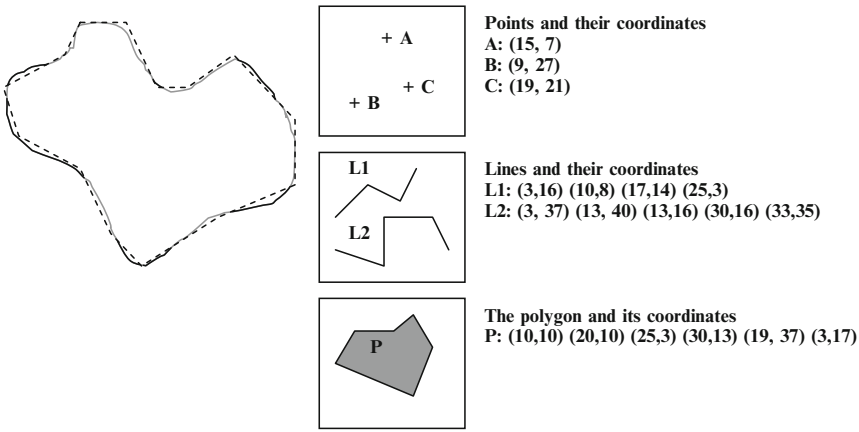


Fig. 31 (*left*) A polygon approximation of an area; (*right*) Representation of point, line, and polygon objects using the vector data model [19]

nature of its representation method, its storage efficiency, and the quality of its cartographic output [19]. The vector model represents each spatial object as a point (used for geographical features that can best be expressed by a single point reference, e.g., wells), a line (used for linear features such as rivers and roads), or a polygon (used for geographical features that cover a particular area of the Earth's surface, such as lakes and buildings) (Fig. 31, right). In the database, each of these geometries is linked to a row that describes the attributes of the related object. For example, the attributes of a river in a database may be its depth, water quality, and pollution level. Compared to raster data, vector data provide practitioners with several advantages. In addition to their easier update and maintenance, they are easy to register, scale, and re-project. This can simplify combining vector layers from different sources. Vector data are more compatible with relational database environments. Moreover,

in addition to their reduced storage space, they allow for extended analysis capability, especially for networks. For example, vector data enables practitioners to query the data for the largest port within 100 miles from a given location and the connecting road that is at least a two-lane highway. Data restructuring can be performed by a GIS in order to convert data from vector to raster and vice versa. The conversion from raster to vector can be achieved with software products like R2V, Able Vector, and AlgoLab. The conversion from vector to raster can be done with software products like Scan2CAD and TracTrix.

4.4 Spatial Analysis with GIS

Spatial analysis (Fig. 32) is an important way to evaluate, estimate, predict, interpret, and understand spatial phenomena. Relevant, and commonly hidden, knowledge/information are discovered with this process that usually involves manipulation or calculation of coordinate or attribute variables with various operators. Examples of these operators include selection, reclassification, dissolving, buffering, overlay, and cartographic modeling. In addition to the use of these operators, the definition of the relevant objects to the current study, the use of computers for analysis, the limitations and particularities of the analysis, and the representation of final results

Fig. 32 Spatial analysis of subspace close to the hydrographic network

are fundamental issues in spatial analysis. These issues include challenges related to spatial characterization, spatial dependency or autocorrelation, scaling, and sampling. In order to tackle the issues of spatial analysis, many products offer facilities like software development kits (SDKs), programming languages, or scripting facilities. In addition, many GIS products provide practitioners with built-in or optional tools for managing spatial data, computing spatial relationships (such as distance, connectivity, and directional relationships between spatial units), and visualizing both the raw data and spatial analytic results within a cartographic context. The use of these tools and facilities depends on the intended spatial analysis type that can be divided into several categories, such as spatial overlay, contiguity analysis, surface analysis, network analysis, and raster analysis. A further important aspect of geospatial analysis is visualization. It consists in the creation and manipulation of images, maps, diagrams, charts, 3D static and dynamic views and their associated tabular datasets. GIS packages increasingly encompass a range of such tools allowing for static or rotating views, animations, dynamic linking and brushing, and spatio-temporal visualizations.

Spatial Overlay

One basic way to create or identify spatial relationships is through the process of spatial overlay (Fig. 33). Spatial overlay is accomplished by joining and viewing together separate data sets that share all or part of the same area. The result of this combination is a new data set that identifies the spatial relationships. Spatial overlay

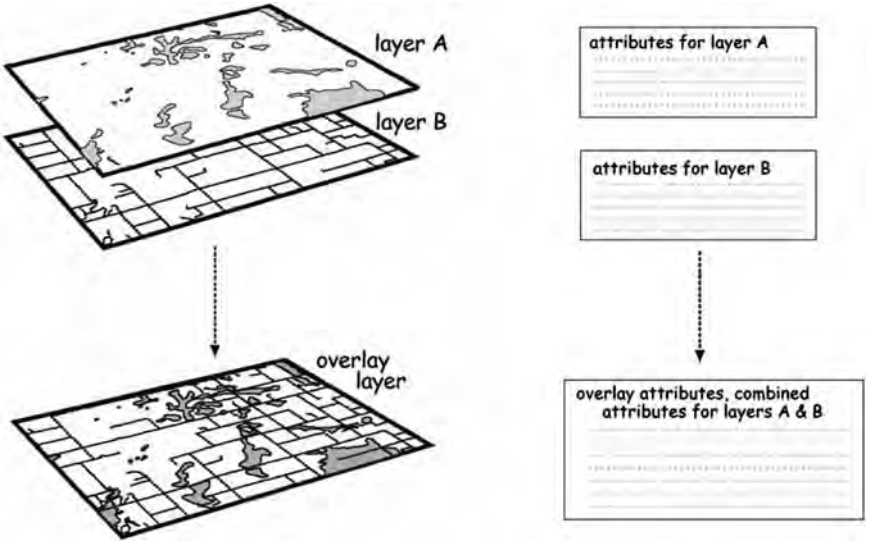


Fig. 33 Principle of spatial overlay

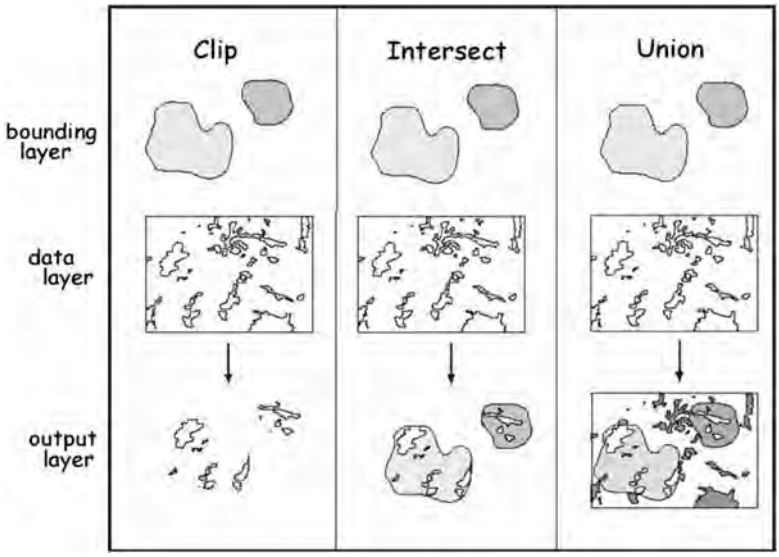


Fig. 34 Spatial overlay applied to vector data

requires that data layers use a common coordinate system [17]. Spatial overlay may be applied either to raster or vector data. On the one hand, raster overlay is typically applied to nominal or ordinal data. The process is done cell by cell resulting in the combination of the two input layers. On the other hand, vector overlay can be applied in three common ways: clip, intersection, and union (Fig. 34).

Contiguity Analysis

The term spatial relationship can be best explained through an example. Consider the question "How many lakes are in the province of Quebec?" This query is non-spatial. The answer does not require knowledge of the physical location of the lakes nor does it describe where the lakes are in relation to one another. However, a question that asks "How many lakes are in the province of Quebec having an area greater than 10 km^2 and 2 km apart?" is of a spatial nature. To answer this question, one must have the ability to determine the location of each lake, measure the distance between the lakes, and examine their attributes (e.g., surface). A GIS can easily provide the user with the right answer. This is possible since the GIS has the ability to link spatial data with information (facts and figures) about a particular feature on a map. To achieve this task, the GIS uses neighborhood analysis. This analysis is sometimes referred to as contiguity analysis. More explicitly, contiguity analysis is an analytical technique to determine whether a set of areas (polygons) are situated next to each other and to examine their interrelationships. A good GIS creates internal data structures ("topology") for finding answers rapidly. This type of

analysis enables the GIS to answer questions, such as: Which properties are next to a flooded plain? Which species have habitats in contact with a protected ecological region?

Surface Analysis

Surface analysis allows for the analysis of the properties of physical surfaces, such as gradient, aspect, and visibility. It analyzes the distribution of a variable which can be represented as the third dimension of spatial data. In addition to *X* and *Y*-coordinates of features, a *Z*-variable (e.g., elevation) is added to represent the variation in the surface. Surface analysis can achieve several operations such as slope (identifies slope or maximum rate of change, from each cell to its neighbor), aspect (identifies the steepest down-slope direction from each cell to its neighbor), hillshade (determines the hypothetical illumination of a surface for either analysis or graphical display), viewshed (identifies which cell locations can be seen from each observation point), and contour (produces an output polyline dataset) [17].

Raster Analysis

Raster analysis is widely used in environmental sciences and remote sensing. It typically means a range of actions applied to the grid cells of one or more maps (or images). It often involves filtering and/or algebraic operations that process one or more raster layers according to simple rules resulting in a new map layer. For example, raster analysis may be achieved by replacing each cell value with some combination of its neighbors' values, or computing the sum or difference of specific attribute values for each grid cell in two matching raster datasets. Descriptive statistics, such as cell counts, means, variances, maxima, minima, cumulative values, frequencies, and a number of other measures and distance computations are also often included in this type of analysis [17].

Network Analysis

Network analysis consists of the examination of the properties of natural and man-made networks. It may be used to address a wide range of practical problems, such as route selection and facility location. It also helps in understanding the behavior of flows within and around such networks. In addition, a GIS can simulate the routing of materials along a linear network. In order to represent the flow of the phenomenon more accurately, values such as slope, speed limit, or pipe diameter can be incorporated into network modeling. Network modeling is commonly employed in transportation planning, hydrology modeling, and infrastructure modeling [17].

4.5 Data Output and Cartography

Cartography is the design and production of maps or visual representations of spatial data. The vast majority of modern cartography is done with the help of computers, usually using a GIS. Traditional maps are abstractions of the real world, a sampling of important elements represented on a sheet of paper with symbols assigned to physical objects. They express a geographical reality, with different data representations and according to a specific scale and purpose. People who use maps must interpret these symbols. Powerful analysis techniques can produce high-quality maps within a short time period by using different data representations. Important elements in these maps can be emphasized with graphical display techniques such as shading [17]. Maps can also be produced in a digital form. In this case, web map servers can be used in order to facilitate their distribution through web browsers using various implementations of web-based application programming interfaces (e.g., AJAX, Java, and Flash). The process of creating (personalized) digital maps is fastidious. GIS is used to help implement this process automatically. The main problem in automating the map-creation process is the use of a single set of data to produce multiple products at a variety of scales [18]. This can be achieved with a technique known as generalization. Cartographic generalization can be defined as the science and art of exaggerating the important aspects (entities), in accordance with the purpose and scale of a particular map, with the exclusion of irrelevant details that may overload the map and confuse its user [20]. Cartographic generalization aims to produce a good map, balancing the requirements of accuracy, information content, and legibility. During this process, logical and unambiguous relationships between map objects must be maintained, while aesthetic quality is preserved. Cartographic generalization is a very complex process. In order to reduce its complexity, the overall process is often decomposed into individual sub-processes, called operators. Each operator defines a transformation that can be applied to a single spatial object, or to a group of spatial objects. Operators include simplification, elimination, aggregation, exaggeration, displacement, size reduction, and typification. For example, if we intend to generate a 1:20 k map from a 1:5 k map, a scale reduction is necessary. Due to this process, some objects are no longer visible. Useless objects are simply eliminated. The improvement of the visibility of some important objects can be achieved by exaggerating their geometrical representation. Exaggeration may result in the overlap between neighboring objects. Displacements may be applied to some objects in order to solve this overlap [18].

4.6 GIS Software

Within industry, several companies such as ESRI, Intergraph, Mapinfo, and Autodesk propose GIS tools. Governmental and military departments often use custom software, specialized products that meet specific needs, or open source products (e.g., GRASS). Open source products allow users to freely access and handle

geo-spatial information. Many open-source GIS software products are currently available. This has been motivated by the broad use of non-proprietary and open data formats such as the Shape File format for vector data and the Geotiff format for raster data. It has also been motivated by the adoption of OGC (Open Geospatial Consortium) standards including Web Map Service (WMS) and Web Feature Service (WFS). Well-known open source GIS software includes GRASS GIS, Quantum GIS, MapServer, uDig, OpenJUMP, and gvSIG. In addition to the open-source GIS software, the free access and handling was made easier with the increase of web-based providers (e.g., Google Earth, MapQuest, and Yahoo maps). These providers are taking benefit from the increasing popularity of Internet and networks that contributed in the gradual change of GIS software from being standalone to distributed. The currently available GIS software targets several topics related to spatial information capture, handling, and production. For the purpose of data creation, specialized high-end type of software are required to deal with the time-consuming task of transforming raw data into a format usable by a GIS. Many standard database and spreadsheet applications can be used to achieve this purpose. AutoCAD software can also be used for digitizing. For the purpose of management and analysis, many software products are available. Examples of professional software include ArcGIS, Smallworld, Civil Designer, Xmap, and GRASS. These products particularly help in data visualization and analysis. They can output a detailed map, image, or movie used to communicate an idea or concept with respect to a region of interest. For querying purposes, standard DBMS, such as MySQL and ArcSDE, can be used to extract information from the spatial databases. More and more databases are currently housed on servers so that they can be queried from web sites. This was particularly helped with the evolution of scripting languages (e.g., JavaScript and VBA) and GIS APIs. It was also helped with applications, such as MapInfo's MapXtreme, Intergraph's Geomedia WebMap (TM), ESRI's ArcIMS, ArcGIS Server, AutoDesk's Mapguide, SeaTrails' AtlasAlive, and the open source MapServer.

References

1. Wikipedia, Global Positioning System (2008), `http://en.wikipedia.org/wiki/GPS`
2. D.J. Johnson, Overcoming challenges to transformational space programs: the global positioning system (GPS), Analysis Center Papers (2006), http://www.analysiscenter.northropgrumman.com, Northrop Grumman Corporation
3. L. Harte, B. Levitan, Systems, Technology and Operation, GPS Quick Course Book, 2007
4. P.L.N. Raju, in *Fundamentals of GPS*, ed. by M.V.K. Sivakumar, P.S. Roy, K. Harmsen, S.K. Saha. Satellite Remote Sensing and GIS Applications in Agricultural Meteorology, 2003, pp. 121–150
5. Kowoma, Sources of Errors in GPS, From GPS-explained (2008), http://www.kowoma.de/en/gps/errors.htm
6. Office of Science and Technology (2000), http://www.ostp.govhtml/0053_2.html
7. RoseIndia, Techniques to Improve GPS Accuracy (2008), `http://www.roseindia.net/` technology/gps/techniques-to-improve-GPS-accuracy.shtml

8. D.A. Grejner-Brzezinska, C.K. Toth, Y. Yi, Photogramm. Eng. Rem. Sens. **71**(4), 377–389 (2005)
9. Y. Yi, *Direct Sensor Georeferencing* (The Ohio State University, Columbus, OH, 2007), Geodetic Science and Surveying Report, number 484, http://www.ceegs. ohio-state.edu/gsreports/
10. CCRS, Remote Sensing, Canadian center for remote sensing (2008), Tutorial course, `http://www.ccrs.nrcan.gc.ca/resource/tutor/fundam/chapter1/05_e.php`
11. S.C. Liew, Principles of remote sensing, Centre for Remote Imaging, Sensing, and Processing, National University of Singapore (1997), tutorial, http://www.crisp.nus.edu.sg/research/tutorial/rsmain.htm
12. CRISP, SAR Imaging – Frequency, Polarisation and Incident Angle, Centre for Remote Imaging, Sensing, and Processing, National University of Singapore (2008), tutorial, http://www.crisp.nus.edu.sg/~research/tutorial/freqpol.htm
13. CRISP, Spaceborne Remote Sensing, Centre for Remote Imaging, Sensing, and Processing, National University of Singapore (2001), tutorial, http://www.crisp.nus.edu.sg/research/tutorial/spacebrn.htm
14. GeoForum, Remote Sensing (2008), tutorial, http://hosting.soonet.ca/eliris/
15. T. Ouattara, R. Couture, P.T. Bobrowsky, A. Moore, Remote Sensing and Geosciences, Geological Survey of Canada, 2008, Open File, 4542
16. H.J. Miller, J. Han, GIS Fundamentals: A first text on Geographic Information Systems, 2nd edn., White Bear Lake, 2005
17. Wikipedia, Geographic Information System (2008), `http://en.wikipedia.org/wiki/GIS`
18. N. Jabeur, *A Multiagent-based Approach for On-the-fly Map Generation and Spatial Conflict Resolution* (Laval University, Canada, 2006)
19. P.A. Longley, M.F. Goodchild, D.J Maguire, D.W. Rhind, *Geographic Information Systems and Science* (Wiley, New York, 2002)
20. P.G. Hardy, Cartographic J. **35**(2), 181–189 (1998)

Computational Chemistry

Hassan Safouhi and Ahmed Bouferguene

1 Introduction

1.1 Molecular Multi-Center Integrals Over Exponential Type Functions

It is well known that in any ab initio molecular orbital (MO) calculation, the major task involves the computation of the so-called molecular multi-center integrals, namely overlap, two-, three-center nuclear attraction; hybrid, two- three- and four-center two-electron Coulomb; and exchange integrals. A great number of these integrals is required for molecular calculations based on the linear combination of atomic orbitals–molecular orbitals approximation (LCAO-MO) [1]. As the molecular system gets larger, computation of these integrals becomes one of the most difficult and time consuming steps in molecular systems calculation. Improvement of the computational methods for molecular integrals would be indispensable to a further development in computational studies of large molecular systems.

In ab initio calculations using the LCAO-MO approximation, molecular orbitals are built from a linear combination of atomic orbitals. Thus, the choice of reliable basis functions is of prime importance [2]. A good atomic orbital basis should satisfy the cusp at the origin [3] and the exponential decay at infinity [4, 5].

The most popular functions used in ab initio calculations are the so-called Gaussian type functions (GTFs) [6, 7]. With GTFs, the numerous molecular integrals can be evaluated rather easily. Unfortunately, these GTF basis functions fail to satisfy the above mathematical conditions for atomic electronic distributions. A large number of GTFs have to be used in order to achieve acceptable accuracy and this increases the computational cost.

Exponential type functions (ETFs) show the same behavior as the exact solutions of atomic or molecular Schrödinger equations satisfying Kato's conditions [8].

H. Safouhi (✉)
Campus Saint-Jean, University of Alberta, 8406, 91 Street, Edmonton, AB, Canada T6C 4G9

M.M. Gaber (ed.), *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*, DOI 10.1007/978-3-642-02788-8_8,

These functions are thus better suited to represent electron wave functions near the nucleus and at long range. This implies that a smaller number of ETFs than GTFs is needed for comparable accuracy. Unfortunately, ETF's molecular multi-center integrals are extremely difficult to evaluate accurately and rapidly. Among ETFs, Slater type functions (STFs) [9, 10] have a dominating position this is due to the fact that their analytical expression is very simple. Unfortunately the multi-center integrals over these functions turned out to be extremely difficult to evaluate. From the early days of quantum chemistry, scientists have been aware of the unfortunate reality that ETF's molecular multi-center integrals are difficult to evaluate. Despite the efforts of many scientists, including the very pioneers of quantum chemistry, no efficient algorithms were proposed for their numerical evaluation. This hindered the development of ETF-based programs for routine computations in quantum molecular structure determination. However, with the recent advances in computer technology, many researchers hope that the next generation of ab initio programs will be based on ETF's [11–14], since the numerical algorithms that were previously unusable are, nowadays, implemented for the routine usage. Therefore, it does not seem impossible to envisage that ETFs may compete with GTFs in accurate and rapid molecular calculations in the near future. Indeed, much effort is being made to develop efficient molecular algorithms for integrals over conventional ETFs (see [8, 15, 16] and references therein).

Various studies have focused on the use of B functions [17–19]. The B functions are analytically more complicated than STFs but they have much more appealing properties applicable to multi-center integral problems [18, 20], in particular, the fact that their Fourier transforms are exceptionally simple [21, 22]. Note that STFs can be expressed as finite linear combinations of B functions [18] and that the basis set of B functions is well adapted to the Fourier transform method thoroughly studied by the Steinborn group [18–30], which led to analytic expressions for all multi-center molecular integrals over B functions.

Note that the Fourier transform method, which is one of the most successful approach for the evaluation of multi-center integrals, was first shown by Prosser and Blanchard [31] and Geller [32].

The analytic expressions obtained for molecular integrals over B functions using the Fourier transform method, turned out to be extremely difficult to evaluate rapidly to a high pre-determined accuracy. This is due to the presence of two- or three-dimensional integral representations. The inner semi-infinite integrals is highly oscillatory because of the presence of spherical Bessel functions $j_\lambda(v\,x)$. When the values of λ and v are large, the accurate and fast numerical evaluation of these semi-infinite integrals becomes extremely difficult.

In previous works [33–35], we demonstrated that the use of Gauss–Laguerre quadrature is inefficient in evaluating this kind of oscillatory integral. It is possible to break up these semi-infinite integrals into infinite series of integrals. These series are slowly convergent to the accuracy required for chemically significant molecular calculations and this is why their use has been prevented. By using the epsilon algorithm of Wynn [36] or Levin's u transform [37], which are the most popular convergence accelerator applied to the molecular integrals, we can

accelerate the convergence of such infinite series, but the calculation times required to achieve sufficient accuracies are still prohibitively long. New techniques are required for a rapid and accurate numerical evaluation of molecular integrals over ETFs.

1.2 Nonlinear Transformations and Extrapolation Methods

In applied mathematics and in the numerical treatment of scientific problems, slowly convergent or divergent sequences and series and oscillatory integrals occur abundantly. Therefore, convergence accelerators and nonlinear transformation methods for accelerating the convergence of infinite series and integrals have been invented and applied to various situations. They are based on the idea of extrapolation. Their utility for enhancing and even inducing convergence has been amply demonstrated by Wynn [38] and Shanks [39]. Via sequence transformations slowly convergent and divergent sequences and series can be transformed into sequences and series with better numerical properties. Thus, they are useful for accelerating convergence of slowly convergent series and integrals. In the case of nonlinear transformations, the improvement of convergence can be remarkable. These methods form the basis of new methods for solving various problems which were unsolvable otherwise and have many applications as well [40–47].

In previous works, [33, 34, 48–57], we showed the efficiency of the nonlinear transformations D [58] and $\bar{D}$ [59, 60], in evaluating spherical Bessel integral functions of the following form $\int_0^\infty g(x)\, j_\lambda(v\,x)\,\mathrm{d}x$, where v is a real number and $g(x)$ is a nonoscillating function.

To apply these two transformations, the integrand should satisfy a linear differential equation with coefficients having asymptotic expansions in a sense of Poincaré series [61]. It is shown that under some conditions on the non-oscillatory part $g(x)$, the integrand of the above semi-infinite integral satisfies a second order linear differential equation of the form required to apply the D and $\bar{D}$ transformations. The approximations $D_n^{(2)}$ and $\bar{D}_n^{(2)}$ of the semi-infinite integral, which converge very quickly to the exact value of the integral as n becomes large, are obtained by solving sets of linear equations of order $2\,n + 1$ and $n + 1$, respectively, where the computation of the $2\,n + 1$ or $n + 1$ successive positive zeros of the integrand is necessary. This requires a large amount of CPU time, in particular when dealing with spherical Bessel integrals.

In [62, 63], we introduced an extremely powerful method which combines the S transformation [64, 65] with the nonlinear $\bar{D}$ transformation. The S transformation transforms the spherical Bessel integral functions into sine integral functions. The strong oscillations of the integrands are then considerably reduced and this helps greatly the extrapolation method. As it is well known, the numerical integration of oscillatory integrands is difficult, especially when the oscillatory part is a spherical

Bessel function and not a simple trigonometric function [66, 67]. We demonstrated that the obtained integrands with the sine function satisfy second order linear differential equations of the form required to apply $\bar{D}$.

Using properties of the sine function, in particular the fact that its zeros are equidistant, allowed the use of Cramer's rule for computing the approximations $S\bar{D}_n^{(2)}$ of semi-infinite integrals. The computation of a method to solve linear systems as well as the computation of the successive positive zeros of the integrands are avoided. This result led to the development of a very simple and rapid algorithm for accurate numerical evaluation of the semi-infinite integrals of interest. Recurrence relations were developed for a better control of the degree of accuracy and to further reduce the calculation times.

1.3 Extrapolation Methods and Molecular Integrals over ETFs

In [33, 48, 49], it is demonstrated that molecular integrals over B functions satisfy the conditions required to apply the D and $\bar{D}$ transformations for improving convergence of oscillatory integrals, and it is shown that these transformations are highly efficient compared with alternatives using classical techniques. The calculation times were considerably reduced (by factor 4–6 over classical methods). In [34, 50, 52, 55], new extrapolation techniques specially suited for ETFs molecular integrals were developed in order to simplify the application of $\bar{D}$, when improving convergence of complicated ETFs molecular integrals, especially when dealing with the notorious four-center two-electron Coulomb and exchange integrals. The numerical results obtained showed a further improvement in accuracy and a substantial gain in calculation times (factor 5–8 over $\bar{D}$). In [62, 64, 65], it is shown that the $S\bar{D}$ that combines the S and $\bar{D}$ transformations can also be applied to the semi-infinite integrals involved in the analytic expressions of all molecular integrals over B functions. Highly efficient and rapid algorithms are now developed for all molecular integrals based on the aforementioned methods. Numerical analysis was performed and it showed that in certain instances the values of the integrands tend towards 0 or $+\infty$. This causes overflow errors, and the program returns the message NaN, which means not a number. In such a situation, we have developed alternate expressions for the approximations of the semi-infinite integrals that avoids the overflow altogether. The obtained expression can also be computed recursively.

The numerical results section shows that the combination S and $\bar{D}$ methods with the recurrence relations leads to highly efficient and rapid algorithms for the numerical evaluation of molecular multi-center integrals over B functions and over STFs. A complete optimized software package for all molecular multi-center integrals based on extrapolation methods will soon be available.

2 General Definitions and Properties

The spherical Bessel function $j_l(x)$ is defined by [68]:

$$j_l(x) \;=\; (-1)^l\, x^l \left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^l \left(\frac{\sin(x)}{x}\right). \tag{1}$$

The spherical Bessel function and its first derivative satisfy the following recurrence relations [68]:

$$\begin{cases} x\; j_{l-1}(x) \;+\; x\; j_{l+1}(x) & = (2l+1)\; j_l(x) \\ x\; j_{l-1}(x) \;-\; (l+1)\; j_l(x) & = x\; j_l'(x). \end{cases} \tag{2}$$

For the following, we write $j^n_{l+\frac{1}{2}}$ with $n = 1, 2, \ldots$ for the successive positive zeros of $j_l(x)$. $j^0_{l+\frac{1}{2}}$ are assumed to be 0.

The reduced Bessel function $\hat{k}_{n+\frac{1}{2}}(z)$ is defined by [17, 19]:

$$\hat{k}_{n+\frac{1}{2}}(z) = \sqrt{\frac{2}{\pi}}(z)^{n+\frac{1}{2}} K_{n+\frac{1}{2}}(z) \tag{3}$$

$$= z^n\, e^{-z} \sum_{j=0}^{n} \frac{(n+j)!}{j!\,(n-j)!} \frac{1}{(2\,z)^j}, \tag{4}$$

where $K_{n+\frac{1}{2}}$ stands for the modified Bessel function of the second kind [68].

A useful property satisfied by $\hat{k}_{n+\frac{1}{2}}(z)$ is given by:

$$\left(\frac{\mathrm{d}}{z\mathrm{d}z}\right)^m \left[\frac{\hat{k}_{n+\frac{1}{2}}(z)}{z^{2n+1}}\right] \;=\; (-1)^m\, \frac{\hat{k}_{n+m+\frac{1}{2}}(z)}{z^{2(n+m)+1}}. \tag{5}$$

If we let the function γ be defined by:

$$\gamma(\kappa, x) = \sqrt{\tau + \kappa\, x^2},$$

then with the help of the Leibnitz formula and the fact that $\dfrac{\mathrm{d}}{\mathrm{d}x} = \dfrac{\mathrm{d}z}{\mathrm{d}x}\dfrac{\mathrm{d}}{\mathrm{d}z}$, one can show that if $n_\gamma = 2\nu$, then for $j \in \mathbb{N}$:

$$\left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^j \left[\frac{\hat{k}_\nu[\gamma(\kappa,x)]}{[\gamma(\kappa,x)]^{n_\gamma}}\right] = (-1)^j\, \kappa^j\, \frac{\hat{k}_{\nu+j}\,[\gamma(\kappa,x)]}{[\gamma(\kappa,x)]^{2(\nu+j)}}, \tag{6}$$

and for $n_\gamma < 2\nu$, we obtain:

$$\left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^j \left[\frac{\hat{k}_\nu[\gamma(\kappa,x)]}{[\gamma(\kappa,x)]^{n_\gamma}}\right] = \frac{\kappa^j}{[\gamma(\kappa,x)]^{n_\gamma+2j}} \sum_{i=0}^{j} \binom{j}{i} \frac{(-1)^{j-i}\,(2\nu-n_\gamma)!!}{(2\nu-n_\gamma-2i)!!} \hat{k}_{\nu+j-i}\left[\gamma(\kappa,x)\right]. \tag{7}$$

The reduced Bessel functions satisfy the recurrence relation [17]:

$$\hat{k}_{n+\frac{1}{2}}(z) \;=\; (2n-1)\;\hat{k}_{n-\frac{1}{2}}(z) + z^2\,\hat{k}_{n-\frac{3}{2}}(z). \tag{8}$$

The B functions are defined as follows [18, 19]:

$$B^m_{n,l}(\zeta,\vec{r}) = \frac{(\zeta r)^l}{2^{n+l}(n+l)!}\;\widehat{k}_{n-\frac{1}{2}}(\zeta r)\;Y^m_l(\theta_{\vec{r}},\varphi_{\vec{r}}), \tag{9}$$

where $Y^m_l(\theta,\varphi)$ stands for the surface spherical harmonic and is defined explicitly using the Condon–Shortley phase convention as follows [81]:

$$Y^m_l(\theta,\varphi) \;=\; i^{m+|m|}\left[\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}\right]^{\frac{1}{2}} P^{|m|}_l(\cos\theta)\,e^{im\varphi}, \tag{10}$$

$P^m_l(x)$ is the associated Legendre polynomial of l degree and m order:

$$P^m_l(x) \;=\; \left(1-x^2\right)^{m/2}\left(\frac{\mathrm{d}}{\mathrm{d}x}\right)^{l+m}\left[\frac{\left(x^2-1\right)^l}{2^l\,l!}\right]. \tag{11}$$

The B function can only be used as a basis functions of atomic orbitals if $n \in \mathbb{N}$ holds. For $-l \le n \le 0$, a B function is singular at the origin, and if $n = -l-\nu$ with $\nu \in \mathbb{N}$ holds, then a B function is no longer a function in the sense of classical analysis but a derivation of the three-dimensional Dirac delta function [69].

The Fourier transform $\bar{B}^m_{n,l}(\zeta,\vec{p})$ of $B^m_{n,l}(\zeta,\vec{r})$ is given by [22]:

$$\bar{B}^m_{n,l}(\zeta,\vec{p}) \;=\; \sqrt{\frac{2}{\pi}}\,\zeta^{2n+l-1}\,\frac{(-i\,|p|)^l}{(\zeta^2+|p|^2)^{n+l+1}}\,Y^m_l(\theta_{\vec{p}},\varphi_{\vec{p}}). \tag{12}$$

The normalized STFs are defined by [10]:

$$\chi^m_{n,l}(\zeta,\vec{r}) \;=\; \mathcal{N}(\zeta,n)\,r^{n-1}\,e^{-\zeta r}\,Y^m_l(\theta_{\vec{r}},\varphi_{\vec{r}}), \tag{13}$$

where $N(\zeta,n)$ stands for the normalization factor and it is given by:

$$\mathcal{N}(\zeta,n) \;=\; \sqrt{\frac{(2\zeta)^{2n+1}}{(2n)!}}. \tag{14}$$

The Gaunt coefficients are defined as [70–72]:

$$< l_1 m_1 | l_2 m_2 | l_3 m_3 > = \int_{\theta=0}^{\pi} \int_{\varphi=0}^{2\pi} [Y_{l_1}^{m_1}(\theta,\varphi)]^* \, Y_{l_2}^{m_2}(\theta,\varphi) \, Y_{l_3}^{m_3}(\theta,\varphi) \, \sin(\theta) \, \mathrm{d}\theta \, \mathrm{d}\varphi. \tag{15}$$

These coefficients linearize the product of two spherical harmonics:

$$[Y_{l_1}^{m_1}(\theta,\varphi)]^* Y_{l_2}^{m_2}(\theta,\varphi) = \sum_{l=l_{\min},2}^{l_1+l_2} < l_2 m_2 | l_1 m_1 | l m_2 - m_1 > Y_l^{m_2-m_1}(\theta,\varphi), \tag{16}$$

where the subscript $l = l_{\min,2}$ in the summation symbol implies that the summation index l runs in steps of 2 from $l_{\min}$ to $l_1 + l_2$. The constant $l_{\min}$ is given by [72]:

$$l_{\min} = \begin{cases} \max(|l_1 - l_2|, |m_2 - m_1|), \\ \quad \text{if } \; l_1 + l_2 + \max(|l_1 - l_2|, |m_2 - m_1|) \; \text{ is } \; \text{even} \\ \max(|l_1 - l_2|, |m_2 - m_1|) + 1, \\ \quad \text{if } \; l_1 + l_2 + \max(|l_1 - l_2|, |m_2 - m_1|) \; \text{ is } \; \text{odd}. \end{cases} \tag{17}$$

STFs can be expressed as finite linear combinations of B functions [18]:

$$\chi_{n,l}^{m}(\zeta, \vec{r}) \; = \; \frac{\mathcal{N}(\zeta, n)}{\zeta^{n-1}} \sum_{p=\tilde{p}}^{n-l} \frac{(-1)^{n-l-p} \; 2^{2p+2l-n} \; (l+p)!}{(2p-n+l)! \; (n-l-p)!} \, B_{p,l}^{m}(\zeta, \vec{r}), \tag{18}$$

where:

$$\tilde{p} = \begin{cases} \dfrac{n-l}{2} & \text{if} \quad n-l \quad \text{is even} \\ \dfrac{n-l+1}{2} & \text{if} \quad n-l \quad \text{is odd}. \end{cases} \tag{19}$$

The Fourier integral representation of the Coulomb operator $\dfrac{1}{|\vec{r}|}$ is given by [73]:

$$\frac{1}{|\vec{r}|} \; = \; \frac{1}{2\pi^2} \int_{\vec{k}} \frac{e^{-i \, \vec{k} \, . \, \vec{r}}}{k^2} \, \mathrm{d}\vec{k}. \tag{20}$$

The hypergeometric function is given by [68]:

$${}_2F_1(\alpha, \beta; \gamma; x) \; = \; \sum_{r=0}^{+\infty} \frac{(\alpha)_r \; (\beta)_r \; x^r}{(\gamma)_r \; r!}, \tag{21}$$

where $(\alpha)_n$ represents the Pochhammer symbol [68]:

$$(\alpha)_n = \begin{cases} (\alpha)_n = 1 \quad \text{if} \quad n = 0 \\ (\alpha)_n = \alpha\,(\alpha+1)\,(\alpha+2)\ldots(\alpha+n-1) = \dfrac{\Gamma(\alpha+n)}{\Gamma(\alpha)} \quad \text{if} \quad n \leq -\alpha \\ (\alpha)_n = 0 \quad \text{if} \quad n \geq -\alpha+1, \end{cases} \tag{22}$$

where Γ stands for the Gamma function [68]. For $n \in \mathbb{N}$:

$$\Gamma(n+1) = n! \quad \text{and} \quad \Gamma\left(n+\frac{1}{2}\right) = \frac{(2n)!}{2^{2n}\,n!}\sqrt{\pi}. \tag{23}$$

The infinite series (21) converge only for $|x| < 1$, and they converge quite slowly if $|x|$ is slightly less than one. The corresponding functions, nevertheless, are defined in a much larger subset of the complex plane, including the case $|x| > 1$. Convergence problems of this kind can often be overcome by using nonlinear sequence transformations [74].

Note that if α or β in the infinite series (21) is a negative integer, then the hypergeometric function will be reduced to a finite sum.

For the following, we define $A^{(\gamma)}$ for certain γ, as the set of infinitely differentiable functions $p(x)$, which have asymptotic expansions in inverse powers of x as $x \to +\infty$, of the form:

$$p(x) \sim x^{\gamma}\left(a_0 + \frac{a_1}{x} + \frac{a_2}{x^2} + \ldots\right). \tag{24}$$

We denote $\tilde{A}^{(\gamma)}$ for some γ, the set of functions $p(x)$ such that:

$$p(x) \in A^{(\gamma)} \quad \text{and} \quad \lim_{x\to+\infty} x^{-\gamma}\,p(x) \neq 0. \tag{25}$$

Thus, $p \in \tilde{A}^{(\gamma)}$ has an asymptotic expansion in inverse powers of x as $x \to +\infty$ of the form given by (24) with $a_0 \neq 0$.

We define the functional $\alpha_0(p)$ by $\alpha_0(p) = a_0 = \lim_{x\to+\infty} x^{-\gamma}\,p(x)$.

Lemma 1. *Let $p(x)$ be in $A^{(\gamma)}$ for some γ. Then,*

1. *If $\gamma \neq 0$ then $p'(x) \in A^{(\gamma-1)}$, otherwise $p'(x) \in A^{(\gamma-2)}$*
2. *If $q(x) \in A^{(\delta)}$ then $p(x)\,q(x) \in A^{(\gamma+\delta)}$*
3. *$\forall\, k \in \mathbb{R}$, $x^k\,p(x) \in A^{(\gamma+k)}$*
4. *If $q(x) \in A^{(\delta)}$ and $\gamma - \delta \geq 0$ then the function $p(x)+q(x) \in A^{(\gamma)}$*
5. *For $m > 0$ an integer, $p^m(x) \in A^{(m\,\gamma)}$*
6. *The function $1/p(x) \in A^{(-\gamma)}$*

The proof of Lemma 1 follows from properties of asymptotic expansions in inverse powers of x.

3 Molecular Integrals and the Fourier Transform

Let A, B, C, and D be four arbitrary points of the euclidian space $\mathcal{E}_3$, while O is the origin of the fixed coordinate system.

3.1 Three-Center Nuclear Attraction Integrals

The three-center nuclear attraction integrals over STFs are given by:

$$\mathcal{I}_{\tilde{n}_1,l_1,m_1}^{\tilde{n}_2,l_2,m_2} = \int_{\vec{R}} \left[\chi_{\tilde{n}_1,l_1}^{m_1}\left(\zeta_1, \vec{R} - \overrightarrow{OA}\right)\right]^* \frac{1}{|\vec{R} - \overrightarrow{OC}|} \chi_{\tilde{n}_2,l_2}^{m_2}\left(\zeta_2, \vec{R} - \overrightarrow{OB}\right) \mathrm{d}\vec{R}, \tag{26}$$

where $\tilde{n}_1$ and $\tilde{n}_2$ stand for the principal quantum numbers.

By performing a translation of vector $\overrightarrow{OA}$, we can rewrite $\mathcal{I}_{\tilde{n}_1,l_1,m_1}^{\tilde{n}_2,l_2,m_2}$ as:

$$\mathcal{I}_{\tilde{n}_1,l_1,m_1}^{\tilde{n}_2,l_2,m_2} = \int_{\vec{r}} \left[\chi_{\tilde{n}_1,l_1}^{m_1}(\zeta_1, \vec{r})\right]^* \frac{1}{|\vec{r} - \vec{R}_1|} \chi_{\tilde{n}_2,l_2}^{m_2}\left(\zeta_2, \vec{r} - \vec{R}_2\right) \mathrm{d}\vec{r}, \tag{27}$$

where $\vec{r} = \vec{R} - \overrightarrow{OA}$, $\vec{R}_1 = \overrightarrow{AC}$, and $\vec{R}_2 = \overrightarrow{AB}$.

By using (18), we can express $\mathcal{I}_{\tilde{n}_1,l_1,m_1}^{\tilde{n}_2,l_2,m_2}$ as a finite linear combination of integrals $\widetilde{\mathcal{I}}_{n_1,l_1,m_1}^{n_2,l_2,m_2}$ involving B functions:

$$\widetilde{\mathcal{I}}_{n_1,l_1,m_1}^{n_2,l_2,m_2} = \int_{\vec{r}} \left[B_{n_1,l_1}^{m_1}(\zeta_1, \vec{r})\right]^* \frac{1}{|\vec{r} - \vec{R}_1|} B_{n_2,l_2}^{m_2}\left(\zeta_2, \vec{r} - \vec{R}_2\right) \mathrm{d}\vec{r}. \tag{28}$$

By substituting Fourier integral representation of the Coulomb operator (20) in (28), we obtain:

$$\widetilde{\mathcal{I}}_{n_1,l_1,m_1}^{n_2,l_2,m_2} = \frac{1}{2\pi^2} \int_{\vec{x}} \frac{e^{i\,\vec{x}.\vec{R}_1}}{x^2} \left\langle B_{n_1,l_1}^{m_1}(\zeta_1, \vec{r}) \left| e^{-i\,\vec{x}.\vec{r}} \right| B_{n_2,l_2}^{m_2}\left(\zeta_2, \vec{r} - \vec{R}_2\right) \right\rangle_{\vec{r}} \mathrm{d}\vec{x}. \tag{29}$$

3.2 Hybrid and Three-Center Two-Electron Coulomb Integrals

Three-center two-electron Coulomb integral over STFs is given by:

$$\mathcal{K}_{n_1 l_1 m_1, n_3 l_3 m_3}^{n_2 l_2 m_2, n_4 l_4 m_4} = \int_{\vec{R},\vec{R}'} \left[\chi_{n_1,l_1}^{m_1}\left(\zeta_1, \vec{R} - \overrightarrow{OA}\right)\right]^* \left[\chi_{n_3,l_3}^{m_3}\left(\zeta_3, \vec{R}' - \overrightarrow{OB}\right)\right]^* \frac{1}{|\vec{R} - \vec{R}'|} \times \chi_{n_2,l_2}^{m_2}\left(\zeta_2, \vec{R} - \overrightarrow{OA}\right) \chi_{n_4,l_4}^{m_4}\left(\zeta_4, \vec{R}' - \overrightarrow{OC}\right) \mathrm{d}\vec{R}\,\mathrm{d}\vec{R}'. \tag{30}$$

The hybrid integral, $\mathcal{H}^{n_2l_2m_2,n_4l_4m_4}_{n_1l_1m_1,n_3l_3m_3}$, corresponds to the case where $B = A$. By performing a translation of vector $\overrightarrow{OA}$, we can rewrite $\mathcal{K}^{n_2l_2m_2,n_4l_4m_4}_{n_1l_1m_1,n_3l_3m_3}$ as:

$$\mathcal{K}^{n_2l_2m_2,n_4l_4m_4}_{n_1l_1m_1,n_3l_3m_3} = \int_{\vec{r},\vec{r}'} \left[\chi^{m_1}_{n_1,l_1}(\zeta_1,\vec{r})\right]^* \left[\chi^{m_3}_{n_3,l_3}\left(\zeta_3,\vec{r}\,' - \left(\vec{R}_3 - \vec{R}_4\right)\right)\right]^* \times \frac{1}{|\vec{r} - \vec{r}\,' - \vec{R}_4|} \times \chi^{m_2}_{n_2,l_2}(\zeta_2,\vec{r})\ \chi^{m_4}_{n_4,l_4}(\zeta_4,\vec{r}\,')\ \mathrm{d}\vec{r}\,\mathrm{d}\vec{r}\,', \tag{31}$$

where $\vec{r} = \vec{R} - \overrightarrow{OA}$, $\vec{r}\,' = \vec{R}' - \overrightarrow{OC}$, $\vec{R}_3 = \overrightarrow{AB}$, and $\vec{R}_4 = \overrightarrow{AC}$ and for hybrid integral, $\vec{R}_3 = \overrightarrow{AB} = \vec{O}$.

By using (18), we can express $\mathcal{K}^{n_2l_2m_2,n_4l_4m_4}_{n_1l_1m_1,n_3l_3m_3}$ and $\mathcal{H}^{n_2l_2m_2,n_4l_4m_4}_{n_1l_1m_1,n_3l_3m_3}$ as finite linear combinations of integrals involving B functions. These integrals over B functions are given by:

$$\widetilde{\mathcal{K}}^{n_2l_2m_2,n_4l_4m_4}_{n_1l_1m_1,n_3l_3m_3} = \int_{\vec{r},\vec{r}'} \left[B^{m_1}_{n_1,l_1}(\zeta_1,\vec{r})\right]^* \left[B^{m_3}_{n_3,l_3}\left(\zeta_3,\vec{r}\,' - \left(\vec{R}_3 - \vec{R}_4\right)\right)\right]^* \times \frac{1}{|\vec{r} - \vec{r}\,' - \vec{R}_4|} \times B^{m_2}_{n_2,l_2}(\zeta_2,\vec{r})\ B^{m_4}_{n_4,l_4}(\zeta_4,\vec{r}\,')\ \mathrm{d}\vec{r}\,\mathrm{d}\vec{r}\,'. \tag{32}$$

By substituting the Fourier integral representation of the Coulomb operator in (32), we obtain:

$$\widetilde{\mathcal{K}}^{n_2l_2m_2,n_4l_4m_4}_{n_1l_1m_1,n_3l_3m_3} = \frac{1}{2\pi^2}\int_{\vec{x}} \frac{e^{i\,\vec{x}.\overrightarrow{R_4}}}{x^2} \left\langle B^{m_1}_{n_1,l_1}(\zeta_1,\vec{r}) \left| e^{-i\,\vec{x}.\vec{r}} \right| B^{m_2}_{n_2,l_2}(\zeta_2,\vec{r}) \right\rangle_{\vec{r}} \times \left\langle B^{m_4}_{n_4,l_4}(\zeta_4,\vec{r}\,') \left| e^{-i\,\vec{x}.\vec{r}\,'} \right| B^{m_3}_{n_3,l_3}\left(\zeta_3,\vec{r}\,' - (\overrightarrow{R_3} - \overrightarrow{R_4})\right) \right\rangle^*_{\vec{r}\,'} d\,\vec{x}. \tag{33}$$

In the term $T_1 = \left\langle B^{m_1}_{n_1l_1}(\zeta_1\vec{r}) \left| e^{-i\,\vec{x}.\vec{r}} \right| B^{m_2}_{n_2l_2}(\zeta_2\vec{r}) \right\rangle_{\vec{r}}$ involved in the above expression, the two B functions are centered on the same point and therefore the radial part of their product has an analytical expression which can easily be obtained using (9) and (4). Consequently, T_1 has an analytic expression, which is given by:

$$T_1 = \left[2^{n_1+l_1+n_2+l_2}(n_1+l_1)!(n_2+l_2)!\right]^{-1} \zeta_1^{l_1}\zeta_2^{l_2}\sqrt{\frac{\pi}{2x}} \times \sum_{l=l_{min},2}^{l_{max}} (-i)^l\ < l_1m_1|lm_1 - m_2|l_2m_2 > [Y_l^{m_1-m_2}(\theta_{\vec{x}},\varphi_{\vec{x}})]^* \times \sum_{k=2}^{n_1+n_2}\sum_{i=k_1}^{k_2} \left[\frac{(2n_1-i-1)!\,(2n_2-k+i-1)!\,\zeta_1^{i-1}\,\zeta_2^{k-i-1}}{(i-1)!\,(n_1-i)!\,(k-i-1)!\,(n_2-k+i)!\,2^{n_1+n_2-k}}\right]$$

$$\times \frac{\left[\frac{x}{2\zeta_s}\right]^{l+\frac{1}{2}} \Gamma(k+l_1+l_2+l+1)}{\zeta_s^{k+l_1+l_2+\frac{1}{2}} \Gamma(l+\frac{3}{2})} \left[1+\frac{x^2}{\zeta_s^2}\right]^{-k-l_1-l_2}$$
$$\times {}_2F_1\left(\frac{l-k-l_1-l_2+1}{2}, \frac{l-k-l_1-l_2}{2}+1; l+\frac{3}{2}; -\frac{x^2}{\zeta_s^2}\right) \tag{34}$$

where $k_1 = \max(1, k-n_2)$, $k_2 = \min(n_1, k-1)$, and $\zeta_s = \zeta_1 + \zeta_2$.

One of the arguments of the hypergeometric function $\frac{\eta}{2} = \frac{l-k-l_1-l_2+1}{2}$ or $\frac{l-k-l_1-l_2}{2}+1 = \frac{\eta+1}{2}$ is a negative integer. By using (22), one can easily show that the hypergeometric series involved in the above equation is reduced to a finite expansion:

$${}_2F_1\left(\frac{\eta}{2}, \frac{\eta+1}{2}; l+\frac{3}{2}; -\frac{x^2}{\zeta_s^2}\right) = \sum_{r=0}^{\eta'} (-1)^r \frac{(\frac{\eta}{2})_r \, (\frac{\eta+1}{2})_r \, x^{2r}}{(l+\frac{3}{2})_r \, r! \, \zeta_s^{2r}}, \tag{35}$$

where $\eta' = -\frac{\eta}{2}$ if η is even, otherwise $\eta' = -\frac{\eta+1}{2}$.

3.3 Four-Center Two-Electron Coulomb Integrals

Four-center two-electron Coulomb integrals over STFs, which are the most difficult type of integrals occuring in molecular structure calculations, are given by:

$$\mathcal{J}^{n_2 l_2 m_2, n_4 l_4 m_4}_{n_1 l_1 m_1, n_3 l_3 m_3} = \int_{\vec{R}, \vec{R}'} \left[\chi^{m_1}_{n_1,l_1}\left(\zeta_1, \vec{R} - \overrightarrow{OA}\right)\right]^* \left[\chi^{m_3}_{n_3,l_3}\left(\zeta_3, \vec{R}' - \overrightarrow{OC}\right)\right]^*$$
$$\times \frac{1}{\left|\vec{R} - \vec{R}'\right|} \times \chi^{m_2}_{n_2,l_2}\left(\zeta_2, \vec{R} - \overrightarrow{OB}\right) \chi^{m_4}_{n_4,l_4}$$
$$\times \left(\zeta_4, \vec{R}' - \overrightarrow{OD}\right) \mathrm{d}\vec{R} \, \mathrm{d}\vec{R}', \tag{36}$$

By using (18), one can express the four-center two-electron Coulomb integrals $\mathcal{J}^{\tilde{n}_2 l_2 m_2, \tilde{n}_4 l_4 m_4}_{\tilde{n}_1 l_1 m_1, \tilde{n}_3 l_3 m_3}$ over STFs (36) as finite linear combinations of integrals over B functions:

$$\widetilde{\mathcal{J}}^{n_2 l_2 m_2, n_4 l_4 m_4}_{n_1 l_1 m_1, n_3 l_3 m_3} = \int_{\vec{R}, \vec{R}'} \left[B^{m_1}_{n_1,l_1}\left(\zeta_1, \vec{R} - \overrightarrow{OA}\right)\right]^* \left[B^{m_3}_{n_3,l_3}\left(\zeta_3, \vec{R}' - \overrightarrow{OC}\right)\right]^*$$
$$\times \frac{1}{\left|\vec{R} - \vec{R}'\right|} \times B^{m_2}_{n_2,l_2}\left(\zeta_2, \vec{R} - \overrightarrow{OB}\right) B^{m_4}_{n_4,l_4}$$
$$\times \left(\zeta_4, \vec{R}' - \overrightarrow{OD}\right) \mathrm{d}\vec{R} \, \mathrm{d}\vec{R}'. \tag{37}$$

By substituting the integral representation of the Coulomb operator in the above expression after performing a translation of vector $\overrightarrow{OA}$ and $\overrightarrow{OD}$, we obtain:

$$\mathcal{J}^{n_2 l_2 m_2, n_4 l_4 m_4}_{n_1 l_1 m_1, n_3 l_3 m_3} = \frac{1}{2\pi^2} \int \frac{e^{i\,\vec{x}.\vec{R}_{41}}}{x^2} \left\langle B^{m_1}_{n_1,l_1}(\zeta_1, \vec{r}) \left| e^{-i\,\vec{x}.\vec{r}} \right| B^{m_2}_{n_2,l_2}\left(\zeta_2, \vec{r} - \vec{R}_{21}\right) \right\rangle_{\vec{r}} \times \left\langle B^{m_4}_{n_4,l_4}\left(\zeta_4, \vec{r}\,'\right) \left| e^{-i\,\vec{x}.\vec{r}\,'} \right| B^{m_3}_{n_3,l_3}\left(\zeta_3, \vec{r}\,' - \vec{R}_{34}\right) \right\rangle^{*}_{\vec{r}\,'} \mathrm{d}\vec{x}, \quad (38)$$

where $\vec{R}_1 = \overrightarrow{OA}$, $\vec{R}_2 = \overrightarrow{OB}$, $\vec{R}_3 = \overrightarrow{OC}$, $\vec{R}_4 = \overrightarrow{OD}$, $\vec{r} = \vec{R} - \vec{R}_1$, $\vec{r}\,' = \vec{R}\,' - \vec{R}_4$, and $\vec{R}_{ij} = \vec{R}_i - \vec{R}_j$.

In the case of three-center two-electron exchange integrals, $\vec{R}_1 = \vec{R}_3$ and in the case of two-center two-electron exchange integrals, $\vec{R}_1 = \vec{R}_3$ and $\vec{R}_2 = \vec{R}_4$.

3.4 Analytic Development

The Fourier transformation method led to an analytic expression for the terms [23, 24]:

$$\left\langle B^{m_i}_{n_i,l_i}(\zeta_i, \vec{r}) \left| e^{-i\,\vec{x}.\vec{r}} \right| B^{m_j}_{n_j,l_j}\left(\zeta_j, \vec{r} - \vec{R}\right) \right\rangle_{\vec{r}}.$$

This result led to the development of analytic expressions for all molecular multi-center integrals over B functions and over STFs.

The obtained analytic expressions turned out to be extremely difficult, because of the presence of highly oscillatory spherical Bessel integral functions. In the case of the molecular integrals under consideration, the analytic expressions are given by:

$$\begin{aligned}
\widetilde{\mathcal{I}}^{n_2,l_2,m_2}_{n_1,l_1,m_1} &= 8\,(4\,\pi)^2\,(-1)^{l_1+l_2}\,(2l_1+1)!! \\
&\times (2l_2+1)!!\,\frac{(n_1+l_1+n_2+l_2+1)!\zeta_1^{2n_1+l_1-1}\zeta_2^{2n_2+l_2-1}}{(n_1+l_1)!(n_2+l_2)!} \\
&\times \sum_{l'_1=0}^{l_1} \sum_{m'_1=-l'_1}^{l'_1} (i)^{l_1+l'_1} \frac{< l_1 m_1 | l'_1 m'_1 | l_1 - l'_1 m_1 - m'_1 >}{(2l'_1+1)!![2(l_1-l'_1)+1]!!} \\
&\times \sum_{l'_2=0}^{l_2} \sum_{m'_2=-l'_2}^{l'_2} (i)^{l_2+l'_2} (-1)^{l'_2} \frac{< l_2 m_2 | l'_2 m'_2 | l_2 - l'_2 m_2 - m'_2 >}{(2l'_2+1)!![2(l_2-l'_2)+1]!!} \\
&\times \sum_{l=l'_{\min},2}^{l'_2+l'_1} < l'_2 m'_2 | l'_1 m'_1 | l m'_2 - m'_1 > \; R_2^l\, Y_l^{m'_2-m'_1}(\theta_{\vec{R}_2}, \varphi_{\vec{R}_2}) \\
&\times \sum_{l=l''_{\min},2}^{l_2-l'_2+l_1-l'_1} (-i)^{\lambda} < l_2 - l'_2 m_2 - m'_2 | l_1 - l'_1 m_1 - m'_1 | \lambda \mu >
\end{aligned}$$

$$\times \sum_{j=0}^{\Delta l} \binom{\Delta l}{j} \frac{(-1)^j}{2^{n_1+n_2+l_1+l_2-j+1}(n_1+n_2+l_1+l_2-j+1)!}$$
$$\times \int_{s=0}^{1} s^{n_2+l_2+l_1-l_1'} (1-s)^{n_1+l_1+l_2-l_2'} Y_\lambda^\mu(\theta_{\vec{v}}, \varphi_{\vec{v}})$$
$$\times \left[\int_{x=0}^{+\infty} x^{n_x} \frac{\hat{k}_\nu[R_2\gamma(s,x)]}{[\gamma(s,x)]^{n_\gamma}} j_\lambda(\nu x)\, dx \right] ds, \tag{39}$$

where:

$$[\gamma(s,x)]^2 = (1-s)\xi_1^2 + s\xi_2^2 + s(1-s)x^2$$
$$\vec{v} = (1-s)\vec{R}_2 - \vec{R}_1$$
$$n_\gamma = 2(n_1+l_1+n_2+l_2) - (l_1'+l_2') - l + 1$$
$$\nu = n_1+n_2+l_1+l_2-l-j+\tfrac{1}{2}$$
$$\mu = (m_2-m_2') - (m_1-m_1')$$
$$n_x = l_1 - l_1' + l_2 - l_2'$$
$$\Delta l = [(l_1'+l_2'-l)/2]$$

ν and R_2 stand for the modulus of $\vec{v}$ and $\vec{R}_2$, respectively.

$$\widetilde{\mathcal{K}}_{n_1 l_1 m_1, n_3 l_3 m_3}^{n_2 l_2 m_2, n_4 l_4 m_4} =$$
$$\frac{(8\pi)^3\sqrt{\pi}\xi_1^{l_1}\xi_2^{l_2}\xi_3^{2n_3+l_3-1}\xi_4^{2n_4+l_4-1}(2l_3+1)!!(2l_4+1)!!(n_3+l_3+n_4+l_4+1)!}{2^{l_1+l_2+1}(n_1+l_1)!(n_2+l_2)!(n_3+l_3)!(n_4+l_4)!}$$
$$\times \sum_{l=l_{\min},2}^{l_1+l_2} \frac{(-i)^l}{2^{2n_1+2n_2+l}} < l_2 m_2 | l_1 m_1 | l m_2 - m_1 >$$
$$\times \sum_{k=2}^{n_1+n_2} \sum_{i=k_1}^{k_2} \left[\frac{2^k (2n_1-i-1)!(2n_2-k+i-1)!\,\xi_1^{i-1}\xi_2^{k-i-1}}{(i-1)!(n_1-i)!(k-i-1)!(n_2-k+i)!} \right]$$
$$\times \sum_{l_4'=0}^{l_4} \sum_{m_4'=-l_4'}^{l_4'} i^{l_4+l_4'} (-1)^{l_4'} \frac{< l_4 m_4 | l_4 - l_4' m_4 - m_4' | l_4' m_4' >}{(2l_4'+1)!![2(l_4-l_4')+1]!!}$$
$$\times \sum_{l_3'=0}^{l_3} \sum_{m_2'=-l_3'}^{l_3'} i^{l_3+l_3'} \frac{< l_3 m_3 | l_3 - l_3' m_3 - m_3' | l_3' m_3' >}{(2l_3'+1)!![2(l_3-l_3')+1]!!}$$
$$\times \sum_{l'=l'_{min,2}}^{l_3'+l_4'} < l_4' m_4' | l_3' m_3' | l' m_4' - m_3' > R_{34}^{l'} Y_{l'}^{m_3'-m_4'}(\theta_{\vec{R}_{34}}, \varphi_{\vec{R}_{34}})$$
$$\times \sum_{l_{34}=l''_{min,2}}^{l_3-l_3'+l_4-l_4'} < l_3 - l_3' m_3 - m_3' | l_4 - l_4' m_4 - m_4' | l_{34} m_{34} >$$

$$\times \sum_{\lambda=\lambda_{min,2}}^{l+l_{34}} i^{\lambda} < lm_2 - m_1 | l_{34}(m_3 - m_3') - (m_4 - m_4')|\lambda\mu >$$

$$\times \sum_{j=0}^{\Delta l} \binom{\Delta l}{j} \frac{(-1)^j \zeta_s^{n_k - l - 1} \Gamma(k + l_1 + l_2 + l + 1)}{2^{n_3+n_4+l_3+l_4-j+1}(n_3 + n_4 + l_3 + l_4 - j + 1)! \Gamma(l + \frac{3}{2})}$$

$$\times \sum_{r=0}^{\eta'} \frac{(-1)^r \, (\frac{\eta}{2})_r \, (\frac{\eta+1}{2})_r}{(l + \frac{3}{2})_r \, r! \, \zeta_s^{2r}} \int_{s=0}^{1} s^{n_3+l_3+l_4-l_4'} \, (1-s)^{n_4+l_4+l_3-l_3'} \, Y_{\lambda}^{\mu}(\theta_{\vec{v}}, \varphi_{\vec{v}})$$

$$\times \left[\int_0^{+\infty} \frac{x^{n_x}}{\left[\zeta_s^2 + x^2\right]^{n_k}} \frac{\widehat{k}_{\nu}[R_{34}\gamma(s,x)]}{[\gamma(s,x)]^{n_\gamma}} \, j_{\lambda}(v\,x) \, \mathrm{d}x \right] \mathrm{d}s, \tag{40}$$

where:

$$k_1 = \max(1, k - n_2), k_2 = \min(n_1, k - 1).$$
$$n_x = l_3 - l_3' + l_4 - l_4' + 2r + l, n_k = k + l_1 + l_2$$
$$n_\gamma = 2(n_3 + l_3 + n_4 + l_4) - (l_3' + l_4') - l' + 1$$
$$\eta = l - k - l_1 - l_2 + 1, \Delta l = \frac{l_3 + l_4 - l'}{2}$$
$$\vec{v} = (1-s)\left(\vec{R}_3 - \vec{R}_4\right) + \vec{R_4} = (1-s)\,\vec{R}_{34} + \vec{R}_4$$
$$\mu = (m_2 - m_1) - (m_3 - m_3') + (m_4 - m_4')$$
$$[\gamma(s,x)]^2 = (1-s)\zeta_4^2 + s\zeta_3^2 + s(1-s)x^2$$
$$\nu = n_3 + n_4 + l_3 + l_4 - l' - j + \tfrac{1}{2}$$
$$m_{34} = (m_3 - m_3') - (m_4 - m_4')$$
v and R_{34} stand for the modulus of $\vec{v}$ and $\vec{R}_{34}$, respectively.

The analytic expression of hybrid integrals over B functions can be obtained by replacing R_3 by 0 in the above equation.

$$\widetilde{\mathcal{J}}^{n_2 l_2 m_2, n_4 l_4 m_4}_{n_1 l_1 m_1, n_3 l_3 m_3} = 8(4\pi)^5 (2l_1+1)!!(2l_2+1)!! \frac{(n_1 + l_1 + n_2 + l_2 + 1)!}{(n_1 + l_1)!(n_2 + l_2)!}$$

$$\times (-1)^{l_1+l_2} (2l_3+1)!!(2l_4+1)!! \frac{(n_3 + l_3 + n_4 + l_4 + 1)!}{(n_3 + l_3)!(n_4 + l_4)!} \zeta_1^{2n_1+l_1-1} \zeta_2^{2n_2+l_2-1}$$

$$\times \zeta_3^{2n_3+l_3-1} \zeta_4^{2n_4+l_4-1} \sum_{l_1'=0}^{l_1} \sum_{m_1'=\mu_{11}}^{\mu_{12}} i^{l_1+l_1'} \frac{\langle l_1 m_1 | l_1' m_1' | l_1 - l_1' m_1 - m_1' \rangle}{(2l_1'+1)!![2(l_1 - l_1') + 1]!!}$$

$$\times \sum_{l_2'=0}^{l_2} \sum_{m_2'=\mu_{21}}^{\mu_{22}} i^{l_2+l_2'} (-1)^{l_2'} \frac{\langle l_2 m_2 | l_2' m_2' | l_2 - l_2' m_2 - m_2' \rangle}{(2l_2'+1)!![2(l_2 - l_2') + 1]!!}$$

$$\times \sum_{l_3'=0}^{l_3} \sum_{m_1'=\mu_{31}}^{\mu_{32}} i^{l_3+l_3'} \frac{\langle l_3 m_3 | l_3' m_3' | l_3 - l_3' m_3 - m_3' \rangle}{(2l_3'+1)!![2(l_3 - l_3') + 1]!!}$$

$$
\begin{aligned}
&\times \sum_{l'_4=0}^{l_4} \sum_{m'_4=\mu_{41}}^{\mu_{42}} i^{l_4+l'_4}(-1)^{l'_4} \frac{\langle l_4 m_4 | l'_4 m'_4 | l_4 - l'_4 m_4 - m'_4 \rangle}{(2l'_4+1)!![2(l_4-l'_4)+1]!!} \\
&\times \sum_{l=l_{1,min},2}^{l'_1+l'_2} \langle l'_2 m'_2 | l'_1 m'_1 | l m' 2 - m'_1 \rangle \, R^l_{21} \, Y_l^{m'_2-m'_1}\left(\theta_{\vec{R}_{21}}, \varphi_{\vec{R}_{21}}\right) \\
&\times \sum_{l_{12}=l'_{1,min},2}^{l_1-l'_1+l_2-l'_2} \langle l_2 - l'_2 m_2 - m'_2 | l_1 - l'_1 m_1 - m'_1 | l_{12} m_{21} \rangle \\
&\times \sum_{l'=l_{2,min},2}^{l'_3+l'_4} \langle l'_4 m'_4 | l'_3 m'_3 | l' m' 4 - m'_3 \rangle R^{l'}_{34} \, Y_{l'}^{m'_4-m'_3}\left(\theta_{\vec{R}_{34}}, \varphi_{\vec{R}_{34}}\right) \\
&\times \sum_{l_{34}=l'_{2,min},2}^{l_3-l'_3+l_4-l'_4} \langle l_4 - l'_4 m_4 - m'_4 | l_3 - l'_3 m_3 - m'_3 | l_{34} m_{43} \rangle \\
&\times \sum_{l=l''_{min},2}^{l_{12}+l_{34}} (-i)^{\lambda} \langle l_{12} m_{21} | l_{34} m_{43} | \lambda \mu \rangle \\
&\times \sum_{j_{12}=0}^{\Delta l_{12}} \sum_{j_{34}=0}^{\Delta l_{34}} \binom{\Delta l_{12}}{j_{12}} \binom{\Delta l_{34}}{j_{34}} \frac{(-1)^{j_{12}+j_{34}}}{2^{\nu_1+\nu_2+l+l'+1}(\nu_1+\frac{1}{2}+l)!(\nu_2+\frac{1}{2}+l')!} \\
&\times \int_{s=0}^{1} \frac{s^{n_2+l_2+l_1}(1-s)^{n_1+l_1+l_2}}{s^{l'_1}(1-s)^{l'_2}} \int_{t=0}^{1} \frac{t^{n_4+l_4+l_3}(1-t)^{n_3+l_3+l_4}}{t^{l'_3}(1-t)^{l'_4}} \, Y_{\lambda}^{m_2-\mu}(\theta_{\vec{v}}, \varphi_{\vec{v}}) \\
&\times \left[\int_{x=0}^{+\infty} x^{n_x} \frac{\widehat{k}_{\nu_1}[R_{21}\gamma_{12}(s,x)]}{[\gamma_{12}(s,x)]^{n_{\gamma_{12}}}} \frac{\hat{k}_{\nu_2}[R_{34}\gamma_{34}(t,x)]}{[\gamma_{34}(t,x)]^{n_{\gamma_{34}}}} j_{\lambda}(vx) \, \mathrm{d}x \right] \mathrm{d}t \, \mathrm{d}s, \qquad (41)
\end{aligned}
$$

$$
\begin{aligned}
&\mu = (m_2 - m'_2) - (m_1 - m'_1) + (m_4 - m'_4) - (m_3 - m'_3) \\
&n_{\gamma_{12}} = 2(n_1 + l_1 + n_2 + l_2) - (l'_1 + l'_2) - l + 1 \\
&n_{\gamma_{34}} = 2(n_3 + l_3 + n_4 + l_4) - (l'_3 + l'_4) - l' + 1 \\
&\mu_{1i} = \max(-l'_i, m_i - l_i + l'_i), \text{ for } i = 1,2,3,4 \\
&\mu_{2i} = \min(l_i, m_i + l_i - l'_i), \text{ for } i = 1,2,3,4 \\
&[\gamma_{12}(s,x)]^2 = (1-s)\zeta_1^2 + s\zeta_2^2 + s(1-s)x^2 \\
&[\gamma_{34}(t,x)]^2 = (1-t)\zeta_3^2 + t\zeta_4^2 + t(1-t)x^2 \\
&n_x = l_1 - l'_1 + l_2 - l'_2 + l_3 - l'_3 + l_4 - l'_4 \\
&\nu_1 = n_1 + n_2 + l_1 + l_2 - l - j_{12} + \tfrac{1}{2} \\
&\nu_2 = n_3 + n_4 + l_3 + l_4 - l' - j_{34} + \tfrac{1}{2} \\
&\vec{v} = (1-s)\vec{R}_{21} + (1-t)\vec{R}_{43} - \vec{R}_{41} \\
&\Delta l_{12} = \frac{l'_1 + l'_2 - l}{2}, \, \Delta l_{34} = \frac{l'_3 + l'_4 - l'}{2} \\
&m_{ij} = m_i - m'_i - (m_j - m'_j).
\end{aligned}
$$

In the following, the semi-infinite integrals in (39), (40), and (41) will be referred to as $\mathcal{I}(s)$, $\mathcal{K}(s)$ and $\mathcal{J}(s,t)$ respectively and their corresponding integrands will be referred to as $f_{\mathcal{I}}(x)$, $f_{\mathcal{K}}(x)$ and $f_{\mathcal{J}}(x)$ respectively.

The integrands $f_{\mathcal{I}}(x)$, $f_{\mathcal{K}}(x)$, and $f_{\mathcal{J}}(x)$ oscillate strongly especially for large values of λ and ν as it can be seen from Figs. 1–3. The non-oscillating part of the integrands are exponentially decreasing functions. Unfortunately, these exponentially decreasing functions become constants when s and t are close to 0 or 1. Indeed, if we make the substitution $s = 0$ or 1, the function $\gamma(s,x)$ or $\gamma_{ij}(s,x)$ becomes a constant. Thus, in the regions where s and t are close to 0 or 1, the strong oscillations of the spherical Bessel function cannot be reduced or damped by the exponentially decreasing functions $\hat{k}_\nu$. In this case, the asymptotic behavior of the integrand cannot

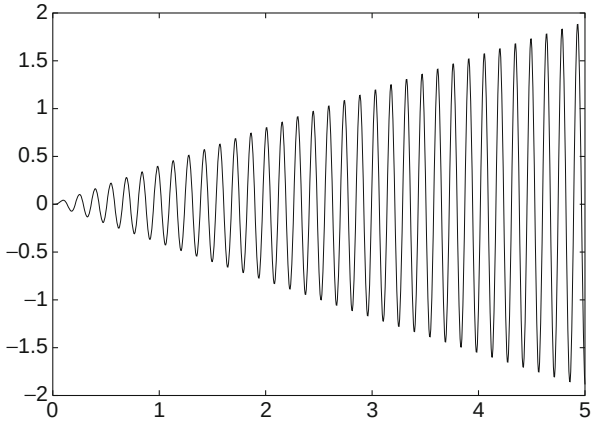


Fig. 1 The integrand $f_{\mathcal{I}}(x)$ of $\mathcal{I}$. $s = 2.40$, $\nu = 9/2$, $n_\gamma = 3$, $n_x = 2$, $\lambda = 2$, $R_1 = 45.$, $R_2 = 2.$, $\zeta_1 = 1.5$ and $\zeta_2 = 1$

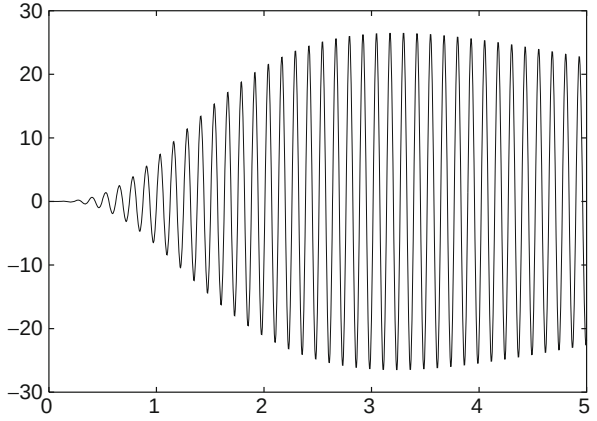


Fig. 2 The integrand $f_{\mathcal{K}}(x)$ of $\mathcal{K}(s)$. $s = 0.998$, $\nu = 13/2$, $n_\gamma = 5$, $n_k = 2$, $n_x = 4$, $\lambda = 4$, $R_{34} = 2.0$, $\zeta_s = 2.0$ and $\zeta_3 = \zeta_4 = 1.0$. ($v = 49.996$)

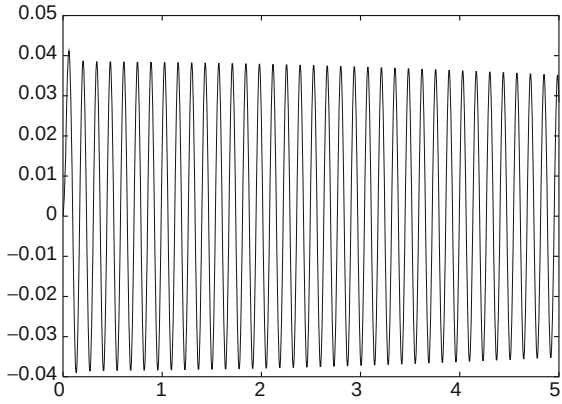


Fig. 3 The integrand $f_{\mathcal{J}}(x)$ of $\mathcal{J}(s,t)$. $s = .999$, $t = .005$, $\nu_{12} = \nu_{34} = 5/2$, $n_{\gamma_{12}} = n_{\gamma_{34}} = 1$, $n_x = \lambda = 1$, $\zeta_1 = \zeta_4 = 1.0$, $\zeta_2 = 1.5$, $\zeta_3 = 2.0$, $R_{12} = 2.0$ and $R_{34} = 1.0$

be represented by a function of the form $e^{-\alpha x}\, g(x)$ where $g(x)$ is not a rapidly oscillating function. This is why, in these regions, Gauss-Laguerre quadrature even to high orders gives inaccurate results and presents severe numerical difficulties.

The semi infinite x integrals can be transformed into an infinite series as follows:

$$\mathcal{I}(s) = \sum_{n=0}^{+\infty} \int_{j_{\lambda,\nu}^n}^{j_{\lambda,\nu}^{n+1}} F(x)\,\mathrm{d}x, \tag{42}$$

where $j_{\lambda,\nu}^0 = 0$ and $j_{\lambda,\nu}^n = j_{l+\frac{1}{2}}^n/\nu$ $(\nu \neq 0)$ for $n = 1, 2, \ldots$, which are the successive positive zeros of $j_\lambda(\nu x)$ and where F stands for $f_{\mathcal{I}}(x)$, $f_{\mathcal{K}}(x)$, or $f_{\mathcal{J}}(x)$.

The infinite series (42) is convergent and alternating; therefore, the sum of N first terms, for N sufficiently large, gives a good approximation of the semi-infinite integrals. Unfortunately, the use of this approach is much time consuming, in particular for large values of ν. We can notice that when the value of ν is large and when s is close to 0 and 1, we need to sum a very large number of terms of the infinite series in order to obtain values with 10–12 correct digits (see numerical tables).

4 Nonlinear Transformations and Extrapolation Methods

4.1 The Nonlinear $\bar{D}$ Transformation

Theorem 1. *[58, 59] Let $f(x)$ be integrable on $[0, +\infty[$ and satisfy a linear differential equation of order m of the form:*

$$f(x) = \sum_{k=1}^{m} p_k(x)\, f^{(k)}(x), \tag{43}$$

where $p_k \in A^{(i_k)}$ *with* $i_k \leq k$ *for* $k = 1, 2, \ldots, m$.

Let also:

$$\lim_{x \to +\infty} p_k^{(i-1)}(x)\, f^{(k-i)}(x) = 0,$$

for $i \leq k \leq m$ *and* $1 \leq i \leq m$.

If for every integer $l \geq -1$*:*

$$\sum_{k=1}^{m} l(l-1)\ldots(l-k+1)\, p_{k,0} \neq 1,$$

where:

$$p_{k,0} = \lim_{x \to +\infty} x^{-k} p_k(x), \quad 1 \leq k \leq m,$$

then as $x \to +\infty$*:*

$$\int_x^{+\infty} f(t)\,\mathrm{d}t \sim \sum_{k=0}^{m-1} f^{(k)}(x)\, x^{j_k} \left(\beta_{0,k} + \frac{\beta_{1,k}}{x} + \frac{\beta_{2,k}}{x^2} + \ldots\right), \tag{44}$$

where:

$$j_k \leq \max(i_{k+1}, i_{k+2} - 1, \ldots, i_m - m + k + 1),\ k = 0, 1, \ldots, m-1.$$

■

The approximation $\bar{D}_n^{(m)}$ of $\int_0^{\infty} f(t)\,\mathrm{d}t$, using the $\bar{D}$ transformation is given by [59]:

$$\bar{D}_n^{(m)} = \int_0^{x_l} f(t)dt + \sum_{k=1}^{m-1} f^{(k)}(x_l) x_l^{\sigma_k} \sum_{i=0}^{n-1} \frac{\bar{\beta}_{k,i}}{x_l^i}, \quad l = 0, 1, \ldots, n(m-1), \tag{45}$$

where $\bar{D}_n^{(m)}$ and $\bar{\beta}_{k,i}$ for $k = 0, 1, \ldots, m-1$ and $i = 0, 1, \ldots, n-1$ are the unknowns of the linear system. The x_l for $l = 0, 1, \ldots$ are leading positive zeros of $f(x)$. σ_k for $k = 0, 1, \ldots, m-1$ are the minima of $k+1$ and s_k where s_k is the largest of the integers s for which $\lim_{x \to +\infty} x^s f^{(k)}(x) = 0$.

4.2 The S Transformation

Theorem 2. *Let* $f(x)$ *be a function of the form:*

$$f(x) = g(x) j_\lambda(x), \tag{46}$$

with $\lambda \geq 1$ *and where* $g(x)$ *is in* $\mathcal{C}^2$ $([0,+\infty[)$*, which is the set of twice continuously differentiable functions defined on the half-open interval* $[0,+\infty[$*. If for all* $l = 0, 1, \ldots, \lambda - 1$*:*

$$\lim_{x\to 0} x^{l-\lambda+1} \left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^l \left(x^{\lambda-1} g(x)\right) j_{\lambda-1-l}(x) = 0, \tag{47}$$

$$\lim_{x\to +\infty} \left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^l \left(x^{\lambda-1} g(x)\right) j_{\lambda-1-l}(x) = 0, \tag{48}$$

then:

$$\int_0^{+\infty} f(x)\mathrm{d}x = \int_0^{+\infty} \left[\left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^\lambda \left(x^{\lambda-1}\, g(x)\right)\right] \sin(x)\,\mathrm{d}x. \tag{49}$$

$$= \frac{1}{\nu^{\lambda+1}} \sum_{n=0}^{+\infty} \int_{n\,\pi/\nu}^{(n+1)\,\pi/\nu} \left[\left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^\lambda \left(x^{\lambda-1}\, g(x)\right)\right] \sin(\nu\, x)\,\mathrm{d}x. \tag{50}$$

■

Proof. Let us consider:

$$\int_0^{+\infty} f(x)\,\mathrm{d}x \;=\; \int_0^{+\infty} g(x)\; j_\lambda(x)\,\mathrm{d}x.$$

By replacing the spherical Bessel function $j_\lambda(x)$ by its analytical expression given by (1), we obtain:

$$\int_0^{+\infty} f(x)\mathrm{d}x = (-1)^\lambda \int_0^{+\infty} x^\lambda\, g(x) \left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^\lambda j_0(x)\,\mathrm{d}x. \tag{51}$$

Integrating by parts the right-hand side of (51), we obtain:

$$\int_0^{+\infty} f(x)\mathrm{d}x = (-1)^\lambda \left[x^{\lambda-1}\, g(x) \left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^{\lambda-1} j_0(x)\right]_0^{+\infty}$$
$$+ (-1)^{\lambda-1} \int_0^{+\infty} \left[\left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)\left(x^{\lambda-1}\, g(x)\right)\right] \left[\left(\frac{d}{x\mathrm{d}x}\right)^{\lambda-1} j_0(x)\right] x\,\mathrm{d}x. \tag{52}$$

By integrating by parts until all the derivatives of $j_0(x)$ disappear in the last term on the right-hand side of (52), one can obtain:

$$\int_0^{+\infty} f(x)\mathrm{d}x = \left[\sum_{l=0}^{\lambda-1} (-1)^{\lambda+l} \left(\left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^l \left(x^{\lambda-1} g(x)\right)\right) \left(\left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^{\lambda-1-l} j_0(x)\right)\right]_0^{+\infty}$$
$$+ \int_0^{+\infty} \left[\left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^\lambda \left(x^{\lambda-1}\, g(x)\right)\right] j_0(x)\, x\, dx. \tag{53}$$

Using (1) and replacing $j_0(x)$ by $\dfrac{\sin(x)}{x}$, the above equation can be rewritten as:

$$\int_0^{+\infty} f(x)\mathrm{d}x = -\left[\sum_{l=0}^{\lambda-1} x^{l-\lambda+1}\left(\left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^l\left(x^{\lambda-1}g(x)\right)\right) j_{\lambda-1-l}(x)\right]_0^{+\infty} + \int_0^{+\infty}\left[\left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^\lambda\left(x^{\lambda-1}\,g(x)\right)\right]\sin(x)\mathrm{d}x. \tag{54}$$

From conditions (47) and (48) of Theorem 1, it follows that:

$$\left[\sum_{l=0}^{\lambda-1} x^{l-\lambda+1}\left(\left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^l\left(x^{\lambda-1}g(x)\right)\right) j_{\lambda-1-l}(x)\right]_0^{+\infty} = 0, \tag{55}$$

and therefore (54) can be rewritten as:

$$\int_0^{+\infty} f(x)\mathrm{d}x = \int_0^{+\infty}\left[\left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^\lambda\left(x^{\lambda-1}\,g(x)\right)\right]\sin(x)\,dx. \tag{56}$$

Theorem 3. *If $f(x)$ is a function of the form given by (46) and satisfying all the conditions of Theorem 2 and if $g(x)$ belongs to $A^{(\gamma)}$ (24) such that $\gamma-\lambda-1<0$, then the approximation of $\displaystyle\int_0^{+\infty} f(x)\mathrm{d}x$ using the nonlinear $\bar{D}$ transformation, is given by:*

$$S\bar{D}_n^{(2,j)} = \frac{\displaystyle\sum_{i=0}^{n+1}\binom{n+1}{i}(1+i+j)^{n-1}\,F(x_{i+j})\,/\left[x_{i+j}^2 G(x_{i+j})\right]}{\displaystyle\sum_{i=0}^{n+1}\binom{n+1}{i}(1+i+j)^{n-1}\,/\left[x_{i+j}^2 G(x_{i+j})\right]}, \tag{57}$$

where $x_l=(l+1)\pi$ for $l=0,1,\ldots$, $F(x)=\displaystyle\int_0^x G(t)\,\sin(t)\,\mathrm{d}t$ and the function $G(x)$ is given by:

$$G(x)=\left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^\lambda\left(x^{\lambda-1}g(x)\right). \tag{58}$$

■

Proof. Using the fact that the integrand f satisfies the conditions of Theorem 2, one can obtain according to (56):

$$\int_0^{+\infty} f(x)\mathrm{d}x = \int_0^{+\infty} G(x)\,\sin(x)\,\mathrm{d}x. \tag{59}$$

To apply the nonlinear $\bar{D}$ transformation [59, 60] for improving convergence of the semi-infinite integral in the RHS of (59), the integrand which will be referred to as $\widetilde{f}(x)$, should satisfy a linear differential equation with coefficients having asymptotic expansions in inverse powers of their argument x as $x \to +\infty$, in a sense of Poincaré series [61]. The coefficients should also satisfy two more conditions [59, 60] that we will list later.

The sine function satisfies a second order linear differential equation given by:

$$\sin(x) \;=\; -\sin''(x). \tag{60}$$

By replacing the sine function in the above differential equation by $\dfrac{\widetilde{f}(x)}{G(x)}$, one can obtain after some algebraic operations, the following differential equation, which is satisfied by the integrand $\widetilde{f}(x)$:

$$\widetilde{f}(x) \;=\; q_1(x)\,\widetilde{f}'(x) + q_2(x)\,\widetilde{f}''(x), \tag{61}$$

where the coefficients $q_1(x)$ and $q_2(x)$ are defined by:

$$\begin{cases} q_1(x) = \dfrac{2\,G(x)\,G'(x)}{G(x)^2 - G(x)\,G''(x) + 2\,G'(x)^2} \\ q_2(x) = \dfrac{-G(x)^2}{G(x)^2 - G(x)\,G''(x) + 2\,G'(x)^2}. \end{cases} \tag{62}$$

Using the fact that $g(x) \in A^{(\gamma)}$ and using the properties of asymptotic expansions given by Lemma 1, one can easily show that the function $G(x) \in A^{(\gamma-\lambda-1)}$.

From (62), it follows by using the properties given by Lemma 1 that:

$$\begin{cases} q_1(x) \in A^{(-1)} \\ q_2(x) \in A^{(0)}. \end{cases} \tag{63}$$

To apply the nonlinear $\bar{D}$ transformation, the coefficients should satisfy two more conditions given by:

1. For every integer $l \geq -1$:

$$\sum_{k=1}^{m} l(l-1)\ldots(l-k+1)q_{k,0} \;\neq\; 1,$$

where:

$$q_{k,0} = \lim_{x \to +\inf} x^{-k}\, q_k(x), \quad 1 \leq k \leq 2.$$

2. For $i \leq k \leq 2$, $1 \leq i \leq 2$:

$$\lim_{x \to +\infty} q_k^{(i-1)}(x)\, \widetilde{f}^{(k-i)}(x) \;= 0.$$

Using the fact that $q_k \in A^{(m)}$ where $m < k$, one can easily show that $q_{k,0} = 0$ for $k = 1, 2$. From this it follows that the condition 1 is satisfied.

Using the facts that $G(x) \in A^{(\gamma-\lambda-1)}$, $\gamma - \lambda - 1 < 0$, $q_1 \in A^{(-1)}$ and $q_2 \in A^{(0)}$, one can easily show that the condition 2 is also satisfied.

All the conditions of the applicability of the nonlinear $\bar{D}$ transformation are now shown to be satisfied by the new integrand $\widetilde{f}(x)$.

The approximation of $\displaystyle\int_0^{+\infty} \widetilde{f}(x)\mathrm{d}x$ using $\bar{D}$ is given by:

$$S\bar{D}_n^{(2,j)} = \int_0^{x_{l+j}} \widetilde{f}(x)\mathrm{d}x + (-1)^{l+j+1} G(x_{l+j}) x_{l+j}^2 \sum_{i=0}^{n-1} \frac{\bar{\beta}_{1,i}}{x_{l+j}^i}, \quad l = 0, 1, \ldots, n, \tag{64}$$

where $j = 0, 1, 2, \ldots$ and $x_l = (l+1)\,\pi$ for $l = 0, 1, \ldots$, which are the successive positive zeros of the sine function.

Using properties of the sine function, in particular the fact that its zeros are equidistant allowed the use of Cramer's rules as demonstrated by Sidi [59] for the computation of the approximation $S\bar{D}_n^{(2,j)}$ of the semi-infinite integral, and this leads to (57).

■

Now, we shall state a theorem, fully demonstrated in [64], and which concern the application of the S and $\bar{D}$ transformations for improving convergence of semi-infinite spherical Bessel integrals, whose integrands are of the following form:

$$f(x) = g(x)\, j_\lambda(x), \tag{65}$$

where $g(x)$ is in $\mathcal{C}^2$ ($[0, +\infty[$) and of the form $g(x) = h(x)\, e^{\varphi(x)}$.

Note that the demonstration of the following theorem differs from the one we presented in this present manuscript, because of the fact that the conditions on the g function are not similar.

Theorem 4. *[64] Let $f(x)$ be a function of the form:*

$$f(x) = g(x)\, j_\lambda(x),$$

where $g(x)$ is in $\mathcal{C}^2$ ($[0, +\infty[$), which is the set of functions that are twice continuously differentiable on $[0, +\infty[$, and of the form:

$$g(x) = h(x)\, e^{\varphi(x)},$$

and where $h(x) \in \tilde{A}^{(\gamma)}$ and $\varphi(x) \in \tilde{A}^{(k)}$ for some γ and k.

If $k > 0$, $\alpha_0(\varphi) < 0$ and for all $l = 0, \ldots, \lambda - 1$:

$$\lim_{x\to 0} x^{l-\lambda+1} \left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^l \left(x^{\lambda-1} g(x)\right) j_{\lambda-1-l}(x) = 0,$$

then $f(x)$ is integrable on $[0,+\infty[$, i.e., $\int_0^{+\infty} f(t)\,\mathrm{d}t$ exists, and:

$$\int_0^{+\infty} f(x)\mathrm{d}x = \int_0^{+\infty} \left[\left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^{\lambda}\left(x^{\lambda-1}\,g(x)\right)\right]\sin(x)\,\mathrm{d}x, \tag{66}$$

$$= \frac{1}{\nu^{\lambda+1}}\sum_{n=0}^{+\infty}\int_{n\,\pi/\nu}^{(n+1)\,\pi/\nu}\left[\left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^{\lambda}\left(x^{\lambda-1}\,g(x)\right)\right]\sin(\nu\,x)\,\mathrm{d}x \tag{67}$$

and an approximation of $\int_0^{+\infty} f(x)dx$ using the $\bar{D}$ transformation can be obtained using (57). ■

4.3 Recurrence Relations for the S Transformation

The computation of the approximation $S\bar{D}_n^{(2,j)}$ using (57) is not advantageous, because of the absence of the control of the degree of accuracy. Note also that (57) cannot be computed recursively. Recurrence relations were developed by Safouhi et al. [63,75] satisfied by both numerator $A_n^{(2,j)}$ and denominator $B_n^{(2,j)}$ of the term in the right hand side of (57).

The approximation $S\bar{D}_n^{(2,j)}$, can be rewritten as:

$$S\bar{D}_n^{(2,j)} = \frac{1}{\nu^{\lambda+1}}\frac{A_n^{(2,j)}}{B_n^{(2,j)}}; \quad n,\ j = 0,1,2,\ldots. \tag{68}$$

Let U_i^n and V_i^n be the i term of the finite sum $A_n^{(2,j)}$ and $B_n^{(2,j)}$, respectively. In [63], we showed that $\left(A_n^{(2,j)}\right)_n$ and $\left(B_n^{(2,j)}\right)_n$ satisfy the following relations:

$$\begin{cases} A_{n+1}^{(2,j)} = \displaystyle\sum_{i=0}^{n+1}\frac{(n+2)}{(n+2-i)}\,(1+i+j)\,U_i^n + U_{n+2}^{n+1} \\ B_{n+1}^{(2,j)} = \displaystyle\sum_{i=0}^{n+1}\frac{(n+2)}{(n+2-i)}\,(1+i+j)\,V_i^n + V_{n+2}^{n+1}. \end{cases} \tag{69}$$

From the above equations, it follows that $S\bar{D}_{n+1}^{(2,j)}$ can be rewritten as [63]:

$$S\bar{D}_{n+1}^{(2,j)} = \frac{1}{\nu^{l+1}}\frac{\displaystyle\sum_{i=0}^{n+1}\frac{(n+2)}{(n+2-i)}\,(1+i+j)\,U_i^n + U_{n+2}^{n+1}}{\displaystyle\sum_{i=0}^{n+1}\frac{(n+2)}{(n+2-i)}\,(1+i+j)\,V_i^n + V_{n+2}^{n+1}}. \tag{70}$$

The most important advantage of using the above equation is the control of the degree of accuracy. In fact, we do not calculate the approximation $S\bar{D}_{k+1}^{(2,j)}$, unless the accuracy obtained using $S\bar{D}_{k}^{(2,j)}$ is not satisfactory. For this we use the following test:

$$\left| S\bar{D}_{k}^{(2,j)} - S\bar{D}_{k-1}^{(2,j)} \right| = \frac{1}{v^{\lambda+1}} \left| \frac{A_k^{(2,j)}}{B_k^{(2,j)}} - \frac{A_{k-1}^{(2,j)}}{B_{k-1}^{(2,j)}} \right| \leq \epsilon, \tag{71}$$

where ϵ is defined according to the pre-determined degree of accuracy. In Table 4, we listed values obtained for the semi-infinite integral $\widetilde{\mathcal{J}}(s,t)$ with ϵ varying from 10^{-8} to 10^{-16}, to show the efficiency of the above test.

The storage of the values of U_i^k and V_i^k, $k = 0, 1, 2, \ldots$ and $i = 0, 1, \ldots, k+1$, led to a substantial gain in the calculation times. The calculation of all values of $x_{i+j}^2\, G(x_{i+j})$ for each order of the $S\bar{D}$ is avoided.

In [63, 76], we discussed the situation where $G(x_{i+j}) \rightarrow 0$ or $+\infty$. We demonstrated that in this situation we can obtain a very good approximation of the semi-infinite integral under consideration using the following formulae:

$$S\bar{D}_{n}^{(2,j)} \approx \frac{1}{v^{\lambda+1}} \frac{\displaystyle\sum_{i\in E} \binom{n+1}{i} (1+i+j)^n \frac{F(x_{i+j})}{x_{i+j}^2}}{\displaystyle\sum_{i\in E} \binom{n+1}{i} (1+i+j)^n \frac{1}{x_{i+j}^2}}, \tag{72}$$

where E is the subset of $I = \{0, 1, 2, \ldots, n+1\}$ defined by:

$$E = \left\{ k \in I \quad \text{such that} \quad G(x_{k+j}) \rightarrow 0 \ \text{ or } \ +\infty \right\}.$$

Note that the relations given by (69) are still applicable to the approximation $S\bar{D}_{n}^{(2,j)}$ given by (72). The following test was included in the algorithm:

$$R = \left| \frac{A_n^{(2,j)}}{\widetilde{A}_n{}^{(2,j)}} - \frac{B_n^{(2,j)}}{\widetilde{B}_n^{(2,j)}} \right| \leq \text{tiny} \quad \text{or} \quad \widetilde{R} = \left| \frac{\widetilde{A}_n^{(2,j)}}{A_n^{(2,j)}} - \frac{\widetilde{B}_n^{(2,j)}}{B_n^{(2,j)}} \right| \leq \text{tiny}. \tag{73}$$

where $\widetilde{A}_n{}^{(2,j)}$ stands for the numerator and $\widetilde{B}_n^{(2,j)}$ for the denominator of the term in the right hand side of (72) and where tiny should be set close to but not identical with the smallest floating point number that is representable on the computer. If the test is realized then the subroutine returns the approximation $S\bar{D}_{n}^{(2,j)}$ using (72) with the recurrence relations (69).

5 Numerical Treatment of ETFs Molecular Multi-Center Integrals

In previous works [33, 48, 49, 55], we have demonstrated the high efficiency of the nonlinear $\bar{D}$ transformation in improving the convergence of the semi-infinite integrals $f_{\mathcal{I}}(x)$, $f_{\mathcal{K}}(x)$, and $f_{\mathcal{J}}(x)$. Unfortunately, the approximation $\bar{D}_n^{(m)}$ requires a large number of calculations as it can be seen from (45). The computation of the successive positive zeros of the integrands is required as well as the computation of a method to solve the linear system, which can be large when the values of n and m are large. We note that these systems cannot be computed recursively and does not use the practical properties of the functions involved in the calculations.

In the case of the integrals under consideration, it is shown that the order of the differential equations of $f_{\mathcal{I}}(x)$, $f_{\mathcal{K}}(x)$ and $f_{\mathcal{J}}(x)$ can be reduced to two using properties of Bessel functions. From this it follows that the order of the linear systems to solve is $n + 1$.

In [77], we developed a highly efficient algorithm based on the $\bar{D}$ transformation and the W algorithm developed for a special case of generalization of the Richardson extrapolation process (GREP) [60, 78]. This algorithm, which can be computed recursively, allows the control of the degree of accuracy by carrying out a comparison between two successive terms and then stops the calculations when the predetermined accuracy is reached. In general, the test of accuracy works very well. However, in certain instances, when the behavior of the integrand is very unstable, the error (absolute or relative) will only grow larger. In such a situation, we developed an efficient test that stops the calculations and returns the most accurate approximation for the semi-infinite integral.

The disadvantages of the approaches described above arise mainly from the spherical Bessel functions.

The integrands $f_{\mathcal{I}}(x)$, $f_{\mathcal{K}}(x)$, and $f_{\mathcal{J}}(x)$ satisfy all the conditions of Theorems 4. The S transformation can be applied to the semi-infinite integrals $\mathcal{I}(s)$, $\mathcal{K}(s)$, and $\mathcal{J}(s,t)$. The corresponding integrals involving the simple sine function are given by:

$$\frac{1}{\nu^{\lambda+1}} \int_0^{+\infty} G(x)\, \sin(\nu x)\, \mathrm{d}x \;=\; \frac{1}{\nu^{\lambda+1}} \sum_{n=0}^{+\infty} \int_{n\frac{\pi}{\nu}}^{(n+1)\frac{\pi}{\nu}} G(x)\, \sin(\nu x)\, \mathrm{d}x, \qquad (74)$$

where the function $G(x)$ is given by:

- Three-center nuclear attraction integrals:

$$G(x) \;=\; \left(\frac{x}{x\,\mathrm{d}x}\right)^{\lambda} \left(x^{n_X+\lambda-1}\, \frac{\hat{k}_\nu[R_2\gamma(s,x)]}{[\gamma(s,x)]^{n_\gamma}} \right). \qquad (75)$$

- Hybrid and three-center two-electron Coulomb integrals:

$$G(x) \;=\; \left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^{\lambda} \left(\frac{x^{n_X+\lambda-1}}{\left[\zeta_s^2 + x^2\right]^{n_K}}\, \frac{\hat{k}_\nu[R_{34}\gamma_{34}(s,x)]}{[\gamma(s,x)]^{n_\gamma}} \right). \qquad (76)$$

- Four-center two-electron Coulomb integrals:

$$G(x) = \left(\frac{\mathrm{d}}{x\mathrm{d}x}\right)^{\lambda}\left(x^{n_x+\lambda-1}\frac{\hat{k}_{\nu_1}[R_{21}\gamma_{12}(s,x)]}{[\gamma_{12}(s,x)]^{n_{\gamma_{12}}}}\frac{\hat{k}_{\nu_2}[R_{34}\gamma_{34}(t,x)]}{[\gamma_{34}(t,x)]^{n_{\gamma_{34}}}}\right). \quad (77)$$

For the following, the obtained integrands with the sine functions will be referred to as $\tilde{f}_{\mathcal{I}}(x)$, $\tilde{f}_{\mathcal{K}}(x)$, and $\tilde{f}_{\mathcal{J}}(x)$, respectively.

With the help of the S transformation, the spherical Bessel integrals are transformed into semi-infinite integrals with the simple sine function. From this, it follows that the strong oscillations of the integrals are considerably reduced as it can be seen from Figs. 4–6, and this helps the extrapolation methods for improving convergence of the semi-infinite integrals under consideration.

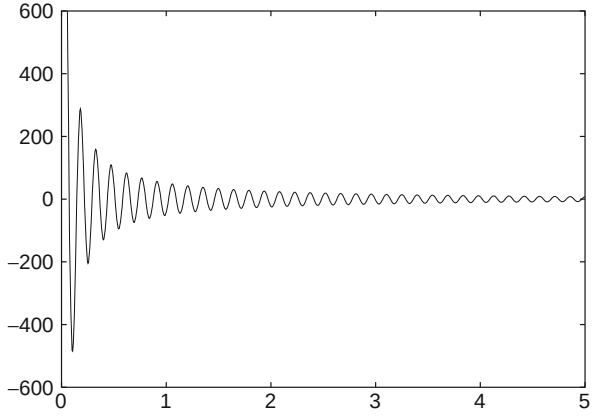


Fig. 4 The integrand $\tilde{f}_{\mathcal{I}}(x)$ of $\mathcal{I}(s)$. s, ν, n_γ, n_x, λ, R_1, R_2, ζ_1 and ζ_2 are given in Fig. 1

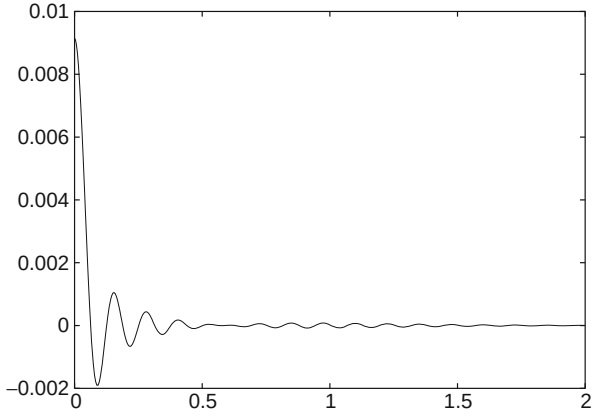


Fig. 5 The integrand $\tilde{f}_{\mathcal{K}}(x)$ of $\mathcal{K}(s)$. s, ν, n_γ, n_k, n_x, λ, R_{34}, ζ_s, ζ_3, and ζ_4 are given in Fig. 2

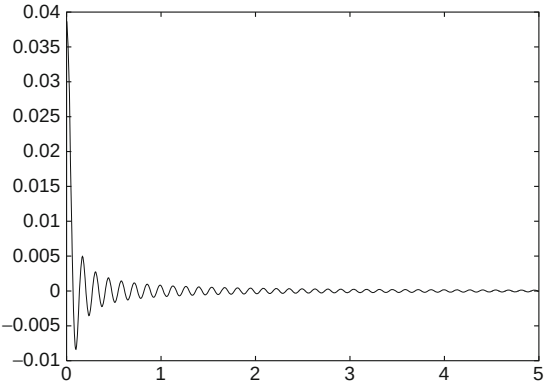


Fig. 6 The integrand $\tilde{f}_{\mathcal{J}}(x)$ of $\mathcal{J}(s,t)$. s, t, ν_{12}, ν_{34}, $n_{\gamma_{12}}$, $n_{\gamma_{34}}$, n_x, λ, ζ_1, ζ_2, ζ_3, ζ_4, R_{12}, and R_{34} are given in Fig. 3

6 Conclusion and Numerical Tables

This work presents the $S\bar{D}$ method for improving convergence of semi-infinite spherical Bessel integrals, which are known to be extremely difficult to evaluate accurately and rapidly. This $S\bar{D}$ approach is based on the S transformation of Safouhi and on the nonlinear $\bar{D}$ transformation of Sidi. The S transformation helped solving the difficulties related to the presence of spherical Bessel functions by transforming the semi-infinite integrals into semi-infinite integrals involving the simple sine function. This reduces considerably the strong oscillations of the integrands (as it can be seen from the figures presented in this paper).

The new integrands with the sine function are shown to satisfy all the conditions required to apply the $\bar{D}$ transformation, which is among the most powerful tools for improving convergence of semi-infinite integrals whose integrands satisfy linear differential equations with coefficients having asymptotic expansions in inverse powers of their argument x as $x \to \infty$.

It is now shown that the $S\bar{D}$ method led to a highly efficient algorithm for the numerical evaluation of spherical Bessel integrals. This algorithm has been applied for the numerical evaluation of the so-called molecular multi-center integrals over exponentially decreasing functions and the results obtained are in complete agreement with those obtained using existing codes.

The algorithm obtained from the $S\bar{D}$ approach is faster than the ones obtained from alternatives using the epsilon algorithm of Wynn, Levin's u transform, and the nonlinear $\bar{D}$ transformation when it is applied without the application of the S transformation.

The $S\bar{D}$ method is also able to reach precision of 10^{-15} and certainly some applications of this extremely high accuracy will be developed in future work.

Table 1 contains values of the semi-infinite integrals involved in the analytic expression of the three-center nuclear attraction integrals (39).

Table 2 contains values of the semi-infinite integrals involved in the analytic expression of the three-center two-electron Coulomb integrals (40).

Table 3 contains values of the semi-infinite integrals involved in the analytic expression of the four-center two-electron Coulomb integrals (41).

Table 1 Values with 15 correct decimals of $\mathcal{I}(s)$ involved in (39) obtained using the infinite series with the sine function (74), the infinite series with spherical Bessel function (42) and using $S\bar{D}_n^{(2,0)}$ with recurrence relations

s	ν	n_γ	n_x	λ	R_1	ν	$N^{\sin}$	$\widetilde{\mathcal{I}}(s)$ (74)	Time	N^{j_λ}	Error^a	Time	n	Error^b	Time
0.240(−2)	5/2	5	1	0	3.5	0.4916	29	0.13737257871458(0)	1.42	29	0.55(−16)	1.48	9	0.84(−12)	0.21
0.998(0)	5/2	5	1	0	3.5	2.9930	179	0.23341051164167(−3)	2.70	173	0.28(−15)	3.11	3	0.48(−11)	0.04
0.240(−2)	7/2	3	1	0	4.5	1.4892	84	0.10869578291379(0)	1.20	83	0.44(−15)	1.46	11	0.13(−11)	0.14
0.998(0)	7/2	7	1	0	4.5	2.9910	141	0.14809891443169(−3)	2.25	136	0.54(−15)	2.47	3	0.51(−11)	0.04
0.240(−2)	5/2	1	1	1	3.5	0.4916	35	0.10484272846079(1)	2.17	37	0.66(−15)	2.06	12	0.17(−12)	0.42
0.998(0)	5/2	5	1	1	3.5	2.9930	169	0.36642816695953(−3)	3.62	173	0.44(−15)	3.37	6	0.12(−10)	0.12
0.240(−2)	7/2	5	3	2	5.5	2.4868	124	0.79429921536274(−2)	2.90	128	0.27(−14)	3.13	43	0.28(−10)	0.79
0.998(0)	7/2	5	3	2	5.5	2.9890	159	0.11798135901654(−3)	3.76	166	0.56(−15)	4.16	29	0.12(−10)	0.53
0.240(−2)	11/2	9	4	3	5.5	2.4868	133	0.69742306814572(0)	3.79	141	0.24(−11)	3.91	53	0.10(−10)	1.10
0.240(−2)	5/2	1	1	0	34.0	32.0048	3,748	0.681713239241357(−3)	53.76	3,376	0.43(−15)	55.44	3	0.48(−16)	0.04
0.998(0)	5/2	1	1	0	34.0	33.9960	4,361	0.151864762852243(−2)	62.57	3,922	0.77(−15)	64.45	3	0.35(−16)	0.04
0.240(−2)	5/2	1	1	1	35.0	33.0048	3,598	0.100692333205546(−2)	77.00	3,474	0.89(−15)	64.50	6	0.19(−13)	0.10
0.998(0)	9/2	3	2	1	30.0	29.9960	4,433	0.590113822812245(−2)	97.51	4,330	0.76(−13)	92.27	3	0.37(−14)	0.07
0.240(−2)	9/2	3	2	2	45.0	43.0048	5,383	0.103067574467870(−2)	136.01	5,539	0.45(−13)	141.97	6	0.19(−13)	0.14
0.240(−2)	15/2	6	4	3	50.0	48.0048	8,022	0.160307326565246(−2)	229.18	8,599	0.50(−08)	248.17	3	0.11(-15)	0.09
0.998(0)	15/2	6	4	3	50.0	49.9960	9,152	0.177013372443209(−1)	261.87	9,853	0.45(−07)	283.86	3	0.68(−14)	0.09
0.240(−2)	17/2	9	5	4	55.0	53.0048	8,515	0.685144224550293(−3)	277.81	9,555	0.18(−06)	309.80	3	0.26(−15)	0.09
0.998(0)	17/2	9	5	4	55.0	54.9960	9,681	0.243705341101324(−1)	315.56	10,927	0.28(−05)	354.25	3	0.42(−13)	0.10

$R_2 = 3.0$, $\zeta_1 = 1.5$ and $\zeta_2 = 2.0$. $R_2, \zeta_1, \zeta_2, s, \nu, n_\gamma, n_x, \lambda, R_1$ and ν are given in Table 3

The values $\widetilde{\mathcal{I}}(s)$ (74) are obtained using the infinite series with the sine function, which we sun until $N^{\sin}$ order

The value $\mathcal{I}(s)^a$ were obtained using the infinite series with spherical Bessel function (42), which we sun until N^{j_λ} order

The value $\mathcal{I}(s)^b$ were obtained using $S\bar{D}_n^{(2,0)}$

Table 2 Values with 15 correct decimals of $\mathcal{K}(s)$ involved in (40), obtained using the infinite series with the sine function (74), the infinite series with spherical Bessel function (42) and using $S\bar{D}_n^{(2,0)}$ with recurrence relations. $s = 0.240(-2)^a$ and $s = 0.998$

ν	n_γ	n_k	n_x	λ	ζ_s	ζ_3	ζ_4	R_{34}	$N^{\sin}$	$\mathcal{K}(s)$	Time	N^{j_λ}	Error^b	Time	n	Error^c	Time
5/2	1	2	0	0	4.0	1.5	0.5	2.5	1,245	0.549128278838(−3)	11.85	1,245	0.22(−18)	12.20	6	0.13(−13)	0.09
5/2	1	2	1	0	2.0	2.0	1.5	2.0	1,260	0.424640684563(−4)	10.96	1,260	0.18(−18)	12.14	3	0.18(−10)	0.04
9/2	4	3	3	3	2.0	1.5	0.5	2.5	482	0.700836644746(−4)	10.89	1,773	0.37(−16)	22.82	18	0.22(−10)	0.45
13/2	5	2	4	4	2.0	1.0	1.0	2.0	642	0.351661145091(−3)	15.29	4,568	0.14(−13)	58.12	20	0.21(−10)	0.53
13/2	11	2	4	4	2.0	1.0	1.0	2.0	714	0.351661145092(−3)	17.14	2,632	0.19(−13)	33.84	20	0.71(−11)	0.54
13/2	13	2	6	6	2.0	1.0	1.0	2.0	361	0.151074181930(−4)	9.15	3,323	0.11(−12)	44.17	15	0.12(−10)	0.42
15/2	6	2	4	4	2.0	2.0	1.5	2.0	835	0.207548974232(−3)	20.23	5,152	0.38(−13)	65.76	21	0.17(−10)	0.53
21/2	9	2	5	5	2.0	2.0	1.5	2.0	1,045	0.559070180641(−1)	28.05	6,629	0.39(−09)	89.96	23	0.65(−11)	0.65
17/2	10	3	3	3	3.0	1.5	1.0	2.5	570	0.470570654794(−1)	13.03	1,256	0.32(−13)	16.89	22	0.41(−11)	0.51
5/2	1	2	0	0	4.0	1.5	0.5	2.5	1,301	0.496748720573(−4)	12.26	1,301	0.47(−19)	12.56	5	0.28(−11)	0.06
5/2	1	2	1	0	2.0	2.0	1.5	2.0	1,393	0.153899211686(−4)	12.23	1,393	0.11(−18)	13.60	3	0.46(−11)	0.06
9/2	4	3	3	3	2.0	1.5	0.5	2.5	307	0.348864079545(−6)	6.79	1,922	0.41(−17)	24.59	4	0.55(−12)	0.12
13/2	5	2	4	4	2.0	1.0	1.0	2.0	724	0.286993071501(−3)	17.28	5,180	0.21(−13)	65.90	20	0.91(−11)	0.53
13/2	11	2	4	4	2.0	1.0	1.0	2.0	772	0.286993071502(−3)	18.50	2,981	0.19(−13)	38.32	20	0.94(−11)	0.53
13/2	13	2	6	6	2.0	1.0	1.0	2.0	405	0.113667442373(−4)	10.28	3,792	0.17(−12)	50.40	14	0.42(−10)	0.40
15/2	6	2	4	4	2.0	2.0	1.5	2.0	471	0.241572463234(−4)	11.64	5,801	0.70(−14)	74.40	4	0.29(−10)	0.14
21/2	9	2	5	5	2.0	2.0	1.5	2.0	754	0.285406859358(−2)	20.90	7,501	0.13(−10)	102.23	21	0.20(−10)	0.65
17/2	10	3	3	3	3.0	1.5	1.0	2.5	576	0.484271561421(−3)	13.09	1,425	0.15(−14)	19.10	20	0.12(−10)	0.46

[a] Numbers in parentheses represent powers of 10

The values $\widetilde{\mathcal{K}}(s)$ (74) are obtained using the infinite series with the sine function, which we sun until $N^{\sin}$ order

The value $\mathcal{K}(s)^a$ were obtained using the infinite series with spherical Bessel function (42), which we sun until N^{j_λ} order

The value $\widetilde{\mathcal{K}}(s)^b$ were obtained using $S\bar{D}_n^{(2,0)}$

Table 3 Evaluation of the semi-infinite integral $\mathcal{J}(s,t)$ involved in (41), using the infinite series with the sine function (74), the infinite series with spherical Bessel function (42) and using $S\bar{D}_n^{(2,0)}$ with recurrence relations. $\lambda = n_x$, $\nu_{34} = \nu_{12}$ and $n_{\gamma_{34}} = n_{\gamma_{12}}$

ν_{12}	$n_{\gamma_{12}}$	n_x	ζ_1	ζ_2	ζ_3	ζ_4	R_{12}	R_{34}	N^{j_λ}	$N^{\sin}$	$\mathcal{J}(s,t)$	Values$^{S\bar{D}_6^{(2,0)}}$
											$s = 0.999$	$t = 0.005$
5/2	1	1	1.0	1.7	2.0	1.0	2.0	1.0	2,172	1,818	0.288 524 041 734(−2)	0.288 524 041 8(−2)
5/2	3	1	1.0	1.2	1.2	1.0	2.0	1.0	1,252	1,035	0.845 400 154 683(−2)	0.845 400 154 8(−2)
5/2	5	1	1.0	1.3	1.3	1.0	2.0	1.0	912	744	0.352 713 843 625(−2)	0.352 713 843 7(−2)
11/2	10	3	8.0	1.7	3.5	1.5	0.5	1.0	1,796	946	0.201 074 719 471(−2)	0.201 074 719 5(−2)
11/2	11	3	8.0	1.4	8.0	1.6	0.5	1.0	1,439	794	0.242 745 163 858(−2)	0.242 745 163 9(−2)
											$s = 0.999$	$t = 0.999$
5/2	5	0	1.5	1.0	1.0	1.5	0.5	1.5	1,424	1,424	0.444 524 869 234(0)	0.444 524 869 3(0)
9/2	9	2	2.0	1.5	1.5	2.0	2.0	1.0	1,543	1,047	0.128 293 378 253(−2)	0.128 293 378 3(−2)
9/2	7	3	6.0	1.4	1.4	5.0	2.0	1.0	2,395	1,359	0.358 146 135 268(−2)	0.358 146 135 3(−2)
9/2	9	3	2.0	1.4	1.4	5.0	2.0	1.0	1,845	1,043	0.981 745 642 221(−3)	0.981 745 642 2(−3)
											$s = 0.005$	$t = 0.005$
7/2	5	1	1.5	1.5	1.5	1.5	0.2	0.4	2,168	1,484	0.923 138 841 518(−2)	0.923 138 841 7(−2)
7/2	7	1	1.4	5.0	5.0	1.4	0.2	0.2	1,084	764	0.433 358 108 553(−2)	0.433 358 108 6(−2)
13/2	11	4	2.0	5.0	2.5	1.7	1.0	2.0	942	533	0.186 622 960 871(−2)	0.186 622 960 8(−2)
13/2	13	4	1.6	2.5	2.5	1.6	0.7	1.0	1,145	571	0.339 934 570 445(−2)	0.339 934 570 4(−2)
17/2	11	4	2.7	2.0	9.0	2.7	1.0	2.0	1,630	935	0.250 394 254 557(−2)	0.250 394 254 6(−2)
17/2	17	4	2.0	6.0	3.0	2.0	1.0	1.0	840	489	.172 215 398 336(−2)	0.172 215 398 4(−2)

The average of the calculation time for a single integral is 0.25 ms

The value $\mathcal{J}(s,t)$ were obtained using the infinite series with spherical Bessel function (42), which we sun until $N^{\sin}$ order

The value Values$^{S\bar{D}_6^{(2,0)}}$ were obtained using the using $S\bar{D}_n^{(2,0)}$ with recurrence relations

Table 4 contains values of the semi-infinite integrals involved in the analytic expression of the three-center nuclear attraction integrals (39). In this table, the test of accuracy (71) was used.

Table 5 contains values of the three-center nuclear attraction integrals over B functions (39) and Table 6 contains values of the three-center nuclear attraction integrals over STFs (26). The calculations were performed with the NCCH molecule.

Table 4 Values of the semi-infinite integral $\mathcal{I}(s)$ involved in (39) obtained using $S\bar{D}_n^{(2,0)}$. The test of accuracy (71) is used for different value of ϵ. $\zeta_1 = 0.25$, $\zeta_2 = 0.3$, and $\nu = 60.0$

ϵ	s	ν	n_γ	n_x	λ	R_1	ν	n	Error
10^{-08}	0.240D−02	9/2	3	2	2	45.00	43.0048	4	0.118D−08
10^{-12}	0.240D−02	9/2	3	2	2	45.00	43.0048	7	0.179D−12
10^{-16}	0.240D−02	9/2	3	2	2	45.00	43.0048	10	0.299D−16
10^{-08}	0.240D−02	5/2	1	1	1	35.00	33.0048	4	0.115D−08
10^{-12}	0.240D−02	5/2	1	1	1	35.00	33.0048	7	0.175D−12
10^{-16}	0.240D−02	5/2	1	1	1	35.00	33.0048	10	0.291D−16
10^{-08}	0.240D−02	9/2	3	2	2	30.00	28.0048	5	0.215D−09
10^{-12}	0.240D−02	9/2	3	2	2	30.00	28.0048	7	0.645D−12
10^{-16}	0.240D−02	9/2	3	2	2	30.00	28.0048	10	0.100D−16

Table 5 Values obtained for $\widetilde{\mathcal{I}}^{n_2,l_2,m_2}_{n_1,l_1,m_1}$ (39). $\mathbf{R}_1 = (R_1, 90°, 0°)$ and $\mathbf{R}_2 = (R_2, 90°, 0°)$

n_1	l_1	m_1	ζ_1	n_2	l_2	m_2	ζ_2	R_1	R_2	Values $S\bar{D}$	Values [24]
1	0	0	1.0	1	0	0	1.0	2.0	1.5	0.292219986(−1)	0.292220008(−1)
1	0	0	1.0	1	0	0	1.0	2.0	10.0	0.419890811(−4)	0.419894695(−4)
5	0	0	1.0	1	0	0	1.0	1.5	2.0	0.824711574(−2)	0.824711555(−2)
5	0	0	1.0	1	0	0	1.0	10.0	2.0	0.158775768(−2)	0.158690139(−2)
1	0	0	1.0	1	0	0	1.0	0.5	2.0	0.281222107(−1)	0.281222151(−1)
1	0	0	1.0	1	0	0	5.0	0.5	2.0	0.400999465(−3)	0.400999369(−3)
3	3	2	1.0	3	3	2	5.0	0.5	2.0	0.261739704(−8)	0.261739704(−8)
1	0	0	1.0	5	4	4	5.0	0.5	2.0	0.621915968(−5)	0.621916063(−5)
1	1	0	1.0	1	0	0	1.0	0.5	2.0	0.156906773(−2)	0.156906740(−2)

Table 6 Three-center nuclear attraction integrals over STFs (26). Values obtained with the linear molecule NCCH, using the following geometry: $N(0.,0.,0.)$, $C_1(2.1864\,au,0.,0.)$, $C_2(4.7980\,au,0.,0.)$, and $H(9.0689\,au,0.,0.)$

$\mathcal{I}^{\tilde{n}_2,l_2,m_2}_{\tilde{n}_1,l_1,m_1}$	ζ_1	ζ_2	Values $S\bar{D}$	Values [15]	Values [79]
$\langle 1\,s^N \vert (r_{NC^1})^{-1} \vert 1\,s^{C_2}\rangle$	8.93	5.23	0.550116114(−10)	0.550082293(−10)	0.550082316(−10)
$\langle 1\,s^N \vert (r_{NC^1})^{-1} \vert 2\,p_z^{C_2}\rangle$	8.93	1.25	−0.296504029(−02)	−0.296504016(−02)	−0.296503961(−02)
$\langle 1\,s^{C_1} \vert (r_{C^{13}})^{-1} \vert 1\,s^H\rangle$	5.23	0.98	0.169488048(−03)	0.169488048(−03)	0.169488044(−03)
$\langle 2\,s^{C_1} \vert (r_{C^{12}})^{-1} \vert 2\,s^{C^3}\rangle$	1.16	1.16	0.118891644(00)	0.118891647(00)	0.118891649(00)
$\langle 2\,p_z^{C_1} \vert (r_{C^{12}})^{-1} \vert 2\,p_z^{C^3}\rangle$	1.25	1.25	−0.188675485(00)	−0.188675450(00)	−0.188675497(00)
$\langle 2\,p_{+1}^{N} \vert (r_{NC^1})^{-1} \vert 2\,p_{+1}^{C^3}\rangle$	1.50	2.72	0.855582583(−04)	0.855582585(−04)	0.855582577(−04)
$\langle 1\,s^N \vert (r_{NC^1})^{-1} \vert 3\,d_z^H\rangle$	8.93	1.50	0.289180618(−04)	0.289180603(−04)	0.289180543(−04)
$\langle 1\,s^N \vert (r_{NC^3})^{-1} \vert 3\,d_z^H\rangle$	8.93	1.50	0.875086040(−05)	0.875085991(−05)	0.875085991(−05)

The abbreviation $2\,p_{+1}$ and $3\,d_z$ refer to the Slater functions defined by the quantum numbers: $(n = 2, l = 1, m = 1)$ and $(n = 3, l = 2, m = 0)$. The symbol $(r_{ab})^{-1}$ and $(r_{a^{ij}})^{-1}$ refer to the Coulomb operator $1/|\mathbf{R} - \mathbf{OC}|$ (27)

Table 7 contains values of the three-center two-electron Coulomb integrals over STFs (30). The calculations were performed with the H_2O molecule.

Tables 8 and 9 contain values of the four-center two-electron Coulomb integrals over STFs (36). These calculations were performed using the C_2H_2 (Table 8) and C_2H_4 (Table 9) molecules.

Table 7 Three-center two-electron Coulomb and hybrid integrals over STFs. Values obtained with the H_2O. The following geometry was used in spherical coordinates: $O(0, 0°, 0°)$, $H_1(1.810, 52.5°, 0°)$, and $H_2(1.810, 52.5°, 180°)$. $\zeta_{1s}^{O} = 7.670$, $\zeta_{2s}^{O} = 2.09$, $\zeta_{2p_z}^{O} = \zeta_{2p_{\pm1}}^{O} = 1.5$, and $\zeta_{1s}^{H} = 1.21$

Integral [a]	Values $S\bar{D}$	STOP [13]	ADGGSTNGINT [80]
$\langle 1s^O 1s^O \vert 1s^{H_1} 1s^{H_2}\rangle$	0.166 733 423(0)	0.166 733 529(0)	0.166 733 320(0)
$\langle 1s^O 2s^O \vert 1s^{H_1} 1s^{H_2}\rangle$	0.336 797 277(−1)	0.336 797 499(−1)	0.336 797 078(−1)
$\langle 1s^O 2p_z^O \vert 1s^{H_1} 1s^{H_2}\rangle$	0.153 901 824(−2)	0.153 901 948(−2)	0.153 901 856(−2)
$\langle 2s^O 2s^O \vert 1s^{H_1} 1s^{H_2}\rangle$	0.145 439 173(0)	0.145 439 265(0)	0.145 439 113(0)
$\langle 2p_z^O 2p_z^O \vert 1s^{H_1} 1s^{H_2}\rangle$	0.134 763 478(0)	0.134 763 562(0)	0.134 763 449(0)
$\langle 2p_{-1}^O 2p_{-1}^O \vert 1s^{H_1} 1s^{H_2}\rangle$	0.128 387 519(0)	0.128 387 598(0)	0.128 387 564(0)
$\langle 1s^{H_1} 1s^{H_1} \vert 1s^{H_2} 2p_z^O\rangle$	0.157 272 684(0)	0.157 272 410(0)	0.157 272 721(0)
$\langle 1s^{H_1} 1s^{H_1} \vert 1s^{H_2} 2p_{+1}^O\rangle$	−0.846 278 918(−1)	−0.846 277 865(−1)	−0.846 278 181(0)

[†] the abbreviations $2p_{+1}$ and $2p_{-1}$ refer to the Slater functions defined by the quantum numbers: $(n = 2, l = 1, m = 1)$ and $(n = 2, l = 1, m = -1)$

Table 8 Four-center two-electron Coulomb integrals over STFs $\mathcal{J}^{n_2l_2m_2,n_4l_4m_4}_{n_1l_1m_1,n_3l_3m_3}$ (36). Values obtained with C_2H_2. The calculation are obtained with the following geometry in Cartesian coordinates: $C_1(0; 0; 1.1405)$, $C_2(0; 0; -1.1405)$, $H_1(0; 0; 3.1425)$, and $H_2(0; 0; -3.1425)$ $\zeta_{1s}^{C} = 5.700$ and $\zeta_{2p_z}^{C} = \zeta_{2p_{+1}}^{C} = 1.625$. $\zeta_{1s}^{H} = 1.200$ and $\zeta_{2p_z}^{H} = \zeta_{2p_{+1}}^{H} = 2.220$

$\mathcal{J}^{n_2l_2m_2,n_4l_4m_4}_{n_1l_1m_1,n_3l_3m_3}$	Values $S\bar{D}$	ADGGSTNGINT [80]
$\langle 1s^{C_1} 1s^{H_1} \vert 1s^{C_2} 1s^{H_2}\rangle$	0.195 966 315 243(−2)	0.195 966 312 886(−2)
$\langle 1s^{C_1} 1s^{H_1} \vert 1s^{C_2} 2p_z^{H_2}\rangle$	0.283 669 225 792(−2)	0.283 669 224 706(−2)
$\langle 1s^{C_1} 2p_z^{H_1} \vert 1s^{C_2} 2p_z^{H_2}\rangle$	−0.410 711 928 328(−2)	−0.410 711 931 655(−2)
$\langle 2p_z^{C_1} 1s^{H_1} \vert 2p_z^{C_2} 1s^{H_2}\rangle$	−0.384 782 080 613(−1)	−0.384 782 080 602(−1)
$\langle 2p_z^{C_1} 2p_z^{H_1} \vert 2p_z^{C_2} 2p_z^{H_2}\rangle$	0.178 337 206 024(−1)	0.178 337 206 021(−1)
$\langle 2p_{+1}^{C_1} 1s^{H_1} \vert 2p_{+1}^{C_2} 1s^{H_2}\rangle$	0.279 688 126 236(−2)	0.279 688 126 236(−2)

Table 9 Four-center two-electron Coulomb integrals over STFs $\mathcal{J}^{n_2l_2m_2,n_4l_4m_4}_{n_1l_1m_1,n_3l_3m_3}$ (36). Values obtained with C_2H_4. The calculation are obtained with the following geometry in Cartesian coordinates: $C_1(0; 0; 1.2755)$, $C_2(0; 0; -1.2755)$, $H_1(1.7528; 0; 2.2875)$, $H_2(1.7528; 0; -2.2875)$, $H_3(-1.7528; 0; 2.2875)$, and $H_4(-1.7528; 0; -2.2875)$. $\zeta_{1s}^{C} = 5.700$, $\zeta_{2p_z}^{C} = 1.625$, and $\zeta_{3d_z}^{C} = 1.250$. $\zeta_{1s}^{H} = 1.200$ and $\zeta_{2p_z}^{H} = 2.220$

$\mathcal{J}^{n_2l_2m_2,n_4l_4m_4}_{n_1l_1m_1,n_3l_3m_3}$	Values $S\bar{D}$	ADGGSTNGINT [80]
$\langle 1s^{H_1} 1s^{H_2} \vert 1s^{H_3} 1s^{H_4}\rangle$	0.121 073 251 2(−2)	0.121 073 251 2(−2)
$\langle 1s^{C_1} 1s^{C_2} \vert 1s^{H_1} 1s^{H_2}\rangle$	0.126 220 327 3(−5)	0.126 220 001 6(−5)
$\langle 1s^{C_1} 1s^{H_2} \vert 1s^{C_2} 1s^{H_3}\rangle$	0.210 918 868 0(−4)	0.210 876 132 2(−4)
$\langle 1s^{C_1} 2p_z^{C_2} \vert 1s^{H_1} 1s^{H_2}\rangle$	0.230 206 462 0(−2)	0.230 206 400 5(−2)
$\langle 2p_z^{C_1} 2p_z^{C_2} \vert 1s^{H_1} 1s^{H_2}\rangle$	−0.102 656 885 6(−1)	−0.102 656 885 6(−1)
$\langle 2p_z^{C_1} 1s^{H_2} \vert 2p_z^{C_2} 1s^{H_3}\rangle$	−0.749 068 354 6(−2)	−0.749 068 354 4(−2)
$\langle 3d_z^{C_1} 2p_z^{H_2} \vert 3d_z^{C_2} 1s^{H_3}\rangle$	0.555 282 977 6(−2)	0.555 282 977 1(−2)
$\langle 3d_z^{C_1} 2p_z^{H_2} \vert 3d_z^{C_2} 2p_z^{H_3}\rangle$	−0.248 064 450 1(−2)	−0.248 064 449 9(−2)

Acknowledgement The authors acknowledges the financial support of this research by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. C.C. Roothaan, Rev. Mod. Phys. **23**, 69–89 (1951)
2. E.R. Davidson, D. Feller, Chem. Rev. **86**, 681–696 (1986)
3. T. Kato, Commun. Pure Appl. Math. **10**, 151–177 (1957)
4. S. Agmon, *Lectures on Exponential Decay of Solutions Of Second-order Elliptic Equations: Bounds of Eigenfunctions of N-body Schrödinger Operators* (Princeton University, Princeton, NJ, 1982)
5. S. Agmon, in *Bounds on Exponential Decay of Eigenfunctions of Schrödinger Operators*, ed. by S. Graffi. Schrödinger operators (Springer, Berlin, 1985)
6. S.F. Boys, Proc. R. Soc. Lond. Series A: Math. Phys. Sci. **200**, 542–554 (1950)
7. S.F. Boys, Proc. R. Soc. Lond. Series A: Math. Phys. Sci. **201**, 125–137 (1950)
8. C.A. Weatherford, H.W. Jones, *ETO Multicenter Molecular Integrals* (Reidel, Dordrecht, 1982)
9. J.C. Slater, Phys. Rev. **36**, 57–64 (1930)
10. J.C. Slater, Phys. Rev. **42**, 33–43 (1932)
11. Amsterdam Density Functional Program, Theoretical Chemistry, Vrije Universiteit, Amsterdam. URL: http://www.scm.com/
12. W. Kutzelnigg, J. Mol. Struct (Theochem) **50**, 33–54 (1988)
13. A. Bouferguene, M. Fares, P.E. Hoggan, Int. J. Quant. Chem. **57**, 801–810 (1996)
14. M.A. Watson, N.C. Handy, A.J. Cohen, J. Chem. Phys. **119**, 6475–6481 (2003)
15. A. Bouferguene, D. Rinaldi, Int. J. Quant. Chem. **50**, 21–42 (1994)
16. F. J. Rico, R. Lòpez, A. Aguado, I. Ema, G. Ramìrez, Int. J. Quant. Chem. **19**, 1284–1293 (1998)
17. I. Shavitt, in *The Gaussian function in Calculation of Statistical Mechanics and Quantum Mechanics*, ed. by B. Alder, S. Fernbach, M. Rotenberg. Methods in Computational Physics. 2. Quantum Mechanics (Academic Press, New York, 1963)
18. E. Filter, E.O. Steinborn, Phys. Rev. A. **18**, 1–11 (1978)
19. E.O. Steinborn, E. Filter, Theor. Chim. Acta. **38**, 273–281 (1975)
20. E.J. Weniger, E.O. Steinborn, J. Math. Phys. **30**, 774–784 (1989)
21. E.J. Weniger, Reduzierte Bessel-Funktionen als LCAO-Basissatz: Analytische und numerische Untersuchungen. Ph.D.thesis, Universität Regensburg, 1982
22. E.J. Weniger, E.O. Steinborn, J. Chem. Phys. **78**, 6121–6132 (1983)
23. H.P. Trivedi, E.O. Steinborn, Phys. Rev. A. **27**, 670–679 (1983)
24. J. Grotendorst, E.O. Steinborn, Phys. Rev. A. **38**, 3857–3876 (1988)
25. E. Filter, Analytische Methoden zur Auswertung von Mehrzentren-Matrixelementen in der Theorie der Molekülorbitale bei Verwendung exponentialartiger Basissätze. Ph.D. thesis, Universität Regensburg, 1978
26. E.J. Weniger, J. Grotendorst, E.O. Steinborn, Phys. Rev. A. **33**, 3688–3705 (1986)
27. J. Grotendorst, E.J. Weniger, E.O. Steinborn, Phys. Rev. A. **33**, 3706–3726 (1986)
28. E. Filter, E.O. Steinborn, J. Math. Phys. **19**, 79–84 (1978)
29. E.J. Weniger, E.O. Steinborn, Phys. Rev. A. **28**, 2026–2041 (1983)
30. E.J. Weniger, Collect. Czech. Chem. Commun. **70**, 1125–1271 (2005)
31. F.P. Prosser, C.H. Blanchard, J. Chem. Phys. **36**, 1112–1112 (1962)
32. M. Geller, J. Chem. Phys. **39**, 853–854 (1963)
33. H. Safouhi, P.E. Hoggan, J. Phys. A: Math. Gen. **31**, 8941–8951 (1998)
34. H. Safouhi, P.E. Hoggan, J. Math. Chem. **25**, 259–280 (1999)

35. H. Safouhi, Nonlinear transformations for accelerating the convergence of molecular multicenter bielectronic integrals. Ph.D thesis, Université de Blaise Pascal, Clermont-Ferrand, France, 1999
36. P. Wynn, Proc. Glascow Math. Assoc. **5**, 160–165 (1962)
37. D. Levin, Int. J. Comput. Math. **B3**, 371–388 (1973)
38. P. Wynn, Math. Tables Aids Comput. **10**, 91–96 (1956)
39. D. Shanks, J. Math. Phys. **34**, 1–42 (1955)
40. C. Brezinski, *Algorithmes d'Accélérations de la Convergence* (Technip, Paris, 1978)
41. C. Brezinski, M. Redivo-Zaglia, *Extrapolation Methods: Theory and Practice* (North-Holland, Amsterdam, 1991)
42. E.J. Weniger, Comput. Phys. **10**, 496–503, 1996
43. E.J. Weniger, B. Kirtman, Comput. Math. Appl. **45**, 189–215 (2003)
44. E.J. Weniger, Numer. Algorithm. **33**, 499–507 (2003)
45. E.J. Weniger, J. Čížek, F. Vinette J. Math. Phys. **34**, 571–609 (1993)
46. E.J. Weniger, Ann. Phys. (NY) **246**, 133–165 (1996)
47. E.J. Weniger, Phys. Rev. Lett. **77**, 2859–2862 (1996)
48. H. Safouhi, D. Pinchon, P.E. Hoggan, Int. J. Quant. Chem. **70**, 181–188 (1998)
49. H. Safouhi, P.E. Hoggan, J. Phys. A: Math. Gen. **32**, 6203–6217 (1999)
50. H. Safouhi, P.E. Hoggan, J. Comp. Phys. **155**, 331–347 (1999)
51. H. Safouhi, P.E. Hoggan, Int. J. Quant. Chem. **80**, 236–248 (2000)
52. H. Safouhi, J. Comp. Phys. **165**, 473–495 (2000)
53. H. Safouhi, J. Math. Chem. **29**, 213–232 (2001)
54. H. Safouhi, P.E. Hoggan, Mol. Phys. **101**, 19–31 (2003)
55. H. Safouhi, Int. J. Quant. Chem. **100**, 172–183 (2004)
56. L. Berlu, H. Safouhi, J. Theor. Comp. Chem. **4**, 787–801 (2005)
57. H. Safouhi, J. Mol. Mod. **12**, 213–220 (2006)
58. D. Levin, A. Sidi, Appl. Math. Comput. **9**, 175–215 (1981)
59. A. Sidi, J. Inst. Math. Appl. **26**, 1–20 (1980)
60. A. Sidi, J. Comp. Appl. Math. **78**, 125–130 (1997)
61. H. Poincaré, Acta. Math. **8**, 295–344 (1886)
62. H. Safouhi, J. Phys. A: Math. Gen. **34**, 881–902 (2001)
63. L. Berlu, H. Safouhi, J. Phys. A: Math. Gen. **36**, 11791–11805 (2003)
64. H. Safouhi, J. Phys. A: Math. Gen. **34**, 2801–2818 (2001)
65. H. Safouhi, J. Comp. Phys. **176**, 1–19 (2002)
66. P.J. Davis, P. Rabinowitz, *Methods of Numerical Integration* (Academic Press, Orlando, 1994)
67. G. Evans, *Practical Numerical Integration* (Wiley, Chichester, 1993)
68. G.B. Arfken, H.J. Weber, *Mathematical Methods for Physicists*, 5th edn. (Academic Press, New York, 1995)
69. E.J. Weniger, E.O. Steinborn, J. Math. Phys. **24**, 2553–2563 (1983)
70. J.A. Gaunt, Phil. Trans. Roy. Soc. A **228**, 151–196 (1929)
71. H.H.H. Homeier, E.O. Steinborn, J. Mol. Struct. (Theocem) **368**, 31–37 (1996)
72. E.J. Weniger, E.O. Steinborn, Comput. Phys. Commun. **25**, 149–157 (1982)
73. I.M. Gel'fand, G.E. Shilov, *Generalized Functions I, Properties and Operations* (Academic, New York, 1964)
74. E.J. Weniger, Comput. Phys. Rep. **10**, 189–371 (1989)
75. H. Safouhi, L. Berlu, J. Comp. Phys. **216**, 19–36 (2006)
76. L. Berlu, H. Safouhi, J. Phys. A: Math. Gen. **36**, 11267–11283 (2003)
77. S. Duret, H. Safouhi, Int. J. Quant. Chem. **107**, 1060–1066 (2007)
78. A. Sidi, Numer. Math. **38**, 299–307 (1982)
79. P.S. Bagus, McLean, M. Yoshimine, B.H. Lengsfield, B. Liu, in *Alchemy II*. International Business Machines, from MOTECC-90 (San Jose, 1990)
80. J.F. Rico, R. Lòpez, I. Ema, G. Ramírez, Comp. Phys. Commun. **105**, 216–224 (1997)
81. E.U. Condon, G.H. Shortley, *The theory of atomic spectra* (Cambridge University Press, Cambridge, England, 1935)

String Mining in Bioinformatics

Mohamed Abouelhoda and Moustafa Ghanem

1 Introduction

Sequence analysis is a major area in bioinformatics encompassing the methods and techniques for studying the biological sequences, DNA, RNA, and proteins, on the linear structure level. The focus of this area is generally on the identification of intra- and inter-molecular similarities. Identifying intra-molecular similarities boils down to detecting repeated segments within a given sequence, while identifying inter-molecular similarities amounts to spotting common segments among two or multiple sequences.

From a data mining point of view, sequence analysis is nothing but *string-* or *pattern mining* specific to biological strings. For a long time, this point of view, however, has not been explicitly embraced neither in the data mining nor in the sequence analysis text books, which may be attributed to the co-evolution of the two apparently independent fields. In other words, although the word "data-mining" is almost missing in the sequence analysis literature, its basic concepts have been implicitly applied. Interestingly, recent research in biological sequence analysis introduced efficient solutions to many problems in data mining, such as querying and analyzing time series [49, 53], extracting information from web pages [20], fighting spam mails [50], detecting plagiarism [22], and spotting duplications in software systems [14].

In this chapter, we review the basic problems in the area of biological sequence analysis. We present a taxonomy of the main problems in this area and introduce basic solutions to them. Moreover, we show some interesting applications of the string data structures to some traditional problems in data mining, such as finding frequent itemsets, computing string kernels, and mining semi- and unstructured text documents.

M. Abouelhoda (✉)
Cairo University, Orman, Gamaa Street, 12613 Al Jizah, Giza, Egypt
Nile University, Cairo-Alex Desert Rd, Cairo 12677, Egypt

M.M. Gaber (ed.), *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*, DOI 10.1007/978-3-642-02788-8_9,

2 Background

The three key types of biological sequences of interest to bioinformatics are DNA sequences, protein sequences and RNA sequences. A DNA sequence, e.g., GTAAACTGGTAC..., is a string formed from an alphabet of four letters (A, C, G, and T), each representing one of four different nucleotides. Using the genetic code, a DNA sequence can be translated into a corresponding protein sequence, whereby each triplet of nucleotides (letters) codes for one amino acid, e.g., the triplet GTA translates into the amino acid Valine represented by V, and the triplet AAC translates into the amino acid Asparagine represented by N, etc. A protein sequence, e.g., VNWYHLDKLMNEFF..., is thus also a string formed from an alphabet of twenty characters each representing an amino acid, these are (A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, and V). Similarly, RNA sequences are formed from a four-letter alphabet of nucleotides (A, C, G, and U).

Knowing the sequence of letters of a DNA, an RNA or a protein is not an ultimate goal in itself. Rather, the major task is to *understand* the sequence, in terms of its structure and biological function. This is typically achieved first by identifying individual regions or structural units within each sequence and then assigning a function to each structural unit. Overall, two broad strategies are used when identifying and understanding the features of a biological sequence; *intrinsic strategy* in which just a single sequence is studied and *comparative strategy* in which the sequence is compared to other sequences, in an attempt to infer how it relates to similar structural and functional units in them.

Intrinsic Strategy

Various methods within the intrinsic strategy can be used for the analysis of genomic (DNA) sequences to identify the roles that different segments (substrings) of the sequence play. Such analysis includes identifying sequence features, such as the G+C content of the different regions, and segmenting the genome into its key structural units, such as genes, repeats, and regulatory elements. In simple cases, identifying the different segments can be achieved by searching for known patterns of substrings that mark these segments. An example is searching for the patterns signifying the *promoter* regions that are essential in the biological transcription process.

In more complex cases, the use of probabilistic predictive methods is needed. These predictive methods take into account different types of evidence that are associated with the region to be identified (e.g., a gene), and are typically based on statistical properties. These predictive methods are typically known as ab-initio methods, (see for example [19] for an evaluation of gene prediction methods).

Similarly, various intrinsic analysis methods can be used for the analysis of protein sequences. In simple cases, the analyses include identifying the basic structural and functional subsequences of the protein, referred to as *domains* and *motifs*, respectively. The analyses can also include the use of statistical methods to predict the 3-D structure and/or the function of the protein (see for example [35] for the use of Hidden Markov Models for the prediction of protein structure).

Comparative Strategy

Comparative analysis methods are used over both DNA and protein sequences to study the relationship of genome structure and function across different biological species or strains. This is important not only to study phylogeny, or how different organisms may have evolved, but also as a means to understand the roles and functions of newly discovered genes and proteins. Given either a DNA or protein sequence, a typical recurring task in bioinformatics is to search known genomic and proteomics databanks for similar sequences. Many of the sequences stored in such databanks have been manually curated and annotated by experts over the years, and their properties and functions were experimentally verified. The identification and study of such similar sequences can provide many valuable clues relating to the sequence under examination.

A number of issues arise when conducting similarity searches over large biological sequence databases. The first relates to quantifying the measure of similarity, or conversely distance, between two sequences. The simplest approach is based on using a Hamming distance measure that counts the number of mismatches between two strings. As an example the Hamming distance between the two sequences SSHLDKLMNEFF and HSHLKLLMKEFF is four, as shown below where an * is used to signify the positions of the mismatches.

```
SSHLDKLMNEFF
*   **  *
HSHLKLLMKEFF
```

However, since typically two sequences being compared are unlikely to be of the same length, a simple sequence similarity algorithm needs to introduce gaps corresponding to insertions/deletions within the sequences. Comparing sequences with gaps can be shown in the example below:

```
MHHNALQRRTVWVNAY
MHH-ALQRRTVWVNAY
```

A second issue that arises when comparing the similarity of biological sequences is that the simple Hamming distance metric does not take into account the likelihood of one amino acid (or nucleotide in case of DNA) changing to another one due to mutations. Although some amino acid substitutions are disastrous and do not survive evolution, others have almost no effect because the two amino acids are chemically quite similar in terms of their properties. Such issues are typically addressed using a scoring scheme, such as Percent Accepted Mutation (PAM) and Block Substitution Matrices (BLOSUM) used in the widely used BLAST sequence comparison algorithm [10].

Taking into consideration the possibilities for *indels* (Insertions or Deletions) as well as the likelihood of amino acid (nucleotide) substitutions leads to a wide spectrum of string mining problems that the bioinformatics community has been addressing over the past two decades.

Data Structures and Algorithms for String Mining

In this chapter, we focus on string mining problems used in both intrinsic and comparative sequence analysis strategies.

In Sect. 3, we start by introducing some basic formal definitions and data structures that are typically used. These basic data structures include Look-up tables, Automata and Tries, and the Suffix Tree data structure. These data structures are key to the efficient execution of many algorithms over large collections of long sequences.

In Sect. 4, we present a taxonomy of the major repeat-related problems in bioinformatics, and describe the basic exact and approximate string mining algorithms algorithms used to address these problems. The problems considered include finding dispersed fixed length and maximal length repeats, finding tandem repeats, and finding unique subsequences and missing (un-spelled) subsequences.

In Sect. 5, we provide a taxonomy of the key algorithms for sequence comparison, including global, semi-global and local sequence alignment and biological database search methods. We also describe the basic algorithms used in each case covering exact and approximate algorithms for both fixed- and variable-length sequences, thus covering most popular sequence comparison algorithms in a unified framework.

In Sect. 6, we describe how the basic algorithms presented in this chapter can be easily applied to string mining applications beyond bioinformatics. These include applications in frequent itemset mining, string kernels as well as information extraction applications for free text and semi-structured data sources.

We note that the prediction of biological sequence features based on statistical inference or machine learning (as in the case of ab-initio gene identification), are not addressed directly in this chapter. However, the key data structures presented in this chapter provide the basis for calculating, in an efficient manner, the statistics required by such algorithms.

3 Basic Definitions and Data Structures

3.1 Basic Notions

Let Σ denote an ordered alphabet and let $|\Sigma|$ denote the size of this alphabet. Let S be a string over Σ of length $|S| = n$. We write $S[i]$ to denote the character at position i in S, for $0 \leq i < n$. For $i \leq j$, $S[i..j]$ denotes the *substring* S starting with the character at position i and ending with the character at position j. The substring $S[i..j]$ is also denoted by the *pair* (i, j) *of positions*. The substring $S[0..i]$ is a prefix of S. The substring $S[i..n-1]$ is the i-th suffix of S, and it is denoted by $S(i)$.

As introduced in Sect. 2, the three types of biological sequences: DNAs, RNAs, and proteins differ in their alphabets. For example, for DNA sequences, the DNA sequence is a string over an alphabet of four characters, namely A, C, G, and T. We also note that in Computer Science terminology, the notion "sequence" is not the same as "string." However, "sequence" in Biology corresponds actually to "string" in Computer Science. So, unless otherwise stated, we will use "sequence" and "string" interchangeably in this chapter.

The *complement* string of a DNA string is obtained, from biological knowledge, by replacing every C with G, G with C, T with A, and A with T; i.e., this replacement follows the base pairing rule. The *reverse* string of $S[1..n]$ is the string $S[n]S[n-1]..S[1]$. The *reverse complement* of S is the complement string of $S[n]S[n-1]..S[1]$. Neither the reverse nor the complement string is of interest in molecular biology. The reverse complement, however, is as important as the DNA string S because it represents the complementary strand of this molecule, where biological information is also encoded. Using the genetic code, any DNA or RNA sequence that encodes for a protein can be transformed to the corresponding sequence of amino acids.

3.2 Look-Up Table

For a given parameter d, the *look-up table* is a table of length $|\Sigma|^d$ such that each string of length d maps to just one entry in the table. Note that the table size corresponds to the $|\Sigma|^d$ possible strings. We call any of these strings *key*. Each entry of the table points to a linked list storing the positions where the associated string occurs in S. Note that some entries may be empty, as the respective substring does not occur in S. The mapping function, for a string $P[0..d-1]$, is

$$x = f(P[0..d-1]) = \sum_{i=0}^{d-1} |\Sigma|^{d-i-1} T(P[i])$$

where x is an integer value representing an index of the look-up table, Σ is the alphabet size, and $T(S[i])$ is a function that maps each character in the alphabet to a number in the range $[0 \ldots |\Sigma|]$. Figure 1 shows an example of a look-up table with the mapping $a \to 0$, $c \to 1$, $g \to 2$, and $t \to 3$.

The look-up table can be naively constructed by sliding a window of length d over S and computing the mapping function $f(w)$ for each substring $w \in S$. This takes $O(d|S|)$ time. This time complexity, however, can be reduced to $O(|S|)$ using the following recurrence, which can be implemented by scanning the string S from left to right.

$$f(S[i+1..i+d]) = (f(S[i..i+d-1]) - T(S[i])|\Sigma|^{d-1}) \times |\Sigma| + T(S[i+d-1])$$

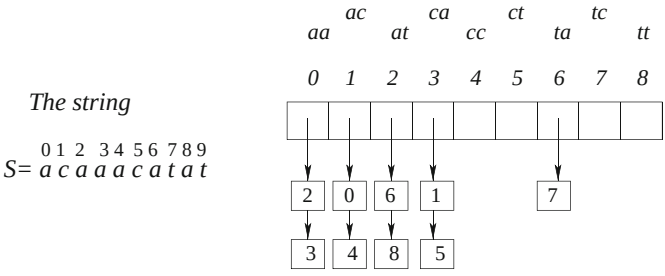


Fig. 1 The look-up table for $S = acaaacatat$

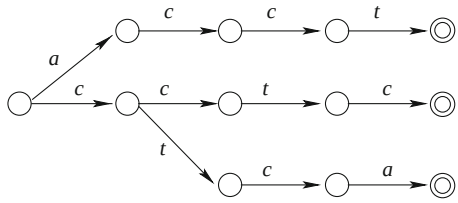


Fig. 2 A trie of the string *acctca*. The keys are the substrings of length 4

3.3 Automata and Tries

Aho and Corasick [9] showed how to construct a deterministic finite automata for a set of keys (words). The target was to improve online string matching for bibliographic search. For a genomic sequence, the keys can be taken as the set of its subsequences (more precisely, its substrings). To reduce the space, one can take subsequences up to certain length k. The locations where these substrings occur in the sequence can be attached to the leaves (the nodes of success) of the automaton. Figure 2 shows an example of an automaton for the string *acctca*, where $k = 4$. The string has the three keys *acct*, *cctc*, and *ctca*.

It is interesting to see that this automaton is nothing but a *pruned non-compact suffix tree*. (Suffix tree is presented in the next subsection.) A non-compact suffix tree, known as *trie*, has the edge labels explicitly stored with the tree, and for each label there is an edge. (As a consequence not all the nodes of the tree are branching.) Because the space consumption of a trie grows quadratically with the sequence length, it is necessary to prune it at an appropriate depth.

For maximum depth k, the automata can be constructed in $O(nk)$ time by adding the n keys successively to the automata as the original method of Aho and Corasick [9]. But this time complexity can be improved by constructing the suffix tree in $O(n)$ time, and pruning it such that the maximum depth is k characters from the root.

3.4 The Suffix Tree

The suffix tree is a focal data structure in string processing and the linear time algorithm for constructing it is a milestone in this area. The suffix tree is defined as follows: Let S be a string of n characters and let the character \$ be appended to S. A *suffix tree* for the string $S\$$ is a rooted directed tree with exactly $n + 1$ leaves numbered from 0 to n. Each internal node, other than the root, has at least two children and each edge is labeled with a nonempty substring of $S\$$. No two edges out of a node can have edge-labels beginning with the same character. The key feature of the suffix tree is that for any leaf i, the concatenation of the edge-labels on the path from the root to leaf i exactly spells out the substring $S[i..n-1]\$$ that denotes the i-th nonempty suffix of the string $S\$$, $0 \leq i \leq n$. Moreover, the concatenation of the edge-labels on the path from the root to a non-leaf node spells out a substring of S that occurs z times in S, where z is the number of all the leaves under the subtree of this non-leaf node. Figure 3 shows the suffix tree for the string $S = acaaacatat\$$. The character \$ is known as the sentinel character and it is appended to obtain a tree in which every suffix corresponds to a leaf. More precisely, without the sentinel character some suffixes would be proper prefixes of other suffixes. For example, assume S to be aa, then without the sentinel character the second suffix is a prefix of the first and it will not be straightforward to have a leaf for the second suffix and to distinguish (lexicographically sort) the two suffixes of S. Throughout this chapter, we use the notion *suffix tree of* S but this implicitly means that the suffix tree is constructed for $S\$$; i.e., the sentinel character \$ is already appended to S.

To store the suffix tree in linear space, the edge-labels are not explicitly stored with the edges. Rather, for each edge a pair of positions (i, j) is attached to represent

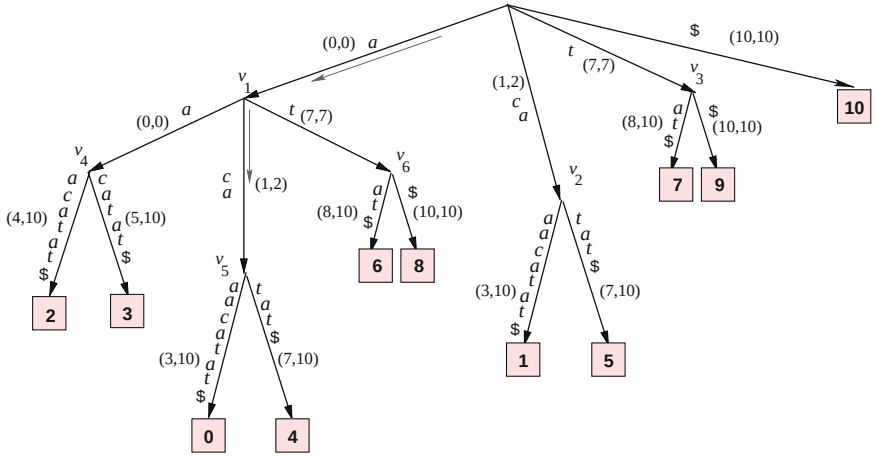


Fig. 3 The suffix tree for $S = acaaacatat\$$. The sentinel character is appended to S. A pair (i, j) represents a substring $S[i..j]$ corresponding to the respective edge-labels. The gray arrows indicates to the edges traversed while searching for the pattern ac

a substring $S[i..j]$ corresponding to the respective edge-labels. This suffix tree representation is commonly referred to as *compact suffix tree*. (If the labels are explicitly attached to the edges, the structure is called non-compact suffix tree or suffix trie. In this case, the space complexity becomes $O(n^2)$; consider, e.g., a string like *abcde*.)

The suffix tree can be constructed in $O(n)$ time and space [45, 60]; see also [28] for a simplified exposition. Once constructed, it can be used to efficiently solve the problems specified in Fig. 5, but it can also be used to solve other string processing applications; see [12, 28].

It is worth mentioning that the space consumption and the poor cache performance of the suffix tree is a bottleneck for large scale applications [25, 33, 42]. The enhanced suffix array [1] represents an alternative data structure to the suffix tree that requires less space and achieves better cache performance. In [1] it was shown that any algorithm using the suffix tree can be systematically replaced with an equivalent one using the enhanced suffix array. Nevertheless, for clarity of presentation, we will describe the algorithms in this chapter over the suffix tree. For mapping the algorithms to the enhanced suffix array, we refer the reader to [1, 7].

4 Repeat-Related Problems

A pair of substrings $R = ((i_1, j_1), (i_2, j_2))$ is a *repeated pair* if and only if $(i_1, j_1) \neq (i_2, j_2)$ and $S[i_1..j_1] = S[i_2..j_2]$. The length of R is $j_1 - i_1 + 1$. A repeated pair $((i_1, j_1), (i_2, j_2))$ is called *left maximal* if $S[i_1 - 1] \neq S[i_2 - 1]$ and *right maximal* if $S[j_1 + 1] \neq S[j_2 + 1]$. A repeated pair is called *maximal* if it is both left and right maximal. A substring ω of S is a (*maximal*) *repeat* if there is a (maximal) repeated pair $((i_1, j_1), (i_2, j_2))$ such that $\omega = S[i_1..j_1]$. A *supermaximal repeat* is a maximal repeat that never occurs as a substring of any other maximal repeat. Figure 4 shows maximal repeated pairs and supermaximal repeats of the string $S = gagctagagcg$.

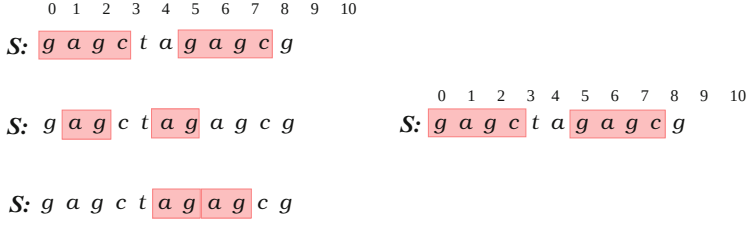


Fig. 4 Repeats of the string $S = gagctagagcg$. *Left*: three maximal repeated pairs of minimum length two. The pairs are $((0, 3), (6, 9))$, $((1, 2), (5, 6))$, and $((5, 6), (7, 8))$. *Right*: one supermaximal repeat of the substrings $(0, 3)$ and $(6, 9)$. The repeat ag is not supermaximal because it is not maximal and contained in *gagc*

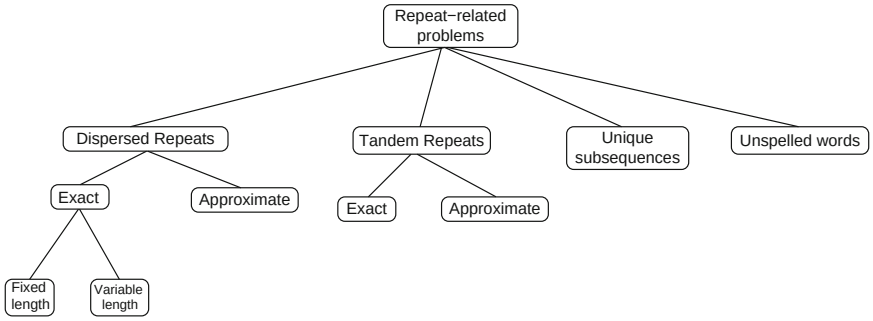


Fig. 5 Taxonomy of the major repeat-related problems in bioinformatics

If the repeated segments are occurring adjacent to each other, then we speak of tandem repeats. Formally, a substring of S is a *tandem repeat* if it can be written as $\omega\omega$ for some nonempty string ω. For example, the subtring $agag$ of the string $S = gagctagagcg$ is a tandem repeat, where $\omega = ag$.

Figure 5 is an overview of the basic string processing problems associated with repeat analysis. For biological sequences, approximate repeats (tandem or interspersed) are the ones occurring in reality. However, identification of exact repeats is also important because many algorithms that find approximate repeats are usually based on exact repeats. In this section, we will show how to solve the problems given in the figure.

4.1 Identifying Dispersed Repeats

Identifying Exact Fixed Length Repeats

The focus of this subsection is on the identification of repeats or repeated pairs of fixed length k, i.e., we find all repeats $\omega = S[i_1..j_1]$ such that $j_1 - i_1 + 1 = k$. Fixed length repeats of length k can be efficiently found using either the look-up table or the suffix tree. But to show the benefits of these data structures, we first briefly address a brute force method.

A brute force method to enumerate all repeats is to take each substring of S of length k and search whether it occurs somewhere else in S by scanning S from left to right. Because we have $O(n)$ substrings and each scan takes $O(nk)$ time (including character comparison), the brute-force method takes $O(kn^2)$ time. Note that by using a more advanced exact pattern matching algorithm (as will be discussed in Sect. 5.1), the running time of the scanning phase can be reduced to $O(k + n)$, which reduces the running time to $(n^2 + kn)$. But this is still quadratic and it is not so helpful.

The look-up table provides a better solution to this problem. If it is built for all substrings of length $d = k$ of S, then it becomes straightforward to extract all the repeats, as follows. The look-up table is sequentially scanned. If the linked

list attached to a cell of the table contains more than one element, then the positions where it starts in the string are reported. Repeated pairs can be computed by applying a Cartesian product operation over the linked list elements (excluding identical pairs). For example, if a k-length substring is repeated three time in S at positions 3, 7, and 20, then the repeated pairs are $((3, 3 + k - 1), (7, 7 + k - 1))$, $((3, 3+k-1), (20, 20+k-1))$, and $((7, 7+k-1), (20, 20+k-1))$. We do not need to report the pairs $((7, 7 + k - 1), (3, 3 + k - 1))$, $((20, 20 + k - 1), (3, 3 + k - 1))$, and $((20, 20 + k - 1), (7, 7 + k - 1))$, as well, because they are redundant.

Constructing the look-up table takes $O(|\Sigma|^d + n)$ time and $O(|\Sigma|^d)$ space. Finding the repeats takes $O(|\Sigma|^d + z)$ time, where z is the number of repeats. (This algorithm is said to be output sensitive, because the complexity is expressed in terms of the output size). Finding the repeated pairs takes $O(|\Sigma|^d + z')$ time, where z' is their number. It is not difficult to see that the space consumption of the look-up table becomes prohibitive for large d and Σ.

The automaton of Sect. 3.3 can also be used to report repeats of fixed length k. The idea is that the keys of the automaton are all the substrings of length k. If one leaf contains more than one position, these positions, which are the occurrences of a repeat, are reported. To find repeated pairs under this leaf, a Cartesian product operation is applied over the occurrences of the repeat (excluding identical pairs).

The suffix tree presents a more efficient solution to this problem. Recall from Sect. 3.4 that the concatenation of the edge-labels on the path from the root to a non-leaf node spells out a substring of S that occurs z times in S, where z is the number of all the leaves under the subtree of this non-leaf node. If each internal node is annotated with the length ℓ of the substring made up of the edge labels from the root to this node, then for each node with $\ell \geq k$ we report all the positions stored in the leaves of its subtree, as the starting positions of a repeated substring associated with this node. For example, if we use the suffix tree of Fig. 3 to find repeats of length two, then the nodes v_2, v_4, v_5, and v_6 are the ones with $\ell \geq 2$. For v_6, e.g., we report the positions 6 and 8 as the starting positions of the repeated string at, and compose the repeated pair $((6, 7), (8, 9))$.

Finding Variable Length Repeats

Maximal Repeated Pairs

The problem of reporting exact repeats of any length are referred to here as the problem of computing exact variable length repeats. The class of *maximal repeats* belongs to the set of variable length repeats. Recall from the introductory part of this section that a repeated pair $((i_1, j_1), (i_2, j_2))$ is called *maximal* if $S[i_1 - 1] \neq S[i_2 - 1]$ and $S[j_1 + 1] \neq S[j_2 + 1]$. That is, the repeated pair cannot extend to the left and to the right simultaneously in S. Recall also that a substring ω of S is a (*maximal*) *repeat* if there is a (maximal) repeated pair $((i_1, j_1), (i_2, j_2))$ such that $\omega = S[i_1..j_1]$. Fig. 4 (left) shows maximal repeated pairs of the string $S = gagctagagcg$.

To compute maximal repeated pairs, we use the algorithm of Gusfield based on the suffix tree [28, page 147]. This algorithm computes maximal repeated pairs of a sequence S of length n in $O(|\Sigma|n + z)$ time, where z is the number of maximal repeated pairs. To the best of our knowledge, Gusfield's algorithm was first used in the bioinformatics practice in the program *REPuter* [42], where maximal repeated pairs are used as seeds for finding approximate repeats, as we will discuss later when handling approximate dispersed repeats.

The algorithm for computing maximal repeated pairs is based on a bottom-up traversal of the suffix tree, in which all the child intervals are processed before the parent interval.

To explain the algorithm, we first introduce some notations: Let $\perp$ denote an undefined character, corresponding to a non-existing character before $S[0]$. We assume that this character is different from all characters in Σ. Let v be a node in the suffix tree and let ℓ_v be the length of the substring produced by concatenating the edge labels from the root to node v. Let u denote this substring. Define $\mathcal{P}_v$ to be the set of positions p such that u is a prefix of $S(p)$, i.e., $\mathcal{P}_v$ contains the positions attached to the leaves (suffixes) of the subtree at node v. We divide $\mathcal{P}_v$ into disjoint and possibly empty sets according to the characters to the left of each position: For any $a \in \Sigma \cup \{\perp\}$ define

$$\mathcal{P}_v(a) = \begin{cases} \{0 \mid 0 \in \mathcal{P}_v\} & \text{if } a = \perp \\ \{p \mid p \in \mathcal{P}_v, p > 0, \text{ and } S[p-1] = a\} & \text{otherwise} \end{cases}$$

Figure 6 shows an example of position sets associated to a suffix tree.

The set $\mathcal{P}_v$ is constructed during the bottom-up traversal as follows. If v is a leaf node of the suffix tree of S referring to the suffix $p = S(i)$. Then $\mathcal{P}_v = \{p\}$ and $\mathcal{P}_v(a) = p$ if $a = S[p-1]$. If v is not a leaf, then for each $a \in \Sigma \cup \{\perp\}$, $\mathcal{P}_v(a)$ is computed step by step while processing the children of v. Let $v_1, .., v_t$ denote the children of v. We start with v_2 and perform two operations:

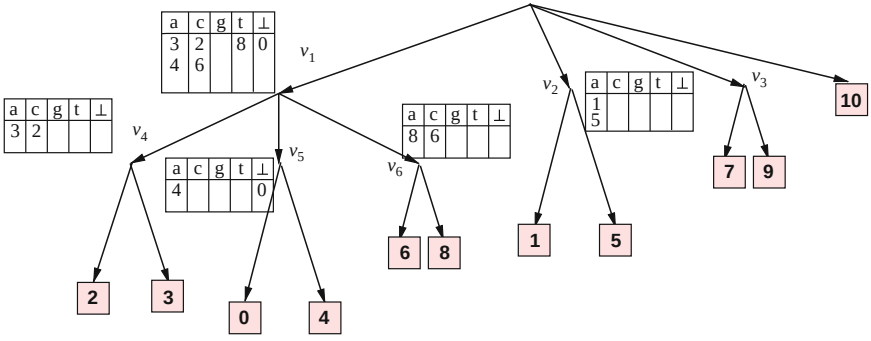


Fig. 6 The position sets of a sub-tree of the suffix tree of Fig. 3. We show these sets together just for illustration, but during the bottom up traversal the sets at v_4, v_5 and v_6, for example, coalesce together into the set at v_1

1. A Cartesian product between the sets $\mathcal{P}_{v_1}$ and $\mathcal{P}_{v_2}$. More precisely, maximal repeated pairs are output by combining the position sets $\mathcal{P}_{v_1}(a)$ with $\mathcal{P}_{v_2}(b)$, for each $a \neq b$. More precisely, a maximal repeated pair $((p, p + \ell_v - 1), (p', p' + \ell_v - 1))$, $p < p'$, is output for all $p \in \mathcal{P}_{v_1}(a)$ and $p' \in \mathcal{P}_{v_2}(b)$.
2. A union operation between $\mathcal{P}_{v_1}$ and $\mathcal{P}_{v_2}$ is performed to construct the position set $\mathcal{P}^{v_1 \cdot\cdot v_2}$, where $\mathcal{P}^{v_1 \cdot\cdot v_2}(a) = \mathcal{P}_{v_1}(a) \cup \mathcal{P}_{v_1}(a)$, for all $a \in \perp \cup \Sigma$.

Then we take v_3 and process it with $\mathcal{P}^{v_1 \cdot\cdot v_2}(a)$ to report *MEMs* and produce $\mathcal{P}^{v_1 \cdot\cdot v_3}(a)$, and so on until we process all the child intervals. The final set $\mathcal{P}^{v_1 \cdot\cdot v_t}$ is $\mathcal{P}_v$; see Fig. 6.

There are two operations performed when processing a node v: First, output of maximal repeated pairs by combining position sets. Second, union of position sets. The first step takes $O(z)$ time, where z is the number of repeats. The union operation for the position sets can be implemented in constant time, if we use linked lists. Altogether, the algorithm runs in $O(n + z)$ time. This algorithm requires linear space. This is because a position set $\mathcal{P}_v(a)$ is the union of position sets of the child intervals of v, and it is not required to copy position sets; rather they are linked together.

In practice, it is usually required to report maximal repeated pairs of length larger than a certain user-defined threshold. This reduces the output size while maintaining meaningful results. This length constraint can be incorporated in the above algorithm by examining for each internal node v the value ℓ_v. If ℓ_v falls below the given threshold, then the attached position set is discarded and one starts from the next non-visited leaf.

Identifying Supermaximal Repeats

A sub-class of the maximal repeats is the *supermaximal repeats*. A *supermaximal repeat* is a maximal repeat that never occurs as a substring of any other maximal repeat. Fig. 4 (right) shows supermaximal repeats of the string $S = gagctagagcg$.

We use the algorithm of [1] to compute supermaximal repeats. This algorithm was originally introduced over the enhanced suffix array but we will explain it here over the suffix tree. Let u_v denote the substring obtained by concatenating the characters on the path from the root to a node v. We call a node v *terminal node*, if all its children are leaves. To each leaf $S(p)$, we attach the character a such that $S[p - 1] = a$. We call this character the *burrows-wheeler* character, analogous to the burrows-wheeler transformation used in text compression.

Because any supermaximal repeat ω is right maximal, there should be a node v in the suffix tree such that $u_v = \omega$. Moreover, because ω does not occur as a substring of any other supermaximal repeat, the node v must be a terminal node and the burrows-wheeler characters of the leaves under v are pairwise distinct. The idea is that if these characters are not pairwise distinct, then there must be another repeat containing u_v as a substring. This suggests the following simple algorithm: Traverse each terminal node v of the suffix tree, and report the substring u_v if the respective *burrows-wheeler* characters are pairwise distinct. In Fig. 3, the terminal nodes are

v_2, v_3, v_4, v_5, and v_6. The condition on the burrows-wheeler character is satisfied for the nodes v_4, v_5, and v_6. Hence, the corresponding substrings aa, aca, and at are supermaximal repeats, respectively. The condition on the burrows-wheeler character is not satisfied for v_2 and v_3, because the suffixes 7 and 9 under v_2 are preceeded by the character a, and the suffixes 1 and 5 are preceeded by the character a. Because the number of terminal nodes and leaves is $O(n)$, this algorithm takes $O(n)$ time.

4.2 Identifying Tandem Repeats

A substring of S is a *tandem repeat*, if it can be written as $\omega\omega$ for some nonempty string ω. An occurrence of a tandem repeat $\omega\omega = S[p..p + 2|\omega| - 1]$ is *branching*, if $S[p + |\omega|] \neq S[p + 2|\omega|]$.

Stoye and Gusfield [56] described how all tandem repeats can be derived from branching tandem repeats by successively shifting a window to the left; see Fig. 7. They also showed that for each branching tandem repeat $\omega\omega$, there is one node in the suffix tree such that the substring obtained by concatenating the edge labels from the root to v is ω. For this reason, we focus on the problem of computing all branching tandem repeats using the suffix tree.

Let v denote a node in the suffix tree and let ω denote the substring obtained by concatenating the edge labels from the root to v. From each node v, one traverses the tree downwards and performs pattern matching, as will be explained in Sect. 5.1, to check if $\omega\omega$ occurs in S. Specifically, we search in the subtree at v to see if there is a path starting from v such that the concatenation of its edge labels spells out $\omega\alpha$, where α is any substring of S (α could be the empty string). For example, $\omega = at$ at the node v_6 in the suffix tree of Fig. 3. If we make a search in the subtree of this node, we find at on the path from this node to the leaf 6. Then we have an occurrence of a branching tandem repeat starting at position 6 in S and composed of the string $atat$.

The worst case running time of this algorithm is $O(n^2)$ (take, e.g., $S = a^n$). However, the expected length of the longest repeat of a string S is $O(\log n)$, where $n = |S|$. That is, $|\omega| = O(\log n)$ in average. Hence, the expected running time of the algorithm is $O(n \log n)$.

It is worth mentioning that there are algorithms that can compute tandem repeats in $O(n)$ time in the worst case; see [29, 40]. However, these algorithms are complicated and less efficient in practice than the algorithm described above, see [1].

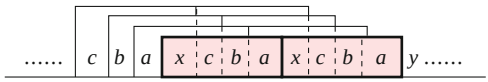


Fig. 7 Chain of non-branching tandem repeats $axcb$, $baxc$, and $cbax$, derived by successively shifting a window one character to the left, starting from the branching tandem repeat $xcba$

4.3 Identifying Unique Subsequences

The problem of finding all unique substrings is relevant for designing *oligonucleotides*. Oligonucleotides are short single-stranded DNA sequences. They are used either as *primers* in a PCR (Polymerase Chain Reaction) process to start the synthesis of a stretch of DNA having the complementary sequence of these oligonucleotides, or as *probes* to detect (and quantify) the existence of its complementary sequence in a pool of DNA sequences. These oligonucleotides should be unique to the DNA stretch they detect or synthesize. For example, if a certain oligonucleotide is used for the detection or isolation of a certain gene in a genome, then the complementary sequence should occur only in this gene, and nowhere else in the studied genome. Furthermore, the length of oligonucleotides should be within certain limits due to chemical and physical considerations (the length of a primer is between 18–30 and that of probes is between 4–10 base pairs). This leads to the definition of *unique substrings*.

Definition 1. A substring of S is *unique* if it occurs only once in S.

Definition 2. Given a string S of length n and two integers $0 < \ell_1 < \ell_2 \leq n$, the *unique substrings problem* is to find all unique substrings of S such that the length of each is between ℓ_1 and ℓ_2. The string S is a unique substring of length n.

For example, the strings ca, cac, aca, and $acac$ are unique substrings in $acac$. It is not difficult to verify that a unique substring in S is associated with a leaf in the suffix tree. In particular, if ω is a unique substring of S of length $\ell + 1$, then there is a leaf corresponding to the suffix $S(p)$ such that ω is a prefix of $S(p)$, and $u[\ell] \neq \$$.

To report each unique substring whose length ranges between ℓ_1 and ℓ_2, we perform a breadth-first traversal of the suffix tree. During the traversal, if the currently visited node v has $l(v) > \ell_2$, where $l(v)$ is the length of the substring obtained by concatenating the edge labels on the path from the root to v, then we visit no descendant nodes of v. If $\ell_1 \leq l(v) \leq \ell_2$ and v is a parent of a leaf corresponding to the suffix $S(p)$, then we report all unique matches starting at $S(p)$ in S and of length $l(v) < \ell \leq \ell_2$ only if $p + \ell < n$; i.e., we report the substrings $S[p..p + l(v)]$, $S[p..p + l(v) + 1], \ldots, S[p..p + \ell_2 - 1]$. For example, if it is required to report all unique matches of length $2 \leq l \leq 3$ of the string $acaaacatat$, whose suffix tree is given in Fig. 3, then the nodes v_2, v_3, v_4, and v_6 are considered. At v_2, for example, we report the substrings caa and cat.

It is not difficult to see that the time complexity of this algorithm is $O(n + z)$, where z is the number of unique matches.

4.4 Finding Absent Words

Absent words are strings not in S. It is clear that there is a prohibitively huge number of absent words. Therefore, we restrict ourselves to the shortest absent words. For short fixed-length absent words, say length $d < n$, we can use the look-up table

method in a straightforward way. We report the substrings corresponding to empty entries in the look-up table. For larger d, we can extend the algorithm for computing unique substrings using the suffix tree as follows. Let v be a node in the suffix tree, let u_v be the substring obtained by concatenating the edge labels on the path from the root to v. If $|u_v| < d$, then we produce all the substrings $u_v w$ such that $|u_v w| = d$ and $u_v w \notin S$, by computing all possible strings $w \in \Sigma^{|w|}$ and appending them to u_v.

4.5 Approximate Repeats

Computing approximate repeats can be achieved by using a local alignment algorithm in which the sequence is aligned to itself. The algorithm of Smith and Waterman that will be introduced in the next section for computing local alignments can be used in a straightforward manner to find approximate repeats. This algorithm takes $O(n^2)$ time, which is not feasible for long sequences. To overcome this, heuristic algorithms, addressed in the next section, were introduced to compute approximate repeats in a faster way.

4.6 Constraints on the Repeats

We address two types of constraints: (1) Constraints on the number of repeat occurrences and (2) proximity constraints.

Constraints on the number of occurrences

Posing constraints on the number of repeats or repeated pairs to be reported is beneficial for many applications in bioinformatics. As mentioned in the introduction, mammalian and plant genomes are full of repetitive elements of different kinds. Dispersed repeats are more abundant; they compose, for example, about 1.4 Gbp from the 3 Gbp human genome. From the different types of the dispersed repeats, three families are prevailing: LINE, SINE, and LTR. If a biologist is interested in locating these three families, then she/he searches for repeats appearing very often in the genome. Otherwise, she/he looks for non-abundant repeats.

In the following, we focus on posing an upper bound constraint on the number of repeats to be reported; posing a lower bound is complementary to what will be addressed here. We start with introducing the definitions of *rare* and *infrequent* maximal repeated pairs.

Definition 3. Let (l, p_1, p_2) denote a repeated pair $((p_1, p_1+l-1)(p_2, p_2+l-1))$, and let a threshold $t \in \mathbb{N}$ be given. A maximal repeated pair (l, p_1, p_2) is called *rare in* S if the string $w = S[p_1..p_1+l-1] = S[p_2..p_2+l-1]$ occurs at most $R_w \leq T$ times in S.

Definition 4. Let $(l, p_1^1, p_2^1), \ldots, (l, p_1^r, p_2^r)$ be maximal repeated pairs with $w = S[p_1^1..p_1^1 + l - 1] = \cdots = S[p_1^r..p_1^r + l - 1]$ and let a threshold $t \in \mathbb{N}$ be given. Any of these repeated pairs is called *infrequent* if $|\{(l^1, p_1^1, p_2^1), \ldots, (l^r, p_1^r, p_2^r)\}| = r_w \leq t$.

Definition 3 adds constraints on the number of repeated substrings that are identical to the substrings from which the maximal repeated pair stems, and Definition 4 adds constraints on the number of the maximal repeated pairs themselves. That is, Definition 4 complements Definition 3 by further controlling the number of repeats.

It is our next goal to calculate the values r_w and R_w. This can be achieved by modifying the algorithm introduced in Sect. 4.1 for computing maximal repeated pairs based on the suffix tree. We use the following notations. For a position set $\mathcal{P}_v$, let $C_{\mathcal{P}_v}(S, a) = |\mathcal{P}_v(S, a)|$ be the number of suffixes under the subtree at the node v and preceeded by character $a \in \Sigma$. Let $C_{\mathcal{P}_v}(S) = \sum_{a \in \Sigma \cup \{\perp\}} C_{\mathcal{P}_v}(S, a)$ be the total number of suffixes under the subtree of the node v. Clearly, $C_{\mathcal{P}_v}(S)$ is the number of occurrences of the substring $w = S[i \ldots i + \ell]$ in S, where $S(i)$ is one of these suffixes and ℓ is the length of the substring produced by concatenating the edge labels from the root to v. Hence, it is easy to see that the value $C_{\mathcal{P}_v}(S) = R_w$.

For any node v containing k child intervals, $v_1, .., v_k$, the value r_w can be calculated according to the following formula, where $q, q' \in [1..k]$:

$$r_w = \frac{1}{2} \sum_{a \in \Sigma \cup \{\perp\}} \sum_{q \neq q'} C_{\mathcal{P}_{v_q}}(S, a) \cdot (C_{\mathcal{P}_{v_{q'}}}(S) - C_{\mathcal{P}_{v_{q'}}}(S, a))$$

In the example of Fig. 6, the calculation of the value r_w for the subtree at node v_1 yields $r_w = (1 * 1 + 1 * 2) + (1 * 1 + 1 * 1) + (1 * 1 + 1 * 2) = 8$.

The values $C_{\mathcal{P}_v}(S)$ and $C_{\mathcal{P}_v}(S, a)$ can be attached to the position set of each node and can be inherited from the child nodes to their parent node. Therefore, adding these constraints to the algorithm for computing maximal repeated pairs mentioned before entails no increase in the time complexity. Note that the values R_w and r_w are considered before performing the Cartesian product operation. That is, if they are under the certain threshold then a Cartesian product is performed, otherwise we discard the whole position sets, and start from the next non-visited leaf.

Proximity constraint

Consider a repeated pair $R(l, i, j)$, where the two substrings composing it are $S[i..i + l - 1]$ and $S[j..j + l - 1]$, $i < j$. We call the former substring the *first instance* and the latter the *second instance*. We define the *span*[1] of R as the number of characters $(j - i - l)$ between these two substrings.

[1] In [17] the word "gap" was used instead of span. But because the word gap has a specific meaning in sequence alignment, which will be addressed later in this chapter, we will use the word span in this section.

Adding a proximity constraint on the reported repeated pairs can be formulated as "find each repeated pair in S such that its span is between two constants c_1 and c_2" [17]. It is not difficult to see that the tandem repeats are a special case of such constrained repeated pairs where the span is zero. To avoid redundant output, we focus on finding maximal repeated pairs with constraints on the span. In the following, we show how to incorporate this constraint into the algorithm presented before for computing maximal repeated pairs using the suffix tree.

Recall from the algorithm presented in Sect. 4.1 for computing maximal repeated pairs that the Cartesian product operation reports the z maximal repeated pairs in time $O(z)$. The straightforward idea is to filter out those pairs not satisfying the proximity constraint. This immediately yields $O(n + z)$ algorithm. But the number of the constrained maximal repeated pairs is much less than z. Therefore, Brodal et al. [17] introduced an algorithm that takes $O(n \log n + z')$ time, where z' is the number of the reported maximal pairs satisfying the constraint. The idea of their algorithm is to represent the position sets as AVL-trees. The advantage is that the pairs satisfying the constraints can be directly reported in time proportional to their number. The union operation between two position sets will no longer be performed in constant time, but it will take $O(\log n)$ time. Hence, we have the $\log n$ factor in the time complexity given above. Because the details are to a great extent involved, we refer the reader to the original paper of Brodal et. al. [17].

5 Sequence Comparison

Due to its numerous applications, the area of string comparison is of great importance to bioinformatics in particular and to computational science in general. Figure 8 is an overview and taxonomy of the basic string comparison problems. In exact string comparison, the goal is to examine if the strings or substrings in comparison are either identical or not, while in approximate string comparison we can tolerate some differences. For applications in molecular biology, approximate comparison is the one of interest due to sequencing errors and mutations. Nevertheless, exact comparison is also important because many algorithms that find approximate similarities are usually based on exact common substrings. Note that some of the problems in the figure are addressed also in other computer science areas such as information retrieval and text mining. It is worth mentioning that the field of string comparison is also referred to as *pattern matching*, in which one string to be compared is considered as a *text* and the other as a *pattern*. Hence, we might use the terms "pattern matching" and "string comparison" synonymously.

5.1 Global and Semi-Global Exact Pattern Matching

The global exact pattern matching is the task we usually encounter in many applications, where one is given two strings S_1 and S_2 and it is required to report whether

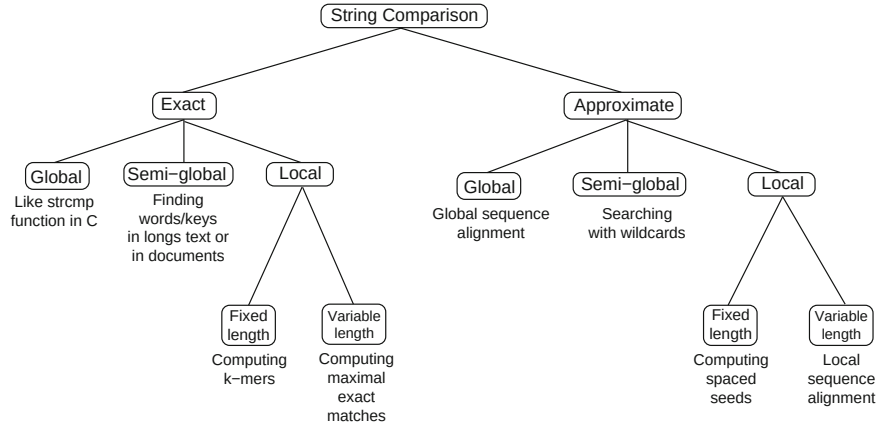


Fig. 8 A taxonomy of string comparison problems. We also show a real world application or problem for each type of string comparison

the two strings are identical or not. The algorithm is to compare the characters from left to right. If there is a mismatch, we report that $S_1 \neq S_2$ and terminate the comparison. If we reach the end of the strings, then we report that $S_1 = S_2$. The function `strcmp` in the programming language C, for example, is an implementation of this easy algorithm.

The semi-global exact pattern matching (known for short as exact pattern matching) problem is defined as follows. Given a string S of length n (usually called the *text*) and a pattern P of length $m \leq n$, the exact pattern matching problem is to find all the occurrences of P, if any, in S.

The brute force method takes $O(mn)$ time by sliding the pattern over S and comparing characters. This time complexity can be reduced to $O(n + m)$ if the pattern is pre-processed; the pre-processing time is $O(m)$ and the query time is $O(n)$. The algorithm of Knuth-Morris-Pratt, e.g., is one of these algorithms; see [28] for a simplified exposition and a survey of methods. Algorithms that pre-process the text itself, by constructing an indexing data structure for the text, achieve also $O(n + m)$ time; the pre-processing time is $O(n)$ and the query time is $O(m)$. But the advantage here is that the index is constructed once and millions of queries can be answered in time proportional to their length, independent of the length of S.

If it is required to search for patterns up to d characters, then one can use the look-up table presented in Sect. 3.2. The idea is to construct the look-up table over the text alphabet plus an additional dummy character, say "#." The reason for having this dummy character is to represent all the substrings of S of length up to d characters. If the dummy character were involved in the look-up table in Fig. 1, then we would have the extra entries corresponding to the substring $a\#$, $c\#$, $t\#$, $\#\#$. The last entry could be clearly not included in the table. The linked list for $a\#$ ($c\#$ or $t\#$) stores the positions where a (c or t) occurs in S.

If the pattern length is $m = d$, we directly compute the hashing function of the pattern, which maps us to a position in the look-up table. We then report the occurrences from the attached linked list. This takes $O(n + z)$ time, where z is the number of pattern occurrences in S. If the pattern length is $m < d$, then we append dummy characters so that the final pattern length becomes d. Thus, we can compute the hashing function and report the pattern occurrences. This takes also $O(d + z)$ time, which is not as optimal as the time complexity one can achieve using the suffix tree.

The exact pattern matching algorithm using the suffix tree works as follows. The suffix tree is traversed in a top-down fashion. We start from the root node and compare the edge labels of the outgoing edges to characters of the pattern. If there is a substring u labeling the edge from the root to a node v such that $P[0..|u-1|] = u$, then we move to node v and compare the sub-pattern $P[u..m-1]$ to the outgoing edge labels. This continues until there is a mismatch or all the characters of the pattern are compared successfully to the edge labels. In the former case the pattern does not occur in the text, while in the latter case the pattern exists and the positions where it occurs are the starting positions of the suffixes (leaves) in the subtree at the node on which the last edge traversed is incident. This procedure is illustrated in Fig. 3, where the gray arrows refer to the edges visited while searching for the pattern ac in S. This pattern starts at positions 0 and 4 in S.

5.2 Local Exact Matches

A similar region occurring in two strings S_1 and S_2 can be specified by the substrings $S_1[l_1 \ldots h_1]$, $S_2[l_2 \ldots h_2]$, where $1 \leq l_i < h_i \leq |S_i|$ and $i \in \{1, 2\}$. If $S_1[l_1 \ldots h_1] = S_2[l_2 \ldots h_2]$, i.e., the substrings are identical, then we have an *exact match*. This exact match can also be specified by the triple (ℓ, l_1, l_2), where $\ell = h_1 - l_1 = h_2 - l_2$. If the equality $S_1[l_1 - 1] = S_2[l_2 - 1]$ does not hold, then the exact match is called *left maximal*, i.e., it cannot extend to the left in the given strings. Similarly, if the equality $S_1[h_1 + 1] = S_2[h_2 + 1]$ does not hold, then the multiple exact match is called *right maximal*. The exact match is called *maximal*, abbreviated by *MEM*, if it is both left and right maximal. Fig. 9 (left) shows examples of maximal exact matches.

All the match types mentioned above can be computed using the exhaustive enumeration technique, which is one of the oldest techniques used to compute matches. It was first used in the software tool DIALIGN [46, 47] to compare protein and relatively short DNA sequences. The basic idea is that matches are the result of Cartesian products between all substrings of the sequences involved. In a sequence of average size n, we have $O(n^2)$ substrings. Thus, for two sequences, there are $O(n^4)$ possible matches. If the Cartesian products are limited to substrings of the same length, then the number of possible matches reduces to $O(n^3)$. To further reduce this number, only substrings of maximum length ℓ can be taken into account. This makes the number $O(n^2 l)$. Even with the $O(n^2 \ell)$ possible matches, this

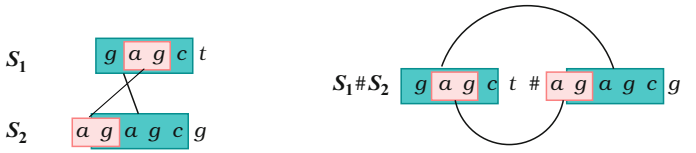


Fig. 9 Matches of the strings $S_1 = gagct$ and $S_2 = agagcg$. *Left*: two maximal exact matches of minimal length two. The first is $(4, 0, 1)$ composed of the substring "$gagc$," and the second is $(2, 1, 0)$ composed of the substring "ag." *Right*: these *MEMs* are maximal repeated pairs in $S_1\#S_2$

technique is still infeasible for comparing large sequences. In the following, better solutions will be presented. First, we will address efficient techniques for finding fixed length exact matches, where we report matches of fixed length $\ell = k$. Then we move to the computation of variable length matches, such as *MEMs*.

Finding Exact k-mers

Substring of S of fixed length k are called k-mers (also known as q-grams). If the matches reported are of certain fixed length, then they are called matches over k-mers. In this chapter, we call these matches, for short, k-mers.

The look-up table can be directly used to compute exact k-mers. We construct the table such that its entries contain the positions of all substrings of length k of one sequence, say S_1. Then the second sequence (say S_2) is streamed against this table to locate the k-mers. More precisely, each substring $S_2[i..i + k - 1]$ of length k of the second sequence is matched against this table, by computing the mapping function in Sect. 3.2. To save computation, we can use the recurrence in the same section to compute the mapping function of $S_2[i + 1..i + k]$ from $S_2[i..i + k - 1]$.

Constructing the look-up table for constant k takes linear time, specifically $O(n_1)$, where n_1 is the length of the first sequence. Deciding whether a query substring exists in the table takes $O(k)$ time and enumerating the matches, if any, takes time proportional to their number. Because there are $O(n_2)$ substrings in the second sequence, which is of length n_2, reporting all the z k-mers takes $O(n_2 + z)$ time. Therefore, the total time complexity is $O(n_1 + n_2 + z)$. The value k is usually chosen to be between 7 and 20, for genome size sequences, because the space consumption of the look-up table increases exponentially with it. Due to its simplicity, the look-up technique is widely used in software tools such as WABA [38], BLASTZ [51] (PipMaker [52]), BLAT [37], ASSIRC [58], GLASS [15], and LSH-ALL-PAIRS [18].

Automata can also be used to compute exact k-mers instead of constructing a look-up table. An automaton is constructed for one of the sequences, where its keys are are the set of the sequence substrings. The locations where these substrings occur in the sequence are attached to the leaves (nodes of success) of the automaton. For computing k-mers, each substring of length k of the other sequence is matched against this automaton.

We can also use the suffix tree for computing k-mers. But the suffix tree is too powerful to be used in this simple task, unless k is too large. We will explain in the next section how to use the suffix tree to compute variable length local matches, which is a more sophisticated task.

Maximal Exact Matches

Recall from the definitions given above that an exact match defined by the triple (l, p_1, p_2) is *maximal*, if $S[p_1 - 1] \neq S[p_2 - 1]$ and if $S[p_1 + l] \neq S[p_2 + l]$. It is not difficult to see that computing maximal exact matches between two strings S_1 and S_2 boils down to computing maximal repeated pairs of the string $S = S_1 \# S_2$, where # is a unique separator symbol occurring neither in the strings S_1 nor S_2. The triple (l, p_1, p_2) specifies a *MEM* if and only if $((p_1, p_1 + l - 1), (p_2, p_2 + l - 1))$ is a maximal repeated pair of the string S such that $p_1 + l - 1 < p < p_2$, where $p = |S_1|$ is the position of # in S.

Therefore, one can use the algorithm of Subsection 4.1 for computing maximal repeated pairs to compute maximal exact matches, but with the following modification. Each position set is further divided into two disjoint and possibly empty sets: One that contains all positions of the suffixes belonging to S_1 (these are smaller than $|S_1|$) and another one that contains all positions of the suffixes belonging to S_2 (these are greater than $|S_1|$). More precisely, we construct the set $\mathcal{P}_v(S_1, a)$ to contain all positions $p \in \mathcal{P}_v$ such that p corresponds to a position in S_1 and $S[p - 1] = a \in \Sigma \cup \{\perp\}$. $\mathcal{P}_v(S_2, a)$ is constructed analogously. (Figure 10 shows an example of position sets for computing *MEMs*.) To compute maximal exact matches, the Cartesian product is built from each position set $\mathcal{P}_v(S_1, a)$,

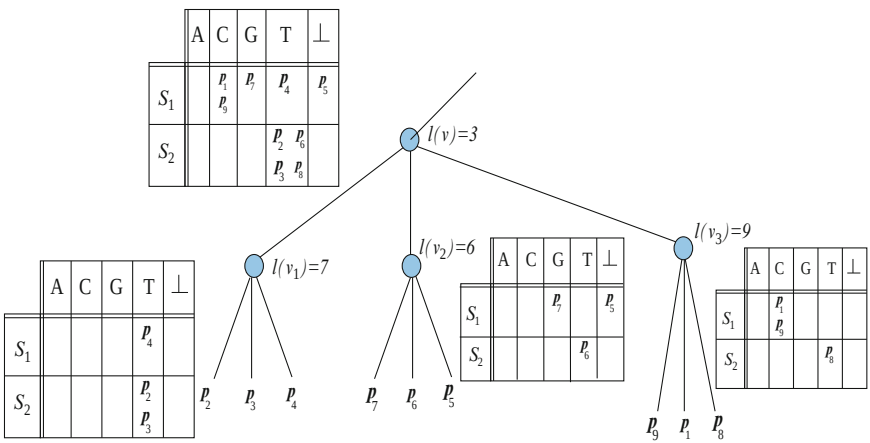


Fig. 10 The position sets of a part of the suffix tree. p_i denotes the suffix $S(p_i)$ starting at position p_i in S. We show these sets together just for illustration, but during the bottom up traversal the sets at v_1, v_2 and v_3 coalesce together into the set at v

$a \in \Sigma \cup \{\perp\}$ and the position sets $\mathcal{P}_v(S_2, b)$, where $a \neq b \in \Sigma \cup \{\perp\}$. It is not difficult to see that this modification does not affect the time and space complexity of the algorithm.

Constraints on the match occurrences

As we did before for maximal repeated pairs, we can add constraints on the number of matches and the substrings composing them. An important constraint is called *rare-MEMs*. A *MEM* is called *rare* if the constituent substrings $S_i[l_i \ldots h_i]$ appear at most T times in S_i, for any $i \in \{1, 2\}$ and T is a natural number specified by the user. A *MEM* is called *unique* if $T = 1$. In this case, we speak of a *maximal unique match*. In the example of Fig. 9 the *MEM* $(2, 1, 0)$ composed of the substring ag is not unique, because ag occurs more than one time in S_1 or S_2, while the *MEM* $(4, 0, 1)$ is unique.

The previous definitions given for maximal repeated pairs can be readily modified as follows. For a position set $\mathcal{P}$, let $C_{\mathcal{P}}(S_1, a) = |\mathcal{P}(S_1, a)|$ and $C_{\mathcal{P}}(S_1) = \sum_{a \in \Sigma \cup \{\perp\}} C_{\mathcal{P}}(S_1, a)$ (the values $C_{\mathcal{P}}(S_2, a)$ and $C_{\mathcal{P}}(S_2)$ are defined similarly). These values refer to the number of the constituent substrings of a maximal exact match at a node in the suffix tree. These values are considered before performing the Cartesian product operation. That is, if they fall under the given threshold, i.e., $C_{\mathcal{P}}(S_1, a) \leq T$ and $C_{\mathcal{P}}(S_2, a) \leq T$, then a Cartesian product is performed. Otherwise we discard the whole position sets, and start from the next non-visited leaf.

We can also have a constraint on the number of *MEMs* in addition to the number of the constituent substrings. For any node v, which contains k children, the number of *MEMs* associated with it (denoted by the value r_w) can be calculated according to the following formula, where q, q' are two child nodes:

$$r_w = \frac{1}{2} \sum_{a \in \Sigma \cup \{\perp\}} \sum_{q \neq q'} C_{\mathcal{P}_v^q}(S_1, a) \cdot (C_{\mathcal{P}_v^{q'}}(S_2) - C_{\mathcal{P}_v^{q'}}(S_2, a))$$

The Cartesian product operation is performed only if $r_w \leq t$, where t is a user defined threshold. Otherwise we discard the whole position sets, and start from the next non-visited leaf.

5.3 Approximate String Comparison and Sequence Alignment

Biological sequence comparison became a known practice since the emergence of biological sequence databases. There were technical as well as biological need that necessitated sequence comparison. The technical need was to avoid redundancy in the database, and the biological need was to infer the structure and function of new sequences by comparing them to those that already existed in the database. For example, two genes might have the same ancestor and function if they have

sufficiently similar sequences. Furthermore, the degree of similarity can indicate how long ago the two genes, and the organisms including them, diverged.

At that time, two or multiple such short sequences were compared on the character level (nucleotides in case of DNA/RNA and amino acids in case of proteins), and the result was delivered in terms of

1. *Replacements*, where characters in one sequence are replaced by other characters in the other sequence. If two characters replacing each other are identical, then they are called a *match*. Otherwise, they are called a *mismatch*.
2. *Indels*, where characters are inserted/deleted in either sequence. (A deletion in one sequence is an insertion in the other.)

In computer science terminology, character matches, mismatches, deletions, insertions are referred to as *edit operations*, and in biology they represent the mutation events.

The Global Alignment Algorithm

Computer scientists introduced sequence alignment algorithms to efficiently compare biological sequences in terms of these edit operations. If two sequences are pre-known to be similar, then *global sequence alignment* is the procedure to be used. Global alignment of two sequences is a series of successive edit operations progressing from the start of the sequences until their ends. This alignment can be represented by writing the sequences on two lines on the page, with replaced characters (matches/mismatches) placed in the same column and inserted/deleted characters placed next to a *gap*, represented by the symbol "_." In Fig. 11 (left), we show an example of aligning the two sequences ACTTAGTG and ACACCTG. From left to right, the series of edit operations is two matches of AC, one insertion (deletion) of T, three mismatches, and two matches of TG.

S_1: ACTTAGTG
xxx
S_2: AC_ACCTG

S		A	C	T	T	A	G	T	G
	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-1	1	0	-1	-2	-3	-4	-5	-6
C	-2	0	2	1	0	-1	-2	-3	-4
A	-3	-1	1	2	1	1	-1	-2	-3
C	-4	-2	0	1	2	1	1	0	-2
C	-5	-3	-1	-1	1	2	1	1	0
T	-6	-4	-2	-2	0	1	2	2	1
G	-7	-5	-3	-3	-1	0	2	2	3

Fig. 11 *Left*: Global alignment, where gaps correspond to indels. The positions where mismatches exist are marked by the symbol x. *Right*: The alignment table (known also as the dynamic programming matrix). The colored cells correspond to the traceback path that recovers the optimal alignment. For this alignment, the score of a match $\sigma = 1$, the cost of a mismatch $\alpha = 0$, and the gap cost $\gamma = -1$

If each of the edit operations has a certain score, then the alignment can be scored as the summation of the edit operation scores. In the example of Fig. 11 (left), if a match scores 1, a mismatch scores 0, and an indel has penalty of -1, then the score of the alignment is 4. In the literature, the indels penalties are also referred to as *gap costs*.

Because the number of possible alignments is very large, one is interested only in an alignment of highest score, which is referred to as *optimal global alignment*. In an optimal alignment, the amount of characters replacing each other (identical or similar characters) are maximized and the amount of gaps (insertions and deletions) is minimized. In 1970, Needleman and Wunsch [48] introduced a dynamic programming algorithm to find an optimal global sequence alignment.

Let two strings S_1 and S_2 be given, and let $A(i, j)$ denote the optimal score of aligning $S_1[0..i]$ to $S_2[0..j]$. We define $A(-1,-1) = 0$, $A(0,-1) = \gamma(S_1[i])$, and $A(-1, 0) = \gamma(S_2[j])$, where $\gamma(a)$ is the cost of deleting character a (gap cost). The following recurrence computes the optimal alignment.

$$A(i, j) = \max \begin{cases} A(i-1, j-1) + \delta(S_1[i], S_2[j]) \\ A(i, j-1) + \gamma(S_1[i]) \\ A(i-1, j) + \gamma(S_2[j]) \end{cases}$$

where δ is the score of replacing one character with another. This score can be based on a substitution matrix such as PAM or BLOSUM in case of comparing protein sequences. In other cases, and also here, we assume a fixed model, which is defined as follows

$$\delta(S_1[i], S_2[j]) = \begin{cases} \sigma \text{ if } S_1[i] = S_2[j] \\ \alpha \text{ otherwise} \end{cases}$$

This recurrence simply determines that the optimal alignment up to (i, j) in S_1 and S_2 can be computed based on the optimal alignments $A(i-1, j-1)$, $A(i, j-1)$, and $A(i-1, j)$. The first clause of the recurrence refers to the case where we replace $S_1[i]$ with $S_2[j]$. Hence, the optimal alignment assuming this replacement would be $A(i-1, j-1) + \delta(S_1[i], S_2[j])$. The second clause of the recurrence refers to the case where character $S_1[i]$ is deleted/inserted. Hence, the optimal alignment assuming this indel would be $A(i, j-1) + \gamma(S_1[i])$. Finally, the third clause of the recurrence refers to the case where character $S_2[j]$ is deleted/inserted. Hence, the optimal alignment assuming this indel would be $A(i-1, j) + \gamma(S_2[j])$. That is, we have three scores: 1) optimal alignment assuming replacement, 2) optimal alignment assuming indel in S_1, and 3) optimal alignment assuming indel in S_2. The ultimate optimal alignment at (i, j) is the one of the highest score among these three possible alignments. Fig. 11 (right) shows an example of the alignment table, in which $\sigma = 1$, $\alpha = 0$, and $\gamma = -1$.

The easiest way to compute this recurrence is to use a 2D table (known also as the dynamic programming matrix) and filling the cell (i, j) based on the cells $(i-1, j-1)$, $(i-1, j)$, and $(i, j-1)$. The optimal score is at the cell on the lower right corner. The alignment itself in terms of matches, mismatches, indels can be spelled out by tracing back the cells from $A(n, m)$ to $A(-1,-1)$. A backward diagonal move from $A(i, j)$ to $A(i-1, j-1)$ corresponds to match/mismatch between

$S_1[i]$ and $S_2[j]$, i.e., $S_1[i]$ matches/mismatches $S_2[j]$. A backward horizontal move from $A(i, j)$ to $A(i-1, j)$ corresponds to a gap in S_2, and an upward vertical move from $A(i, j)$ to $A(i, j-1)$ corresponds to a gap in S_1. The move direction itself is determined such that the score $A(i, j)$ equals either $A(i-1, j-1)+\delta(S_1[i], S_2[j])$, $A(i, j-1)+\gamma(S_1[i])$, or $A(i-1, j)+\gamma(S_2[j])$. The colored cells in the alignment table of Fig. 11 corresponds to the traced back cells, which recover the alignment on the left part of the figure. Multiple move possibilities imply that more than one optimal alignment exist.

The Needleman and Wunsch [48] algorithm takes $O(n^2)$ time and space. This complexity scales up to $O(n^k)$ in case of comparing k sequences of average length n.

Interestingly, it is possible to use only $O(n)$ space to compute the optimal score and to spell out the edit operations of an optimal alignment. This is based on the observation that computing the score at cell (i, j) depends only on the score at the three cells $(i-1, j-1)$, $(i, j-1)$, $(i-1, j)$. Hence, if we fill the array row-wise (column-wise), we need just to store the previous row (column) to compute the optimal score. Recovering the alignment itself by back tracing requires one additional trick, because the rows except for the last two ones of the matrix are no longer stored. This trick for a linear space dynamic programming algorithm is given in [32] and explained in [28].

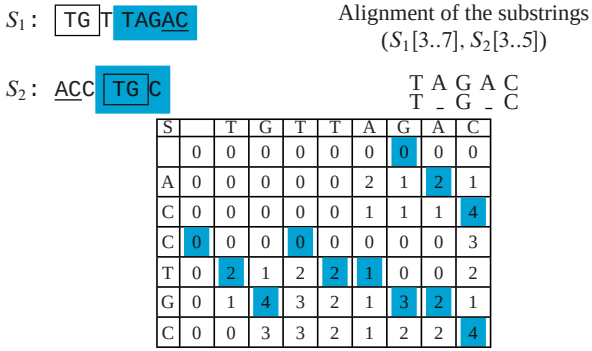
The Local Alignment Algorithm

In 1981, Smith and Waterman [54] introduced an essential modification to the Needleman and Wunch algorithm. Instead of computing an optimal global alignment, they computed local alignments of subsequences of the given sequences; their algorithm is, therefore, known as *local sequence alignment*. In Fig. 11 on the left, we show two sequences that are locally aligned. Note that the locally aligned regions are not in the same order in the two sequences.

Local alignments are important for three reasons: First, many biologically significant regions are subregions that are similar enough and the remaining regions are made up of unrelated sequences; the similar regions are known in proteins, for example, as *domains*. Second, insertions and deletions of any sizes are likely to be found as evolutionary changes; this is known as *domain insertions/deletions*. Third, the alignable subregions do not necessarily occur in the same order in the compared sequences; this corresponds to *domain shuffling*.

The algorithm of Smith and Waterman [54] is a very simple modification to the recurrences of the global alignment, but proofing the correctness of the recurrences is worth reading [54]. The modification is to use negative mismatch and gap costs and not allow any cell to have a negative score, which leads to the recurrence

$$A(i, j) = \max \begin{cases} A(i-1, j-1) + \delta(S_1[i], S_2[j]) \\ A(i, j-1) + \gamma(S_1[i]) \\ A(i-1, j) + \gamma(S_2[j]) \\ 0 \end{cases}$$



S		T	G	T	T	A	G	A	C
	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	2	1	2	1
C	0	0	0	0	0	1	1	1	4
C	0	0	0	0	0	0	0	0	3
T	0	2	1	2	2	1	0	0	2
G	0	1	4	3	2	1	3	2	1
C	0	0	3	3	2	1	2	2	4

Fig. 12 *Upper left*: Local alignment, where the underlined region in S_1 aligns to the underlined region in S_2, and the region in the rectangle in S_1 aligns to the one in rectangle in S_2. *Bottom*: The alignment table storing scores for local alignment. There are three possible highest scoring local alignments. The first is between the substrings ($S_1[0..1]$, $S_2[3..4]$), marked by *boxes*, see the bottom part of the figure. The second is between the substrings ($S_1[3..7]$, $S_2[3..5]$), marked by *colored boxes*, whose detailed alignment is in the *top right* part of the figure. The third is between the substrings ($S_1[6..7]$, $S_2[0..1]$), which are underlined. For this alignment, the match score $\sigma = 2$, the mismatch cost $\alpha = -1$, and the gap cost $\gamma = -1$

Regarding the complexity, the Smith–Waterman algorithm still takes $O(n^2)$ time; this complexity scales up to $O(n^k)$ in case of comparing k sequences of average length n. In Fig. 12 on the right, we show the alignment table for computing the local alignment of the two strings TGTTAGAC and ACCTGC. In this example, $\sigma = 2$, $\alpha = -1$, and $\gamma = -1$. A best local alignment ends at a cell of maximum score. The edit operations themselves can be recovered by back tracing, as we did before for global alignment. But in this case, we start from this highest scoring cell until reaching a cell of value zero. Note that there might be more than one cell of maximum score, implying that more than one local alignment over different subsequences exist. In the example of Fig. 12, we have three highest scoring alignments, two of them are overlapping. Recovering the best set of non-overlapping local alignments is an interesting problem addressed in the paper of Bafna et al. [13].

Semi-Global Sequence Alignment

Semi-global sequence alignment (usually known as approximate pattern matching) produces alignment that is global with respect to one sequence but local with respect to the other. For example, if we search for a short biological sequence S_1 in a much larger biological sequence S_2 then S_1 should be globally aligned to a subsequence of S_2. The following is a definition given by Gusfield [28] that specifies the semi-global sequence alignment problem.

Definition 5 ([28]). Given a parameter δ, a substring $S_2[l..h]$ of S_2 is said to be an approximate match of $S_1[0..n-1]$ if and only if the optimal alignment of S_1 to $S_2[l..h]$ has value at least δ.

To solve the semi-global alignment problem, we use the same recurrence for global sequence alignment, but we initialize the cells of the initialization row (row -1) with zeros (assuming that the horizontal axis of the table is arrayed with S_2). There is an occurrence of S_1 in S_2 ending at position j of S if and only if $A(n-1, j) \geq \delta$. In other words, each cell in the last row and in column j of the alignment matrix with score at least δ corresponds to an occurrence of S_1 ending at position j of S_2. To find the start position of an occurrence ending at position j of S_2 and to spell out the alignment, we trace back from the cell $(n-1, j)$ until we reach a cell in row -1. Assume the column number of the last traversed cell is l, then the occurrence starts at position $l+1$ in S_2. Fig. 13 shows an example, in which the score of a match $\sigma = 2$, the cost of a mismatch $\alpha = -1$, and the gap cost $\gamma = -1$.

Variations of Approximate Matching

Fixed-length local approximate matches

The notion of exact k-mers of Sect. 5.2 can be extended to allow some errors within the fixed-length match. There are two kinds of non-exact k-mers: The first restricts the allowed mismatches to fixed positions of the k-mer, but the second does not. The former kind of k-mers can be specified by a certain *pattern* or *mask* of ones and zeros. For example, it can be required to report all 8-mers such that the third and sixth characters are not necessary identical (do not care) in both sequences.

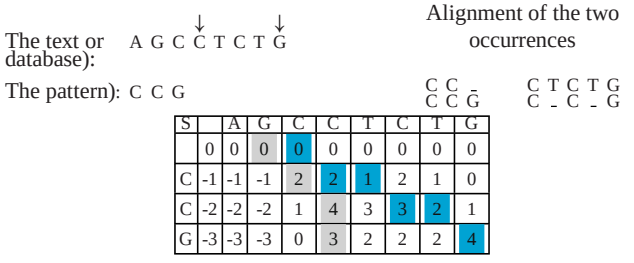


Fig. 13 *Upper left*: The given database (text) and the pattern. The *arrows* above the text indicates to where the matching pattern ends. *Top right*: Alignment of the two occurrences, recovered from the dynamic programming matrix given on the right. We report all occurrences with score larger than or equal to 3. The traceback starts from the cells with score larger than 3 and ends at the row -1. *Bottom*: The alignment matrix in which the cells of row -1 are initialized with zeros. The dark grey back tracing path corresponds to one occurrence of the pattern starting at position 3 in the text and the light grey path corresponds to another occurrence starting at position 2. For this alignment, the match scores $\sigma = 2$, a mismatch costs $\alpha = -1$, and a gap costs $\gamma = -1$

The respective mask is 11011011, where the ones correspond to the locations of the identical characters, and the zeros correspond to the locations where mismatches are allowed.

The look-up table can be used to compute k-mers according to a pre-determined mask, as follows. The look-up table is constructed such that its entries contain positions of substrings of the first sequence S_1, but the indices encode subsequences of these substrings. For a substring of length k, the respective subsequence is made up of the $d' < k$ characters that match the 1s of the mask. Let us take one example, suppose that S_1 contains the substring $w = aabc$ at position l_1. Assuming that $k = 4$ and the mask is 1101, then the index corresponding to w is computed according to the subsequence aac, and the entry next to it stores l_1. After constructing the look-up table, the second sequence is streamed against it as follows. For each substring of length k of the second sequence, one constructs the substring β, $|\beta| = d'$, containing the subsequence of characters that fit the 1s of the mask. For example, for a substring $aagc$ at position l_2, $\beta = aac$. If β is used as a query against the above look-up table, which has an entry for w, then the match (l_1, l_2, k) is obtained.

The look-up table was used first in PatternHunter [44] to locate non-exact k-mers according to a certain mask, and it was employed later in a recent version of BLASTZ [51].

The look-up table can also be used to find non-exact k-mers so that mismatches are allowed at any positions. Assume that it is required to generate non-exact k-mers such that at most one mismatch is allowed at any position of the k-mer. The look-up table is constructed as usual for the first sequence. For each substring of the second genome, the streaming phase has to be modified: Every character is replaced by every other character in the alphabet one at a time. Then the resulting substring is queried against the look-up table. In total, there are $k(|\Sigma|-1)+1$ queries launched for each substring: one for the original substring and $k(|\Sigma - 1|)$ for the replaced characters. If more than one mismatch is allowed in the same k-mer, then the queries are composed by considering all combinations of all replaced characters at every two positions of each substring.

Non-exact k-mers have been shown useful for comparing distantly related sequences. For example, a recent version of BLASTZ used them to investigate more diverged regions between human and mouse genomes.

Note that the look-up table based technique can be used for small and moderate k. For larger k, the suffix tree is to be used. We leave the details as an exercise for the reader.

Finding approximate tandem repeats

Recall from Sect. 4.2 that an exact *tandem repeat* is a substring of S that can be written as $\omega\omega$ for some nonempty string ω. An approximate tandem repeat can also be written as $\omega_1\omega_2$ such that ω_1 and ω_2 are not exact but are similar.

For each position i in the given sequence S, $0 \leq i < n$, we run a variation of the global alignment algorithm on the substrings $S[0..i]$ and $S[i+1..n]$ to find the

best alignment between a suffix of $S[0..i]$ and a prefix of $S[i+1..n]$, a variation known as suffix-prefix alignment. As we shall see in the next section, computing an optimal suffix-prefix alignment takes $O(n^2)$, the same as the global alignment problem. Therefore, the complexity of finding all approximate tandem repeats is $O(n^3)$ time and $O(n)$ space. It is worth mentioning that more sophisticated algorithms with better complexity already exist, see for example [36, 43].

Now we discuss how to compute an optimal suffix–prefix alignment. Let $S_{suf}[0..i] = S[0..i]$ and let $S_{prf}[0..n-i-2] = S[i+1..n-1]$. We use the same recurrence for global sequence alignment, but we initialize the cells of the initialization column (column -1) with zeros (assuming that the vertical axis of the table is arrayed with S_{suf}). An optimal suffix-prefix alignment ends at the highest scoring cell in the last row. To spell out the alignment, we trace back from this cell until we reach column -1. Assume that the highest scoring cell is at column l and we end the traceback procedure at row k, then the optimal prefix alignment is between $S_{suf}[k+1..i]$ and $S_{prf}[0..l]$. Fig. 14 shows an example of a suffix–prefix alignment.

It is worth mentioning that the suffix-prefix alignment plays an important role in genome assembly, where a whole genome is given as a set of substrings (called reads) and it is required to reconstruct it. The first step towards genome assembly is to solve the suffix-prefix alignment problem for each pair of reads. The pair of reads with high suffix–prefix alignment score are likely following each other in the genome. Assuming enough overlapping reads, the genome can be assembled by compiling the reads based on their suffix–prefix overlapping.

Heuristics for sequence alignment

Undoubtedly, alignment algorithms based on dynamic programming are invaluable tools for the analysis of short sequences. However, the running time of these

S_{prf} : TTTGACC

S_{suf} : AGCCTCTG

Alignment of S_{prf} and S_{suf}

```
A G C C T C T G
T T T G A - C C
```

S		A	G	C	C	T	C	T	G
	0	-1	-2	-3	-4	-5	-6	-7	-8
T	0	-1	-2	-3	-4	-2	-3	-4	-5
T	0	-1	-2	-3	-4	-2	-3	-1	-2
T	0	-1	-2	0	-1	-2	-3	-1	-2
G	0	-1	1	0	-1	-2	-3	-2	1
A	0	2	1	0	-1	-2	-3	-3	0
C	0	-1	1	3	2	1	0	-1	-1
C	0	-1	0	3	5	4	3	2	1

Fig. 14 *Upper left*: The two strings S_{suf} and S_{prf}. *Top right*: The best suffix-prefix alignment of S_{suf} and S_{prf}, recovered from the alignment matrix on the right. *Bottom*: The alignment table storing scores for the suffix-prefix alignment, in which the cells of column -1 are initialized with zeros. The highest score at the last row is 5. From the cell containing the highest score we trace back until reaching column -1. For this alignment, a match scores $\sigma = 2$, a mismatch costs $\alpha = -1$, and a gap costs $\gamma = -1$

algorithms renders them unsuitable for large scale and high-throughput comparisons. In the following, we will classify and address the basic heuristics for fast sequence alignment.

All the heuristics that speed-up the alignment process are based on what is called the *filtering technique*, see [5] for a survey. The idea of this technique is to first generate matches, usually exact ones. Then the regions containing no matches are not processed (filtered out), and the regions containing matches are further processed to produce detailed alignment. There are two basic techniques to process the matches: seed-and-extend and chaining.

Seed-and-extend techniques

The seed and extend strategy determines regions of local similarity by extending each match (seed) to the left and to the right by applying a standard sequence alignment algorithm (based on dynamic programming) starting from the terminals of the seed. This extension continues until the score falls under a certain user defined threshold. An extended seed whose score is less than a certain threshold is discarded. Seed extension is carried out to filter out seeds that appeared by chance and are not part of a shared and inherited (homologous) region. Fig. 15 (left) illustrates the strategy of single seed extension. To avoid redundancy, a seed is extended only if it belongs to no region that is a result of an extension of another seed.

Some tools start the extension only if there are two matches occurring near each other; this is referred to as *double seed extension*, see Fig. 15 (right). The reason for this is to reduce the number of false positive extensions and speed up the alignment process. The idea of double seed extension was suggested first in Gapped-BLAST [11] for searching sequence databases, and was later used in BLAT [37] for searching EST databases.

The seed and extend strategy is suitable for finding short local similarities with higher sensitivity. However, to start from a single seed and to extend it to a large region of a genome is very time consuming. In other words, this would be nothing but a standard dynamic programming algorithm on the character level except that it starts from the seed positions in the sequences. Therefore, this strategy is recommended for finding relatively short similarities of the genomes in comparison.

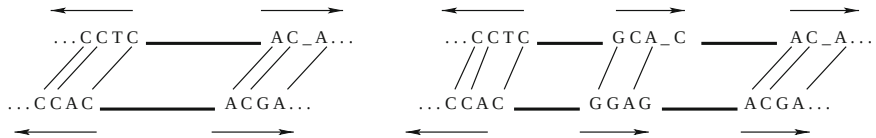


Fig. 15 Seed-and-extend technique. *Left*: single seed extension. *Right*: double seed extension. The bold *solid lines* are the seeds. The *arrows* show the direction of extension

Chaining techniques

Chaining techniques became the strategy of choice for comparing large genomes. This is because of their flexibility for various tasks, and their ability to accommodate large gaps. A chaining algorithm does not handle one match (or two) at a time. Rather, it tries to find a set of colinear non-overlapping matches according to certain criteria. This can be viewed as a generalization of the seed and extend strategy. Two matches are said to be colinear and non-overlapping, if the end positions of one match is strictly before the start positions of the second in all genomes. For example, in Fig. 16 (left) the two matches 2 and 3 are colinear and non-overlapping, while the matches 1 and 3 are not colinear, and the matches 1 and 2 are overlapping. (In the literature of chaining algorithms, the matches are referred to as *fragments* or *blocks*.)

Geometrically, a match f composed of the substrings $(S_1[l_1..h_1], S_2[l_2..h_2])$ can be represented by a rectangle in $\mathbb{N}^2_{\geq 0}$ with the two extreme corner points $beg(f) = (l_1, l_2)$ and $end(f) = (h_1, h_2)$. In Fig. 17, we show the matches of Fig. 16 (left) represented as two dimensional rectangles in a 2D plot.

For constructing a global alignment, a *global* chain of colinear non-overlapping matches is constructed. This global chain starts from the beginning of the sequences until their ends. The matches of the resulting global chain comprise what is called *the anchors*. These anchors are fixed in the alignment and the regions between the anchors are aligned using standard dynamic programming. The rationale is that if the two sequences share high similarity, then the anchors usually cover most of the sequences and the remaining parts to be aligned on the character level become very short. In Fig. 16 (left), we show an example of a global chain.

For constructing local alignments, sets of local chains of colinear non-overlapping matches are computed according to certain criteria. The score of each chain is the summation of the lengths of the involved matches minus the distances between the successive matches in the chain. (The distance between a match f

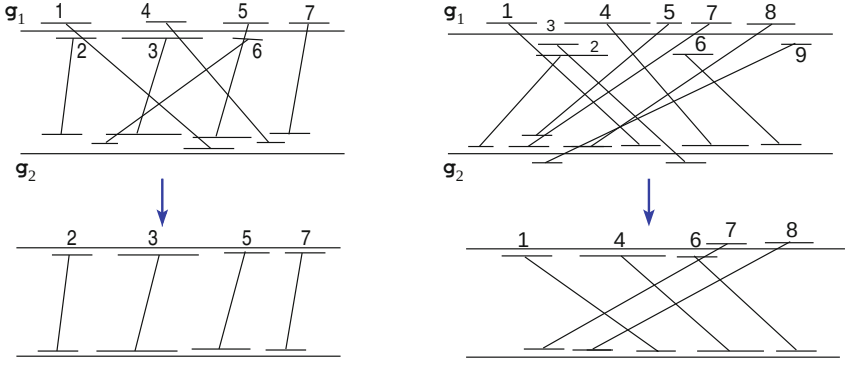


Fig. 16 *Left*: global chaining. The global chain is composed of the matches 2, 3, 5, and 7. *Right*: local chaining. There are two local chains. The first is composed of the three matches 1, 4, and 6, and the second is composed of the two matches 7, and 8

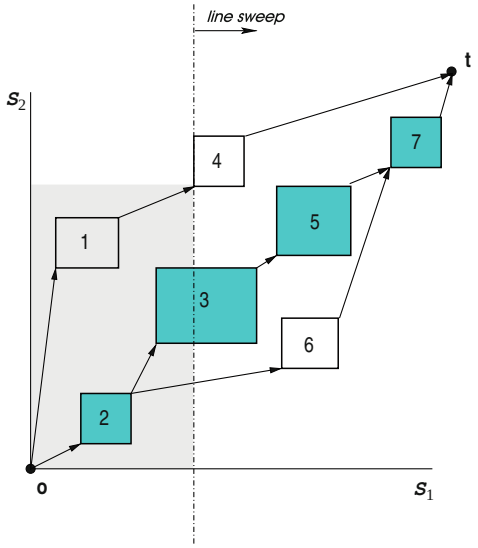


Fig. 17 Computation of a highest-scoring global chain of colinear non-overlapping matches. The search area (for finding the highest scoreing chain) at match 4 is bounded by the gray rectangle. The chain $\langle 2, 3, 5, 7\rangle$ is an optimal global chain, whose matches are plotted in a different color

followed by another match f' in a chain is the geometric distance between $end(f)$ and $beg(f')$ either in the L_1 or L_∞ metric.) Each local chain of a significant score (defined by the user) corresponds to a region of local similarity, and the matches of each local chain are the anchors for each local alignment. The regions between the anchors in each local chain are aligned on the character level to produce the final local alignment. In Fig. 16 (right), we show examples of local chains. Note that we might have more than one significant local chain, in an analogous way to the traditional local alignment on the character level.

The exact chaining algorithms (local and global) can be sub-divided into two groups according to the underlying computational paradigm/model: graph based and geometric based. In this chapter, we will briefly handle the global chaining problem. For the local chaining problem and other variations for comparative genomics, we refer the reader to [2, 3].

In the graph-based paradigm, the global chaining problem is solved as follows. A directed acyclic graph is constructed so that the matches are the nodes and the edges are added according to the colinearity and non-overlapping constraint. The edges are weighted, for example, by the length or statistical significance of the matches. The optimal solution is then reduced to finding an optimal path in this graph. In Fig. 17, we show an example of a graph constructed over the set of matches of Fig. 16 (left) (not all edges are drawn), and an optimal path in this graph.

The graph based solution takes $O(m^2)$ time to find an optimal solution, where m is the number of matches. This can be a severe drawback for large m. Fortunately, the complexity of the chaining algorithm can be improved by considering the geometric nature of the problem. A sweep line based algorithm can be used and it works as follows: The matches are first sorted in ascending order w.r.t. one se-

quence, and processed in this order to simulate a line scanning the points. During the line sweep procedure, if an end point of a match is scanned, then it is activated. If a start point of a match is scanned, then we connect it to an activated match of highest score occurring in the rectangular region bounded by the start point of the match and the origin, see Fig. 17. Details of this algorithm, which is subquadratic in time can be found in [4].

The repeat finding heuristics

It is worth mentioning that the seed-and-extend strategy and the chaining techniques can also be used to find local approximate repeats, in which the sequence is compared to itself. To take two examples, the program *REPuter* [42] first computes maximal repeated pairs then uses the seed-and-extend strategy to obtain approximate repeated pairs. The program CoCoNUT [2] uses a variation of the chaining technique over (rare) maximal repeated pairs to find the large segmental duplications in a genomic sequence. For a survey about methods and tools for repeat analysis, we refer the reader to [30].

6 Applications of String Mining in Other Areas

The string mining data structures and algorithms presented in this chapter are generic and provide efficient solutions to many problems in both exploratory and predictive data mining tasks both in bioinformatics and beyond. In this section, we review some of these applications. In particular, we show how the trie data structure is used in a popular algorithm for frequent itemset mining, a problem that is typically associated with business applications. We also describe how efficient string comparison methods can be used in computing string kernels required by data mining techniques in different applications. We also describe how sequence alignment algorithms have been used for pattern mining in unstructured (free text) and semi-structured (e.g., web-based) documents.

6.1 Applying the Trie to Finding Frequent Itemsets

The *association rule mining* problem is concerned with finding relations between database items. For example, given a set of purchased items in a supermarket, it would be useful to derive an assumption (rule) in the form “customers, who buy milk, also buy bread.” Finding frequent itemsets is a basic step in association rule mining, in which the items bought frequently and together seem to be associated by

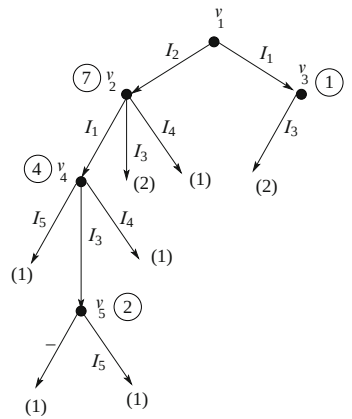
an implication rule. A related *sequence rule mining* problem is concerned with finding similar rules where the order of the transactions is important, e.g., "customers, who buy milk, also buy bread at a later date."

In this section we focus on the problem of finding frequent itemsets in a database of transactions, and we follow the definitions and notations introduced in [8]. Let $\mathcal{I} = \{I_1, I_2, \ldots, I_m\}$ be a set of items. Let $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$ be a set of transactions, where each transaction T_i is made up of a set of items, i.e., $T_i \subset \mathcal{I}$. An *itemset* $IS \subseteq \mathcal{I}$ is a set of items. The *frequency* of an itemset *IS*, denoted by $F(IS)$, is the number of transactions containing *IS* as a subset. An itemset *IS* is called *frequent*, if $F(IS) > \tau$, where τ is a given threshold called *minimum support count*. A frequent itemset is called maximal if it is not a subset of another frequent itemset.

Figure 18 shows an example of transactions. For $\tau = 2$, the frequent itemsets with more than one element are $\{I_1, I_2\}$, $\{I_2, I_5\}$, $\{I_1, I_5\}$, $\{I_1, I_3\}$, $\{I_2, I_3\}$, $\{I_2, I_4\}$,

Transaction	Items	Sorted items
T_1	I_1, I_2, I_5	I_2, I_1, I_5
T_2	I_2, I_4	I_2, I_4
T_3	I_2, I_3	I_2, I_3
T_4	I_1, I_2, I_4	I_2, I_1, I_4
T_5	I_1, I_3	I_1, I_3
T_6	I_2, I_3	I_2, I_3
T_7	I_1, I_3	I_1, I_3
T_8	I_1, I_2, I_3, I_5	I_2, I_1, I_3, I_5
T_9	I_1, I_2, I_3	I_2, I_1, I_3

Item	Freq.
I_2	7
I_1	6
I_3	6
I_4	2
I_5	2



Item	FIS
I_5	$(\{I_2, I_5\}, 2)$, $(\{I_1, I_5\}, 2)$, $\underline{(\{I_2, I_1, I_5\}, 2)}$
I_4	$\underline{(\{I_2, I_4\}, 2)}$
I_3	$(\{I_2, I_3\}, 4)$, $(\{I_1, I_3\}, 2)$, $\underline{(\{I_2, I_1, I_3\}, 2)}$
I_1	$(\{I_2, I_1\}, 2)$

Fig. 18 A number of transactions over the set $\{I_1, I_2, I_3, I_4, I_5\}$ of items. The second column contains the items of each transaction sorted with respect to their count in the given dataset, which is shown in the table on its right. The FP-tree is shown on the right, where the numbers between the brackets are the frequency of each key (transaction). The circled numbers are the internal node counts. The table below shows the frequent itemsets (FIS) with minimum support 2. The maximal frequent itemsets are underlined

$\{I_1, I_2, I_3\}$, and $\{I_1, I_2, I_5\}$. The maximal frequent itemsets are $\{I_1, I_2, I_3\}$, $\{I_1, I_2, I_5\}$, and $\{I_2, I_4\}$. It is easy to see that reporting the maximal frequent itemsets only limits the redundancy without sacrificing meaningful results.

Naive solutions for finding frequent itemsets that are based on enumerating all possible combinations of candidate itemsets and then evaluating their support and confidence values are combinatorially expensive. Efficient solutions to the problem based on the *a priori principal* were developed in the mid nineties by Agarwal et al. [8] to prune candidate sets that have an infrequent sub-pattern.

An alternative solution to the problem that avoids candidate generation was developed by Han et al. [31]. This is known as the *FP-growth* algorithm and is one of the most efficient algorithms for finding the frequent itemsets [31]. The algorithm is based on constructing a data structure called *FP-tree* which is nothing but a slight variation of the trie data structure presented before in Sect. 3.2.

The FP-growth algorithm starts with counting how many times each item occurs in the transaction list, i.e., we compute $F(i)$, for each $i \in \mathcal{T}$. The second table in the example of Fig. 18 shows the frequency of each item. The items in each transaction are then sorted in decreasing order w.r.t. to their frequency, see the second column in the leftmost table of Fig. 18. Then a trie is constructed for all the transactions, where each transaction is considered as a key. If one key is a prefix of another key, then we add a dummy branching edge labeled with an empty character $''\_''$. For computing the frequent itemsets, we attach with each leaf the number of occurrences of the associated key, and for each node we attach the summation of the numbers attached to all the leaves in its subtree. In the sequel, we call this attached number the *node-count*, see Fig. 18. It is not difficult to see that the FP-tree can be constructed in $O(mn)$.

The FP-growth algorithm computes the frequent itemsets by processing the edges associated with the less frequent items before those associated with the more frequent items. Suppose that the edges associated with the items $I_{r_1}..I_{r_{x-1}}$ are already processed, where $F(I_{r_1}) < \cdots < F(I_{r_x})$. Assume that the edges associated with the item I_{r_x} are being currently processed, where $F(I_{r_{x-1}}) < F(I_{r_x})$. Then we run the following three steps.

1. We locate the internal nodes or the leaves such that the edges annotated with I_x are incident to them. Let the set $V = \{v_1, v_2, ..v_t\}$ denote such nodes.
2. For each node $v_i \in V$, $1 \leq i \leq t$, with node count greater than the minimum support count, we report the set $H \cup I_x$ as a frequent itemset, where the set H is made up of the edge labels of the nodes on the path from the root to the node v_i. Moreover, we report the sets $H' \cup I_x$, for each $H' \subset H$. Note that the set $H \cup I_x$ is the maximal frequent itemset and the sets $H' \cup I_x$ are non-maximal.
3. For each subset $V' \subset V$ such that $|V' > 1|$, we locate the node z in the tree such that z is the farthest node from the root and all the nodes of V' are descendant of z (z is called least common ancestor node of the nodes of V'). If the total node count of the nodes of V' in the subtree of z is greater than the minimum support count, then we report the set $H' \cup I_x$ as a frequent itemset, where the set H' is made up of the edge labels of the nodes on the path from the root to the node z. Moreover, we report the non-maximal frequent itemsets $H'' \cup I_x$, for each $H'' \subset H'$.

We take the edges associated with the next less frequent item after I_x and repeat the three steps mentioned above. Iteratively, we repeat this procedure until no more edges are left unprocessed. In the example of Fig. 18, where the minimum support is two, we start with the edges associated with item I_5, which is the least frequent item. There are two edges labeled with I_5 and incident to two leaves. Each leaf has node count less than the minimum support. Hence, we ignore them. The least common ancestor node of these two leaves is the node v_4, and the total node count of the two leaves associated with I_5 is 2, which satisfies the minimum support. Hence, we report the set $\{I_2, I_1, I_5\}$ as a maximal frequent itemset and the sets $\{I_1, I_5\}$ and $\{I_2, I_5\}$ as frequent itemsets.

6.2 Application of String Comparison to a Data Mining Technique: Computing String Kernels

The classifiers used in many problems in machine learning may have to work on discrete examples (like strings, trees, and graphs) [59]. Because the discrete examples, if not strings, can be readily reduced to strings, string kernels are used to measure the similarity between two strings, and accordingly between the two structures. Roughly speaking, the measure of similarity in string kernels is the total number of all common substrings between two input strings. This is made precise in the following.

Let $num_u(v)$ denote the number of occurrences of substring v in string u and let $\Sigma^\star$ denote the set of all possible non-empty strings over the alphabet Σ. The kernel $k(S, R)$ on the two strings S and R is defined as follows:

$$k(u, v) = \sum_{v \in \Sigma^\star} (num_S(v) \times num_R(v))\omega(v)$$

where $\omega(v) \in \mathbb{R}$ weighs the number of occurrences of each string v in both S and R.

A crucial observation that makes the kernel computation straightforward is that maximal exact matches form a compact representation of all the substrings in both S and R. That is, any substring occurring in S and R should be a substring of a maximal exact match of S and R. This suggests the following two phases algorithm:

1. Count the number of all maximal exact matches of length ℓ. For this purpose, we can use the procedure used for handling constraints on the number of maximal exact matches *MEMs*. Specifically, we accumulate the r_w values associated with a *MEM* of length ℓ, Let T_ℓ denote occurrences of the ℓ length *MEMs*.
2. For each ℓ, the number of matches included in it with length $1 \leq h \leq l$ equals $\ell(\ell + 1)T_\ell/2$.
3. Summating over all the lengths of maximal exact matches, we obtain $k(u, v)$.

By using the suffix tree data structure, it is not difficult to see that computing the string kernel takes $O(|S| + |R|)$ time and space.

6.3 String Mining in Unstructured and Semi-Structured Text Documents

Many of the string mining algorithms presented in this chapter form the basis for techniques used in pattern mining over both free text documents and semi-structured documents. For example Rigoustos and Huynh [50] describe a system that learns patterns and keywords that typically appear in spam emails. However, rather than simply looking for the exact string patterns in new emails, their approach applies a sequence similarity search similar to the one used in comparing protein sequences. Such an approach can discover variations of the strings including deletions and insertions of characters as well as substitutions of individual characters. With such an approach, having identified the string "Viagra" as a spam string, strings such as "Viagra," "V|agra," "Vigra" and "v I a G r A" would all be automatically scored as similar with a high score. Similar concepts form the basis for identifying plagiarism in both free text documents as well semi-structured documents such as program code; see for example [14, 22]. Another typical application of string mining algorithms is in the automatic extraction of information from web pages. In such applications particular textual information of interest to the user, e.g., specifications of a particular product or its price, is embedded within the HTML tags of the document. These tags provide the overall structure of the document as well as its formatting. Manual extraction of the required information is typically easy for humans based on both the semantic clues (e.g., the meaning of keywords such as price) as well as formatting cues (titles, tables, itemized lists, images, etc.). Identifying and extracting the same information using automatic methods is, however, more challenging. One reason is that the web page layout is typically designed for human users rather than information extraction robots; the proximity of visual clues and cues on the 2-dimensional page layout does not directly translate to proximity at the string level or even at the HTML structure level. A second reason is that different products could be described differently on each web page; e.g., some information might be missing or described in slightly different ways on each web page.

A family of approaches typically known as *wrapper induction* methods [55] are used to address the problem. For systems based on these approaches, a user provides a set of different example web pages and annotates the target information he would like to extract from them. The system then uses machine learning methods to learn patterns that best describe the target information to be extracted and then generates rules that can be used by specialized extractors to extract such information. The patterns learned from each example patterns are typically strings of tokens that represent both HTML tags as well regular text. The string mining algorithms presented in this chapter form the basic components for addressing some of the key challenge for such applications. For example, these wrapper induction tools typically need to generalize multiple identification patterns into a single patterns. This can be achieved by using the basic alignment algorithms, or variations

that align multiple sequences (e.g., CLUSTALW [57]). For example, given the two patterns listed below:

```
String 1 : <div> <b> <br> <span> <br> a </br> <br>
target <br><br>
String 2 : <div> <b> <br> <span> <br> target> <br><br>
```

the system would first align both patterns indicating the missing tokens

```
String 1 : <div> <b> <br> <span> <br> a <br> target
<br><br>
 String 2 : <div> <b> <br> <span> <br> - -
target <br><br>
```

and then generate a generalized pattern. where * can match any number of tokens.

```
String 2 : <div> <b> <br> <span> <br> * target
<br><br>
```

Examples of using such alignment methods, and also methods for identifying and dealing with repeated patterns within the context of information extraction from web sources, are described in Chang et al. [20]. Examples of using other string mining techniques for context-specific searching in XML documents are described in [23].

7 Conclusions

In this chapter we addressed the basic data structures and algorithms used in analyzing biological sequences. We also explored the application of these techniques in other areas of data mining.

The look-up table is an easy to manipulate data structure. But its space consumption increases exponentially with the key size. A recent method that is suitable for our data, which is static in nature, is the perfect hashing technique [16, 24]. In this technique, the keys can be stored in linear space and the constant access time is still guaranteed. Applying this technique to biological sequences is addressed in [6].

For string comparison based on the suffix tree, the strategy was to concatenate the sequences to be compared and reduce the problem to a variation of the problem of finding repeats. This strategy requires to construct the suffix tree for the two sequences, which might be infeasible for large sequences. The method of Chang and Lawler [21], which was introduced for computing matching statistics, can be used to overcome this problem. The idea of their method is to construct the index of only one sequence (usually the shorter one) and matches the second against it. To achieve linear time processing, they augmented the suffix tree with extra internal links called *suffix links*.

The suffix tree is an essential data structure for sequence analysis, but its space consumption and cash performance is a bottleneck for handling large genomic sequences. The enhanced suffix array is a recent alternative that overcomes these problems, see the original paper of the enhanced suffix array [1] and the recent paper with theoretical fine tuning of it [6].

There are a number of packages implementing the algorithms in this chapter. The strmat [39] package implements many of the algorithms in this chapter based on the suffix tree. The Vmatch [41] and GenomeTools [27] implement many of the repeat analysis and sequence comparison algorithms based on the enhanced suffix array. Other packages in this regard include SeqAn [26] and mkESA [34].

References

1. M.I. Abouelhoda, S. Kurtz, and E. Ohlebusch. Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2(1):53–86, 2004.
2. M.I. Abouelhoda, S. Kurtz, and E. Ohlebusch. CoCoNUT: An efficient system for the comparison and analysis of genomes. *BMC Bioinformatics*, 9:476, 2008.
3. M.I. Abouelhoda and E. Ohlebusch. A local chaining algorithm and its applications in comparative genomics. In *Proceedings of the 3rd Workshop on Algorithms in Bioinformatics*, volume 2812 of *LNBI*, pages 1–16. Springer Verlag, 2003.
4. M.I. Abouelhoda and E. Ohlebusch. Chaining algorithms and applications in comparative genomics. *Journal of Discrete Algorithms*, 3(2-4):321–341, 2005.
5. M.I. Abouelhoda. *Algorithms and a software system for comparative genome analysis*. PhD thesis, Ulm University, 2005.
6. M.I. Abouelhoda and A. Dawood. Fine tuning the enhanced suffix array. In *Proceedings of 4th Cairo International Biomedical Engineering Conference, IEEExplore, DOI:10.1109/CIBEC.2008.4786047*, 2008.
7. M.I. Abouelhoda, S. Kurtz, and E. Ohlebusch. The enhanced suffix arrays. In Srinivas Aluru, editor, *Handbook of Computational Molecular Biology (Chapter 7)*. Chapman & Hall/CRC Computer and Information Science Series, 2006.
8. R. Agarwal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of ACM-SIGMOD*, pages 207–216, 1993.
9. A. Aho and M. Corasick. Efficient string matching: An aid to bibliographic search. *Communications of the ACM*, 18:333–340, 1975.
10. S.F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. A basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.
11. S.F. Altschul, T. L. Madden, A. A. Schäffer, and et al.˙Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.
12. A. Apostolico. The myriad virtues of subword trees. In *Combinatorial Algorithms on Words*, NATO ISI, pages 85–96. Springer-Verlag, 1985.
13. V. Bafna, B. Narayanan, and R. Ravi. Nonoverlapping local alignments (weighted independent sets of axis-parallel rectangles). *Discrete Applied Mathematics*, 71:41–53, 1996.
14. B.S. Baker. On finding duplications and near duplications in large software systems. In *Proceedings of the 2nd Working Conference on Reverse Engineering*, page 86, 1995.
15. S. Batzoglu, L. Pachter, J. P. Mesirov, B. Berger, and E. S. Lander. Human and mouse gene structure: Comparative analysis and application to exon prediction. *Genome Research*, 10:950–958, 2001.

16. F.C. Botelho, R. Pagh, and N. Ziviani. Simple and space-efficient minimal perfect hash functions. In *Proceedings of the 10th International Workshop on Algorithms and Data Structures*, volume 4619 of *LNCS*, pages 139–150. Springer, 2007.
17. G. Brodal, R. Lyngso, Ch. Pedersen, and J. Stoye. Finding maximal pairs with bounded gap. *Journal of Discrete Algorithms*, 1(1):134–149, 2000.
18. J. Buhler. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, 17(5):419–428, 2001.
19. T. Schiex C. Mathe, M.F. Sagot and P Rouze. Current methods of gene prediction, their strengths and weaknesses. *NAC*, 30(19):4103–17, 2002.
20. C.H. Chang, C.N. Hsu, and S.C. Lui. Automatic information extraction from semi-structured web pages by pattern discovery. *Decision Support Systems*, 35:129–147, 2003.
21. W.I. Chang and E.L. Lawler. Sublinear approximate string matching and biological applications. *Algorithmica*, 12(4/5):327–344, 1994.
22. X. Chen, B. Francia, M. Li, B. McKinnon, and A. Seker. Shared information and program plagiarism detection. *IEEE Transactions Information Theory,*, 50:1545–1550, 2004.
23. K.T. Claypool. SUSAX: Context-specific searching in xml documents using sequence alignment techniques. *Data & Knowledge Engineering*, 65(2):177 – 197, 2008.
24. Z. Czech, G. Havas, and B.S. Majewski. Fundamental study perfect hashing. *Theoretical Computer Science*, 182:1–143, 1997.
25. A. L. Delcher, A. Phillippy, J. Carlton, and S. L. Salzberg. Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Research*, 30(11):2478–2483, 2002.
26. A. Döring, D. Weese, T. Rausch, and K. Reinert. SeqAn an efficient, generic c++ library for sequence analysis. *BMC Bioinformatics*, 9:9, 2008.
27. G. Gremme, S. Kurtz, and et. al. GenomeTools. http://genometools.org.
28. D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, New York, 1997.
29. D. Gusfield and J. Stoye. Linear time algorithms for finding and representing all the tandem repeats in a string. Report CSE-98-4, Computer Science Division, University of California, Davis, 1998.
30. B.J. Haas and S.L. Salzberg. Finding repeats in genome sequences. In T. Lengauer, editor, *Bioinformatics - From Genomes to Therapies*. Wiley-VCH, 2007.
31. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of ACM-SIGMOD*, pages 1–12, 2000.
32. D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM* , 18:341–343, 1975.
33. M. Höhl, S. Kurtz, and E. Ohlebusch. Efficient multiple genome alignment. *Bioinformatics*, 18:S312–S320, 2002.
34. R. Homman. mkESA - enhanced suffix array construction tool. `http://bibiserv.techfak.uni-bielefeld.de/mkesa/`.
35. R.L. Dunbrack Jr. Hidden markov models in bioinformatics. *Current Opinion in Structural Biology*, 16(3):374–84, 2006.
36. S. Kannan and E.G. Myers. An algorithm for locating non-overlapping regions of maximum alignment score. *SIAM Journal on Computing*, 25(3):648–662, 1996.
37. W.J. Kent. BLAT—the BLAST-like alignment tool. *Genome Research*, 12:656–664, 2002.
38. W.J. Kent and A.M. Zahler. Conservation, regulation, synteny, and introns in large-scale *C. briggsae–C. elegans* genomic alignment. *Genome Research*, 10:1115–1125, 2000.
39. J. Knight, D. Gusfield, and J. Stoye. The Strmat Software-Package. `http://www.cs.ucdavis.edu/ gusfield/strmat.tar.gz`.
40. R. Kolpakov and G. Kucherov. Finding maximal repetitions in a word in linear time. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 596–604. IEEE, 1999.
41. S. Kurtz. The Vmatch large scale sequence analysis software. http://www.vmatch.de.
42. S. Kurtz, J. V. Choudhuri, E. Ohlebusch, C. Schleiermacher, J. Stoye, and R. Giegerich. REPuter: The manifold applications of repeat analysis on a genomic scale. *Nucleic Acids Research*, 29(22):4633–4642, 2001.

43. G. Landau, J. Schmidt, and D. Sokol. An algorithm for approximate tandem repeats. *Journal of Computational Biology*, 8(1):1–18, 2001.
44. B. Ma, J. Tromp, and M. Li. PatternHuner: Faster and more sensitive homology search. *Bioinformatics*, 18(3):440–445, 2002.
45. E. M. McCreight. A space-economical suffix tree construction algorithm. *Journal of Algorithms*, 23(2):262–272, 1976.
46. B. Morgenstern. DIALIGN 2: Improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, 15(3):211–218, 1999.
47. B. Morgenstern. A space efficient algorithm for aligning large genomic sequences. *Bioinformatics*, 16(10):948–949, 2000.
48. S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino-acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.
49. L. Pichl, T. Yamano, and T. Kaizoji. On the symbolic analysis of market indicators with the dynamic programming approach. In *Proceedings of the 3rd International Symposium on Neural Networks 2006*, LNCS 3973, 2006.
50. I. Rigoutsos and T. Huynh. Chung-Kwei: a Pattern-discovery-based System for the Automatic Identification of Unsolicited E-mail Messages (SPAM). In *Proceedings of the 1st Conference on Email and Anti-Spam (CEAS)*, 2004.
51. S. Schwartz, W. J. Kent, A. Smit, Z. Zhang, R. Baertsch, R. C. Hardison, D. Haussler, and W. Miller. Human-mouse alignments with BLASTZ. *Genome Research*, 13:103–107, 2003.
52. S. Schwartz, Z. Zhang, K. A. Frazer, A. Smit, C. Riemer, J. Bouck, R. Gibbs, R. Hardison, and W. Miller. PipMaker–a web server for aligning two genomic DNA sequences. *Genome Research*, 10(4):577–586, 2000.
53. R. Sherkat and D. Rafiei. Efficiently evaluating order preserving similarity queries over historical market-basket data. In *The 22nd International Conference on Data Engineering (ICDE)*, page 19, 2006.
54. W. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Computational Biology*, 147:195–197, 1981.
55. S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, 1999.
56. J. Stoye and D. Gusfield. Simple and flexible detection of contiguous repeats using a suffix tree. *Theoretical Computer Science*, 270(1-2):843–856, 2002.
57. J. Thompson, D. Higgins, and Gibson. T. CLUSTALW: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties, and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.
58. P. Vincens, L. Buffat, C. Andrè, J. Chevrolat, J. Boisvieux, and S. Hazout. A strategy for finding regions of similarity in complete genome sequences. *Bioinformatics*, 14(8):715–725, 1998.
59. S.V.N. Vishwanathan and A.J Smola. Fast kernels for string and tree matching. In K. Tsuda, editor, *Kernels and Bioinformatics*. Cambridge, MA: MIT Press, 2004.
60. P. Weiner. Linear pattern matching algorithms. In *Proceedings of the 14th IEEE Annual Symposium on Switching and Automata Theory*, pages 1–11, 1973.

“This page left intentionally blank.”

Part III
Data Mining and Knowledge Discovery

“This page left intentionally blank.”

Knowledge Discovery and Reasoning in Geospatial Applications

Nabil Sahli and Nafaa Jabeur

1 Introduction

In the last decade and following the big advent in remote sensing, terabytes of geographic data have been collected on a daily basis. However, the wealth of geographic data cannot be fully realized when information implicit in data is difficult to discern. This confronts researchers with an urgent need for new methods and tools for automatically transform geographic data into information and, furthermore, synthesize geographic knowledge. New approaches in geographic representation, query processing, spatial analysis, and data visualization [1, 2] are thus necessary.

Geo-referenced data sets include geographical objects, locations or administrative sub-divisions of a region. The geographical location and extension of these objects define implicit relationships of spatial neighborhood. Knowledge Discovery (KD) in databases is a process that aims at the discovery of relationships within data sets. Data Mining, the focus of this chapter, is the central step of this process. It corresponds to the application of algorithms for identifying patterns within data. The Data Mining algorithms have to take this spatial neighborhood into account when looking for associations among data. They must evaluate if the geographic component has any influence in the patterns that can be identified or if it is responsible for a pattern. Most of the geographical attributes normally found in organizational databases (e.g., addresses) correspond to a type of spatial information, namely qualitative, which can be described using indirect positioning systems. In systems of spatial referencing using geographic identifiers, a position is referenced with respect to a real world location defined by a real world object. This object represents a location that is identified by a geographic identifier (more details can be found in Chapter Spatial Techniques). These geographic identifiers are very common in organizational databases, and they allow the integration of the spatial component associated with them in the process of knowledge discovery.

Unfortunately, classical data mining algorithms often make assumptions (e.g., independent, identical distributions) which violate Tobler's first law of Geography:

N. Sahli (✉)
Dhofar University, Salalah, Sultanate of Oman
e-mail: nabil_sahli@du.edu.om

M.M. Gaber (ed.), *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*, DOI 10.1007/978-3-642-02788-8_10,

everything is related to everything else but nearby things are more related than distant things [3]. Applying traditional data mining techniques to geospatial data can result in patterns that are biased or that do not fit the data well. Chawla et al. [4] highlight three reasons that geospatial data pose new challenges to data mining tasks: "First, classical data miningdeals with numbers and categories. In contrast, spatial data is more complex and includes extended objects such as points, lines, and polygons. Second, classical data mining works with explicit inputs, whereas spatial predicates (e.g., overlap) are often implicit. Third, classical data mining treats each input to be independent of other inputs whereas spatial patterns often exhibit continuity and high auto-correlation among nearby features." Besides, current database techniques use very simple representations of geographic objects and relationships (e.g., point objects, polygons, and Euclidean distances).

Chawla et al. [4] suggest that data mining tasks should be extended to deal with the unique characteristics intrinsic to geospatial data. Data structures, queries, and algorithms need to be expanded to handle other geographic objects (e.g., moving objects) and relationships (e.g., non-Euclidean distances) [2]. One of the most serious challenges is integrating time into database representations. Integrating geospatial data sets from multiple sources (with varied formats, semantics, precision, and coordinate systems) is another big challenge. The aforementioned challenges explain the recent arise of a new discipline called Spatial (or Geospatial) Data Mining [4–6].

Spatial data mining is a subfield of data mining. It is concerned with the discovery of interesting and useful but implicit knowledge in spatial databases. Sources of geospatial data include satellite imagery, cartographic maps, census data, and modeling runs of spatial differential equations. Similarly, [7] defines spatial data mining as the extraction of knowledge, spatial relations, or other interesting patterns not explicitly stored in spatial databases. As for the term "Geospatial" in Geospatial Data Mining (GDM), it refers to geographic data. "Geographic" refers to the specific case where the data objects are geo-referenced and the embedding space relates (at least conceptually) to locations on or near the Earth's surface.

In this chapter, we first enumerate the most important characteristics of spatial data that will be mined. We then briefly describe some well-known techniques (namely, clustering, classification, association rules, and outlier detection) of GDM. Few applications of GDM in the real life are also illustrated. Finally, we present the main challenges that still to be faced in order to better support geospatial-temporal applications in general and data mining in particular.

2 Data Characteristics in Geospatial Data Mining

The characteristics that make geospatial data "special" as a computing problem have been acknowledged in many other writings such as [8] and in what follows a brief summary of these characteristics (non-exhaustive list) is presented.

The first characteristic is that geospatial data repositories are very large. Moreover, existing GIS datasets usually consists of feature and attribute components

which are archived in hybrid data management systems. Algorithmic requirements differ substantially for relational (attribute) data management and for topological (feature) data management [9]. Knowledge Discovery's procedures have also to be diversified in order to achieve all the operational goals within a geospatial computing environment. The range and diversity of geographic data formats are also challenging constraints. In fact, the traditional "vector" (spatial objects in an embedding space) and "raster" (discrete representations of continuous spatial fields using a grid-cell tessellation) formats are no longer the only data types in use. New types of data formats have been proposed the last few years and which go beyond the traditional formats. New sensors (e.g., satellites) are nowadays collecting such new ill-structured data such as imagery and geo-referenced multi-media. geo-referenced multi-media include audio, imagery, video, and text that can be related unambiguously to a location on the Earth's surface based on the location of the data collection or its content [10]. This data structure rises up the complexity of the discovering knowledge process [11].

Another characteristic of geospatial data relates to phase characteristics of data collected cyclically. Data discovery must accommodate collection cycles that may be unknown (e.g., identifying the cycles of shifts in major geological faults) or that may shift from cycle to cycle in both time and space (for example the dispersion patterns of a health epidemic, or of toxic waste).

Contrarily to other KD applications, geospatial–temporal applications involve up to four interrelated dimensions of the information space. This additional unique property of geographic data requires thus special consideration and techniques. For example, some geographic phenomena such as travel times within urban areas or mental images of geographic space and disease propagation over space and time [12, 13], have properties that are non-Euclidean.

Spatial dependency and spatial heterogeneity are also other properties of geographic data. The former refers to the tendency of observations that are more proximal in geographic space to exhibit greater degrees of similarity or dissimilarity (depending on the phenomena). The latter refers to the non-stationarity of most geographic processes, meaning that global parameters do not reflect well the process occurring at a particular location. While these properties have been traditionally treated as nuisances, recent geographic information technologies have provided tools that can exploit these properties for new insights into geographic phenomena (e.g., [14, 15]). In [4], it is even claimed that ignoring these properties could affect the patterns derived from data mining techniques.

Finally, another unique aspect of geographic information is the complexity of spatio-temporal objects and patterns. While non-geographic data objects can be meaningfully represented discretely without losing important properties, geographic objects cannot necessarily be reduced to points or simple line features without information loss. Relationships such as distance and direction are also more complex with dimensional objects [16, 17]. Including time introduces additional complexity. A simple strategy that treats time as an additional spatial dimension is not sufficient. Time has different semantics than space. Indeed time is directional, has unique scaling and granularity properties, and can be cyclical and even branching with parallel

local time streams [18]. Spatial transformations are also complex: for example, a formal model of possible transformations in spatial objects with respect to time includes the operators create, destroy, kill, reincarnate, evolve, spawn, identity, aggregate, disaggregate, fusion and fission. Nevertheless, these transformations are information-bearing [19]. Roddick and Lees [18] also argue that the potential complexity of spatio-temporal patterns may require meta-mining techniques that search for higher-level patterns among the large number of patterns generated from spatio-temporal mining.

3 Spatial Data Mining Techniques

The first option towards solving spatial data mining problems is to use classical data mining tools which assume independence between different data points. However, classical data mining methods often perform poorly on spatial data sets which have high spatial auto-correlation. In this section, we briefly review the classical techniques and then we focus on the specialized spatial data mining techniques which can effectively model the notion of spatial-autocorrelation.

3.1 Classical Data Mining Techniques

Relationships among spatial objects are often implicit. It is possible to materialize the implicit relationships into traditional data input columns and then apply classical data mining techniques. Many generic algorithmic strategies have been generalized to apply to spatial data mining. For example, as shown in Table 1, algorithmic strategies, such as divide-and-conquer, Filter-and-refine, ordering, hierarchical structure, and parameter estimation, have been used in spatial data mining. Few approaches have proposed to use classical data mining algorithms in GDM. For example, in [20], the authors presented an approach for knowledge discovery in geo-referenced databases, based on qualitative spatial reasoning principles, where the location of geographic data was provided by qualitative identifiers. Direction, distance and topological spatial relations were defined for a set of municipalities (in Portugal). This knowledge and the composition table constructed for integrated spatial reasoning, about direction, distance and topological relations, allowed for the inference of new spatial relations analyzed in the data mining step of the knowledge discovery process. The results obtained with the new system point out that traditional KDD systems, which were developed for the analysis of relational databases and that do not have semantic knowledge linked to spatial data, can be used in the process of knowledge discovery in geo-referenced databases, since some of this semantic knowledge and the principles of qualitative spatial reasoning are available as domain knowledge. The main advantages of such approaches include the use of already existing data mining algorithms developed for the analysis of non-spatial data;

Table 1 Algorithmic strategies in spatial data mining [21]

Generic	Spatial Data Mining
Divide-and-Conquer	Space Partitioning
Filter-and-Refine	Minimum-Bounding-Rectangle(MBR)
Ordering	Plane Sweeping, Space Filling Curves
Hierarchical Structures	Spatial Index, Tree Matching
Parameter Estimation	Parameter estimation with spatial autocorrelation

an avoidance of the geometric characterization of spatial objects for the knowledge discovery process; and the ability of data mining algorithms to deal with geospatial and non-spatial data simultaneously, thus imposing no limits and constraints on the results achieved.

3.2 Specialized Spatial Data Mining Techniques

Another way to deal with implicit relationships is to use specialized spatial data mining techniques. We highlight four of the most common, namely, clustering, classification, association rules, and outlier detection.

Clustering

Clustering is a well established technique for data interpretation. It usually requires prior information, e.g., about the statistical distribution of the data or the number of clusters to detect. "Clustering" attempts to identify natural clusters in a data set. It does this by partitioning the entities in the data such that each partition consists of entities that are close (or similar), according to some distance (similarity) function based on entity attributes. Conversely, entities in different partitions are relatively far apart (dissimilar). An example of cluster is shown in Fig. 1.

Existing clustering algorithms such as K-means [22], PAM [23], CLARANS [24], DBSCAN [25], CURE [26], and ROCK [27] are designed to find clusters that fit some static models. For example, K-means, PAM, and CLARANS assume that clusters are hyper-ellipsoidal or hyper-spherical and are of similar sizes. The DBSCAN assumes that all points of a cluster are density reachable [25] and points belonging to different clusters are not. However, all these algorithms can breakdown if the choice of parameters in the static model is incorrect with respect to the data set being clustered, or the model did not capture the characteristics of the clusters (e.g., size or shape). Because the objective is to discern structure in the data, the results of a clustering are then examined by a domain expert to see if the groups suggest something. For example, crop production data from an agricultural region may be clustered according to various combinations of factors, including soil type, cumulative rainfall, average low temperature, solar radiation, availability of irrigation, strain of seed used, and type of fertilizer applied. Interpretation by a domain expert

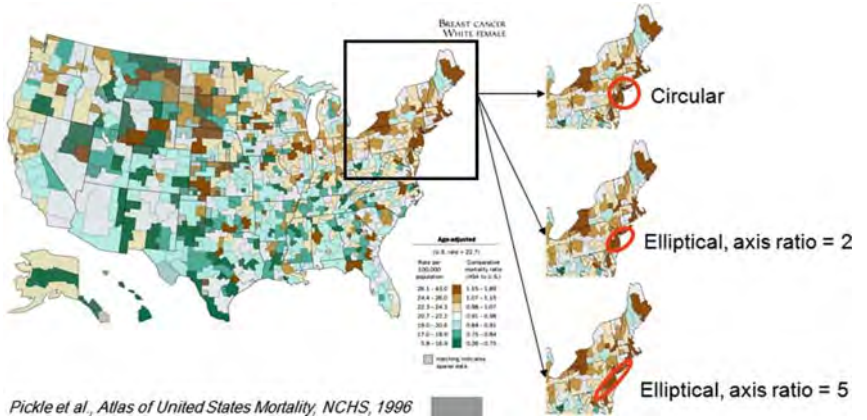


Fig. 1 Examples of likely cluster of breast cancer mortality rates in the USA

is needed to determine whether a discerned pattern- such as a propensity for high yields to be associated with heavy applications of fertilizer-is meaningful, because other factors may actually be responsible (e.g., if the fertilizer is water soluble and rainfall has been heavy). Many clustering algorithms that work well on traditional data deteriorate when executed on geospatial data (which often are characterized by a high number of attributes or dimensions), resulting in increased running times or poor-quality clusters [28]. For this reason, recent research has centered on the development of clustering methods for large, highly dimensioned data sets, particularly techniques that execute in linear time as a function of input size or that require only one or two passes through the data. Recently developed spatial clustering methods that seem particularly appropriate for geospatial data include partitioning, hierarchical, density-based, grid-based, and cluster-based analysis. Hierarchical methods build clusters through top-down (by splitting) or bottom-up (through aggregation) methods. Density-based methods define clusters as regions of space with a relatively large number of spatial objects; unlike other methods, these can find arbitrarily-shaped clusters. Grid-based methods divide space into a raster tessellation and clusters objects based on this structure. Model-based methods find the best fit of the data relative to specific functional forms. Constraints-based methods can capture spatial restrictions on clusters or the relationships that define these clusters.

Classification

Whereas clustering is based on analysis of similarities and differences among entities, “classification” constructs a model based on inferences drawn from data on available entities and uses it to make predictions about other entities. For example, suppose the goal is to classify forest plots in terms of their propensity for landslides. Given historical data on the locations of past slides and the corresponding environmental attributes (ground cover, weather conditions, proximity to roads and streams,

land use, etc.), a classification algorithm can be applied to predict which existing plots are at high risk or whether a planned series of new plots will be at risk under certain future conditions. Spatial classification algorithms determine membership based on the attribute values of each spatial object as well as spatial dependency on its neighbors. Various classification methods have been developed in machine learning, statistics, databases, and neural networks; one of the most successful is decision trees. Concerning machine learning methods, and in [29], the approach of [30] [1] is generalized through a spatial classification learning algorithm that considers spatial relationships defined as path relationships among objects in a defined neighborhood of a target object. As for the decision trees method, authors in [31] apply a decision tree induction algorithm to extracting knowledge about complex soil-landscape processes. Their system combines background knowledge in the form of a soil survey map with other environmental data to extract the expert's judgments underlying the subjective map. In [32], the authors use a type of artificial neural known as adaptive resonance theory networks to extract knowledge from a remotely sensed imagery.

Association Rules

"Association rules" attempt to find correlations (actually, frequent co-occurrences) among data. For instance, the association rules method could discover a correlation of the form "forested areas that have broadleaf hardwoods and occurrences of standing water also have mosquitoes." In Fig. 2, Karasova et al. [33] represent in the city map incident locations and the main interest points as well. They conclude

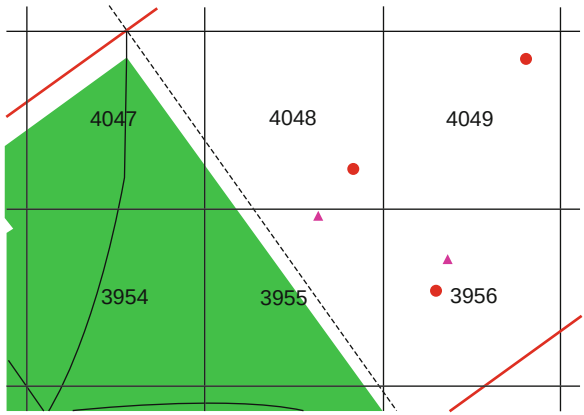


Fig. 2 Spatial representation of incident locations (*red circle* represents incident locations. Bars and restaurants are illustrated by the *ginger pink triangles*)

[1] [30] uses spatial buffers to classify objects based on attribute similarity and distance-based proximity.

that most of the incidents occur close to bars and restaurants. Koperski and Han [34] pioneered the concept of association rules, providing detailed descriptions of their formal properties as well as a top-down tree search technique that exploits background knowledge in the form of a geographic concept hierarchy. Spatial association rules include spatial predicates such as topological distance or directional relations – in the precedent or antecedent [2]. Several new directions have been proposed, including extensions for quantitative rules, extensions for temporal event mining, testing the statistical significance of rules, and deriving minimal rules [7]. One of the most known types is co-location pattern: these are subsets of spatial objects that are frequently located together. Huang and his colleagues [35] develop a multi-resolution filtering algorithm for discovering co-location patterns in spatial data.

Outlier Detection

In [19], the authors define a spatial outlier as a spatially-referenced object whose non-spatial attributes appear inconsistent with other objects within some spatial neighborhood. This does not imply that the object is significantly different than the overall database as a whole: it is possible for a spatial object to appear consistent with the other objects in the entire database but nevertheless appear unusual with a local neighborhood. "Outlier detection" involves identifying data items that are atypical or unusual. In a sense, this definition leaves it up to the analyst (or a consensus process) to decide what will be considered as unusual. Before abnormal observations can be singled out, it is necessary to characterize normal observations. In this context, two activities are essential for characterizing a set of data:

- Examination of the overall shape of the graphed data for important features, including symmetry and departures from assumptions.
- Examination of the data for unusual observations that are far removed from the mass of data. These points are often referred to as outliers. Two graphical techniques for identifying outliers, scatter plots and box plots, along with an analytic procedure for detecting outliers when the distribution is normal (Grubbs' Test), are used. For spatial problems, Ng [36] suggests that the distance-based outlier analysis method could be applied to spatiotemporal trajectories to identify abnormal movement patterns through a geographic space.

Representing geospatial data for use in outlier analysis remains a difficult problem. Outlier detection, as shown in Fig. 3 is applied to detect significant outliers (unusual occurrences) of the number of new cancer cases in the USA. Different colors represent the number of lung cancer mortalities (of white males, 1950–1969) in different states. The map shows a "hot spot" in MT, which is actually a site of copper smelter. Typically, two or more data mining tasks are combined to explore the characteristics of data and identify meaningful patterns. A key challenge is that, as Thuraisingham argues in [37], "Data mining is still more or less an art." Indeed,

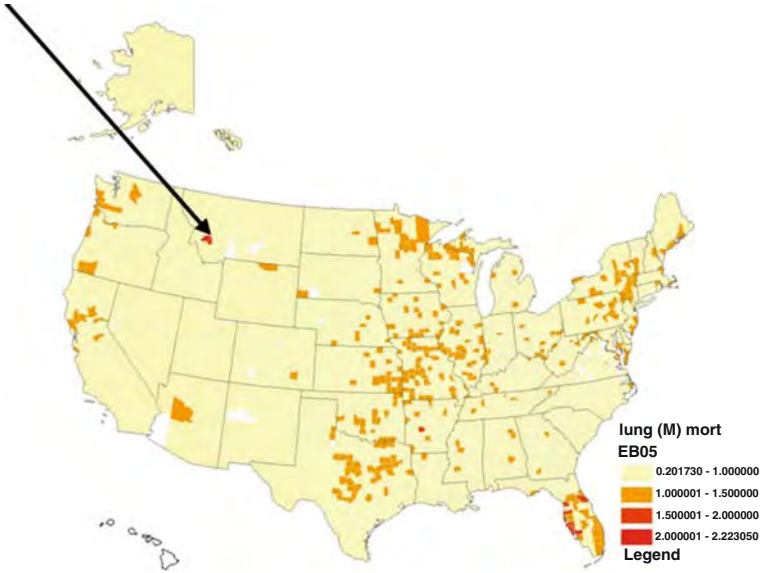


Fig. 3 Lung cancer mortality in the USA 1950–1969 [38]

we can never be sure if a given technique will always be effective in obtaining the expected results. Similarly, we cannot affirm that given certain data characteristics, a certain sequence of tasks would achieve the expected goal. Consequently, high levels of experience and expertise are as important as applying the aforementioned data mining techniques. The expert absolutely needs to try different options and will often make errors before reaching a satisfactory result with the help of data mining techniques. Thus, the development of geospatial-specific data mining tasks and techniques will be increasingly important to help people analyze and interpret the vast amount of geospatial data being captured. In addition, more research is needed to establish firm methodologies to perform data mining on spatial data. More specifically, many research challenges have to be faced. They are presented at the end of this chapter.

4 Applications

Efficient tools for extracting information from spatial data sets can be of importance to organizations which own, generate and manage large geospatial data sets. In this section, we present some examples of the applications of GDM tools and techniques in different domains, namely, Business, traffic networks, and Earth observation. These examples are only chosen for illustrative reasons; they do not constitute an exhaustive list of applications. Other applications include the prediction of forest fire hazardous areas [39].

4.1 Application of GDM in Business

Nowadays, there is a huge amount of information describing both consumer and business demographics related to practically every point on the map. Attached to each map object (or point) can be a very detailed set of data describing it, such as consumer and business demographics, site survey information, and historical customer purchasing patterns. GIS provides operators that allow you to link tables based on proximity as well as join related databases' key fields. With this capability, you can address questions such as: Where do my best customers live? Or what competitive choices do my customers have in a given geographical area? GIS can directly answer these and other similar questions [40]. Demographic reporting is commonly used to generate a detailed characterization of potential store locations, broader marketing regions, or individual customers. However, more information is only better if you can effectively analyze it and assimilate it into actionable decisions. Unfortunately, detailed information provided by GIS often does not directly yield a decisive score that can be acted upon. Efficient tools for extracting information from geospatial data are crucial to organizations which make decisions based on large spatial data sets. Data mining products can be a useful tool in decision-making and planning just as they are currently in the business world. To take GIS analysis beyond demographic reporting to true market intelligence, it is then necessary to incorporate the ability to analyze and condense a large number of variables into a single score [40]. This is the strength of the data mining technology and the reason why there is a new link between GIS and data mining. Depending upon the application domain, GIS can combine historical customer or retail store sales data with demographic, business, traffic, and market data. This data set is then used for building predictive models to score new locations or customers for sales potential, targeted marketing, and other similar applications. Thanks to spatial data mining techniques, problems from different domains similar to those following examples would be easily answered in retail, banking, and telecoms domain, respectively:

- How should inventory, merchandising and pricing be adjusted in different locations in order to make more profits?
- Which of savings account customers are the best candidates for brokerage accounts and what are the most effective incentives to attract them?
- Which of my existing customers are most at risk to move to a competing carrier and which non-customers are the best prospects to switch to my service?

Although many are already applying spatial data mining techniques in their business, the current processes tend to be very disjointed and requiring the manual movement of data between various software applications [40]. This tends to limit the use of the technology to those who have skills in database management, GIS, data analysis and predictive modeling, programming and, most importantly, the specific business application. It is, thus, necessary in the future to integrate these technologies in a manner that will streamline the process. This will allow companies to focus on the business opportunities instead of the technical aspects.

4.2 Traffic Networks

One major challenge of traffic controlling is traffic prediction. In the last few years, researchers proposed methods that are able to estimate the traffic density for a future point of time. A lot of work has been published in the field of traffic data mining. Some work has been published for the detection of traffic jams [41, 42]. Some of them are based on clustering trajectories. The resulting clusters indicate routes with potentially high traffic load. Further approaches are based on the detection of dense areas of moving objects [43]. This approach tries to find moving clusters in moving object databases. The difference with clustering trajectories is that the identity of a moving cluster remains unchanged while its location and content may change over time. Other traffic prediction approaches use historical observations which are based on regression analysis [44] while some derive the current traffic data from a sensor network measuring the traffic at certain locations in the traffic network [45]. More recently, Kriegel et al. [46] have proposed a statistical approach (based on short-time observations of the traffic history) to predict the traffic density on any edge of the network at a future point of time, which offers valuable information for traffic control, congestion prediction and prevention. Another application (other than traffic prediction) is the detection of suitable traveling paths for individuals that want to travel to a certain destination in a possibly most cost-efficient way. An example is given in [47] where the authors use an adaptive navigation method based on historical traffic patterns mined from a large set of given traffic data.

4.3 Application to Earth Observation and Change Detection

Earth science data consists of a sequence of global snapshots of the Earth taken at various points in time. Each snapshot consists of measurement values for a number of variables (e.g., temperature, pressure, and precipitation) collected globally. This remote sensing data (mainly conducted by the satellites of the NASA's Earth Observing System (EOS)), combined with historical climate records and predictions from ecosystem models, offers new opportunities for understanding how the Earth is changing, for determining what factors cause these changes, and for predicting future changes. Data mining techniques provide tools to discover interesting patterns that capture complex interactions among global ocean temperatures, land surface meteorology, and terrestrial carbon flux [48]. Although Earth Scientists have traditionally used statistical tools as their preferred method of data analysis, they are interested in using data mining tools to complement the statistical tools for the following reasons. Zhang et al. [49] give three main reasons to justify the need of data mining techniques. First, the statistical method of manually analyzing a single dataset via making hypothesis and testing them is extremely demanding due to the extremely large families of spatiotemporal hypotheses and patterns. Second, classical statistics deals primarily with numeric data, whereas Earth Science data contains many categorical attributes such as types of vegetation, ecological events,

and geographical landmarks. Third, Earth Science datasets have selection bias in terms of being convenience or opportunity samples rather than traditional idealized statistical random samples from independent and identical distributions [50]. Data mining allows Earth Scientists to spend more time choosing and exploring interesting families of hypotheses derived from the data. More specifically, by applying data mining techniques, some of the steps of hypothesis generation and evaluation will be automated, facilitated, and improved. For example, images taken by satellites can be analyzed in order to identify homogeneous groups of pixels which represent various features or land cover classes of interest. Elements of visual interpretation are then used to classify the features included in images. In digital images it is possible to model this process, to some extent, by using two methods: Unsupervised Classifications and Supervised Classifications. The unsupervised classification is a computerized method without direction from the analyst in which pixels with similar digital numbers are grouped together into spectral classes using statistical procedures such as nearest neighbor and cluster analysis. The resulting image may then be interpreted by comparing the clusters produced with maps, air-photos, and other materials related to the image site. In a supervised classification, the analyst identifies several areas in an image which represent known features or land cover. These known areas are referred to as "training sites" where groups of pixels are a good representation of the land cover or surface phenomenon. Using the pixel information the computer program (algorithm) then looks for other areas which have a similar grouping and pixel value. The analyst decides on the training sites, and thus supervises the classification process.

5 Research Challenges

In this section, we briefly present some research challenges that have to be faced in order to better support geospatial–temporal applications in general and data mining in particular. A first set of research topics is related to geospatial databases. In addition, research investments are needed in geometric algorithms to manipulate the huge amount of available geospatial data. Finally, considerable work remains to be done to improve the discovery of structure (in the form of rules, patterns, regularities, or models) in geospatial databases.

5.1 Geospatial Databases

Relational DBMSs are not appropriate for storing and manipulating geospatial data because of the complex structure of geometric information [51]. Besides, geospatial data often span a region in continuous space and time, but computers can only store and manipulate finite, discrete approximations, which can cause inconsistencies and erroneous conclusions. A particularly difficult problem for geospatial data

is representing both spatial and temporal features of objects that move and evolve continuously over time. Examples include hurricanes, pollution clouds, and pods of migrating whales. Several commercial systems are now available with spatiotemporal extensions. Nevertheless, these extensions are not comprehensive (i.e., they still require application-specific extensions) and do not accurately model space and time for continuously moving objects [52]. Most of the research in moving-object databases has concentrated on modeling the locations of moving objects as points (instead of regions). Although few researchers have proposed new data models such as in [53], where the authors present an abstract model for implementing a spatiotemporal DBMS extension based on a comprehensive model of geospatial data types, more research is needed to achieve the requirements of moving objects. Besides, researchers have to address problems of reasoning and modeling on very short or highly variant temporal cycles. This would be very useful to support real-time tornado tracking or avalanche prediction for instance. To achieve this goal, more work is needed to develop real-time data mining, use knowledge discovery tools to guide correlation of discovered data patterns across time, and validate data trends across temporal discontinuities.

Query languages also will need to be extended to provide high-level access to the new geospatial data types. For example, it should be possible to refer to future events and specify appropriate triggers, such as "Sound an alarm when the fire is within 10 km of any habitation." The development of ontologies for geospatial phenomena is another critical challenge. In the geospatial domain, ontologies would define (non-exhaustive list) geographic objects, spatial relations, processes, and situational aspects. Research on ontologies for geographic phenomena began only recently [54]. More research is needed to design geospatial ontologies for each specific application and be able to select the appropriate ontology for a given context. According to [52], these ontologies should also be extendable (can evolve over time). Another important challenge while dealing with geospatial databases is the integration of geospatial data. The purpose of data integration is to combine data from heterogeneous, multidisciplinary sources into one coherent data set. A well known problem is conflation. Conflation refers to the integration of geospatial data sets that come from multiple sources, are at different scales, and have different positional accuracies. Making the conflation process completely automatic is one of the most challenging research issues here. More research is also required to create new data models and languages specifically designed to support heterogeneous spatiotemporal data sets. Besides, new languages and mechanisms must be developed to explicitly express spatial attributes. In this context, languages like XML as well as resource discovery technologies may play an important role. Finally, and due to the nature of the spatial data (missing or incomplete data), a preprocessing phase is often required. However, research on the preprocessing of spatial data has lagged behind. Hence, there is a need for preprocessing techniques for spatial data to deal with problems such as treatment of missing location information and imprecise location specifications, cleaning of spatial data, feature selection, and data transformation [21].

5.2 Issues in Geospatial Algorithms

Following the technologic expansion (especially in remote sensing), captured geospatial data are available under different resolutions, levels of accuracy, and formats. This diversity causes what is called imperfections (noisy and/or uncertain data). For instance, while grid terrain representations (the format usually provided by remote-sensing devices) are easily supported by simple algorithms, other representations which are superiors to grids such as triangular irregular networks (TINs) (used as massive terrain datasets for certain kinds of applications) introduce inconsistencies when converted to the grid format. Besides, classical algorithms which assume perfect data may thus not function correctly or efficiently in practice since real-world data usually present many imperfections. New algorithms are then required to deal with these constraints. Another limitation of existing algorithms is related to computational memory. Indeed, most of these algorithms assume that each memory access costs one unit of time regardless of where the access takes place. This assumption is nowadays unrealistic since modern machines contain a hierarchy of memory ranging from small, fast cache to large, slow disks. For this reason, and according to [52], it is becoming increasingly important to design memory-aware algorithms, in which data accessed close in time also are stored close in memory. Mining spatial patterns is often computationally expensive. For example, co-location mining algorithm is more expensive than the apriori algorithm for classical association rule mining [55]. Research is needed to reduce the computational costs of spatial data mining algorithms by a variety of approaches including the classical data mining algorithms as potential filters or components.

To conclude this second set of challenges, it is worth to say few words about the classical/specialized algorithms dilemma. As previously discussed, both classical and "specialized" algorithms can be used to mine spatial data. However, existing literature does not provide guidance regarding the choice between the two classes of techniques. Therefore, new research is needed to compare the two sets of approaches in terms of effectiveness and computational efficiency [21].

5.3 Geospatial Data Mining

One of the most difficult data mining problems is to determine which task to perform (i.e., which class of patterns to look for) in a given data set. Besides, expert users have to specify new types of patterns since the assumptions required for the classical stochastic representations do not always hold. As a consequence, data mining tools should not only support frequent patterns but also be extensible to search for new patterns. In a typical data mining process, an analyst tries several data mining tools (e.g., clustering, classification, association rules, and outlier analysis) and data transformations (e.g., log transformations, dimensionality reductions, aggregation for a coarser granularity). This process is repeated until the analyst discovers some new regularities or, conversely, detects anomalous conditions. However, the

main problem is that analysts may not be trained to use all the aforementioned data mining tools. The challenge is automate, as much as possible, the process by using efficient algorithms. According to [52], software agents could assist the analyst by indicating for which data sets a new algorithm is applicable, applying the algorithm, and comparing the new results with the previous ones.

Moving and evolving objects are not only difficult to model (as previously mentioned), but also to be mined. A key problem is how to identify and categorize patterns in the trajectories of moving objects, such as migrating birds or fish. Another difficult problem is to identify and categorize patterns in objects that evolve, change, or appear and disappear over time, such as animal habitats. Thus, more data mining algorithms are required to handle temporal dimensions and to support complex spatial objects (other than points). According to [52], it is necessary to rethink current data mining algorithms (clustering, classification, and association rules) in terms of trajectories. Another direction is to create specialized algorithms for trajectories with no constraints in two- or three-dimensional space (plus time) as opposed to constrained trajectories such as movement along a network, or even more constrained, movement on a circuit.

The last challenge in this third set is related to visualization. Visualization in spatial data mining is useful to identify interesting spatial patterns. During the data mining process and to facilitate the visualization of spatial relationships, research is needed to represent both spatial and non-spatial features. For example, many visual representations have been proposed for spatial outliers. However, we do not yet have a way to highlight spatial outliers within visualizations of spatial relationships. For instance, in most existing visualizations, the spatial relationship between a single spatial outlier and its neighbors is not obvious [21]. It is thus necessary to transfer the information back to the original map in geographic space to check neighbor relationships.

6 Conclusion

To conclude, the development of data mining and knowledge discovery tools must be supported by a solid geographic foundation that accommodates the unique characteristics (detailed in Sect. 2) and challenges presented by geospatial data. Classical data mining and knowledge discovery methods have not been implemented to deal effectively with geospatial data, whose sensitivities are known widely to geographers. Specialized techniques have been used to deal with this problem (see Sect. 3). Even though many challenges are still to be addressed by researchers (see the previous section), geospatial data mining is at the heart of many applications. In fact, and as illustrated in the fourth section of this chapter, extracting information from geospatial data are crucial to different types of organizations which make decisions based on large spatial data sets. With the expansion of the modern remote sensing devices, geo-spatial data mining will certainly be crucial to capture relevant information.

References

1. M. Gahegan, *On the Application Of Inductive Machine Learning Tools To Geographical Analysis, Geographical Analysis*, vol. 32, 2000, pp. 113–139
2. H.J. Miller, J. Han, *Geographic Data Mining and Knowledge Discovery* (Francis and Taylor, London, 2001)
3. W.R. Tobler, *Cellular Geography Philosophy in Geography*, ed. by Gale, Olsson (Dordrecht Reidel, 1979)
4. S. Chawla, S. Shekhar, W. Wu, U. Ozesmi, Extending Data Mining for Spatial Applications: A Case Study in Predicting Nest Locations. in *Proceedings of the International Conference on ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2000
5. M. Ester, H.-P. Kriegel, J. Sander, Knowledge Discovery in Spatial Databases, KI '99: in *Proceedings of the 23rd Annual German Conference on Artificial Intelligence* (Springer, London, 1999), pp. 61–74, ISBN 3-540-66495-5
6. J.F. Roddick, M. Spiliopoulou, A Bibliography Of Temporal, Spatial And Spatio-temporal Data Mining Research, SIGKDD Explor. News vol. 1(1), 1999. ISSN 1931-0145 (ACM, New York, USA), pp. 34–38
7. J. Han, M. Kamber, *Data Mining: Concepts and Techniques*, 2nd ed. (Morgan Kaufmann, CA, 2006)
8. B. Buttenfield, M. Gahegan, H. Miller, M. Yuan, Geospatial Data Mining and Knowledge Discovery, A UCGIS White Paper on Emergent Research Themes Submitted to UCGIS Research (2000)
9. R. Healey, in *Geographic Information Systems: Principles and Applications*, ed. by D. Maguire, M.F. Goodchild, D. Rhind. Database Management Systems (Longman, London, 1991)
10. A.S. Camara, J. Raper *Spatial Multimedia and Virtual Reality* (Taylor and Francis, London, 1999)
11. O.R. Zaane, J. Han, Z.N. Li, J. Hou, Mining Multimedia Data, in *Proceedings of CASCON'98: Meeting of Minds*, 1998, Toronto, Canada
12. A.D. Cliff, P. Haggett, in *Geocomputation: A Primer*. On Complex Geographical Space: Computing Frameworks For Spatial Diffusion Processes (Wiley, New York, 1998)
13. H.J. Miller, J. Geogr. Syst. **2**(1), 55–60 (2000)
14. A.S. Fotheringham, M. Charlton, C. Brunsdon, *Two Techniques For Exploring Non-stationarity In Geographical Data, Geographical Systems*, vol. 4, 1997, pp. 59–82
15. A. Getis, J.K. Ord, *Local Spatial Statistics: An Overview, Spatial Analysis: Modeling in a GIS Environment* (GeoInformation International, Cambridge, UK, 1996), pp. 261–277
16. M.J. Egenhofer, J.R.Herring, Categorizing binary topological relations between regions, lines and points in geographic databases, the 9-intersection: Formalism and its Use for Natural-language Spatial Predicates, Santa Barbara CA National Center for Geographic Information and Analysis Technical Report 94-1, pp. 1–28, 1994
17. A. Okabe, H.J. Miller, Cartography Geogr. Inf. Syst. **23**, 180–195 (1996)
18. J. Roddick, B. Lees, in *Paradigms for Spatial And Spatio-temporal Data Mining* ed. by H. Miller, J. Han. Geographic Data Mining and Knowledge Discovery (Taylor and Francis, London, 2001)
19. S. Shekhar, C.-T. Lu, P. Zhang, A Unified Approach to Detecting Spatial Outliers, Geoinformatica **7**(2), 2003. ISSN 1384-6175 (Kluwer Academic Publishers, Hingham, MA), pp. 139–166
20. M.Y. Santos, L.A. Amaral, Geospatial Data Mining In The Analysis Of A Demographic Database, Soft Comput. **9**(5), 2005. ISSN 1432-7643 (Springer, Berlin, 2005), pp. 374–384
21. S. Shekhar, P. Zhang, Y. Huang, R. Vatsavai, in *Trend in Spatil Data Mining*, ed. by H. PKargupta, A. Joshi, K. Sivakumar, Y. Yesha. Data Mining: Next Generation Challenges and Future Directions (AAAI/MIT, 2004), pp. 357–381
22. A.K. Jain, R.C. Dubes, *Algorithms for Clustering Data* (Prentice-Hall, Upper Saddle River, NJ, 1988), ISBN 0-13-022278-X

23. L. Kaufman, P.J. Rousseeuw, *Finding Groups In Data: An Introduction To Cluster Analysis* (Wiley, New York, 1990)
24. R.T. Ng, J. Han, Efficient and Effective Clustering Methods for Spatial Data Mining, University of British Columbia, 1994, Technical report, Number TR-94-13
25. M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, in *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pp. 226–231, 1996
26. Sudipto Guha, Rajeev Rastogi and Kyuseok Shim, CURE: An Efficient Clustering Algorithm for Large Databases, SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA, Laura M. Haas and Ashutosh Tiwary (Eds.), ACM Press, 1998, ISBN 0-89791-995-5, pp. 73-84
27. Sudipto Guha, Rajeev Rastogi and Kyuseok Shim, ROCK: a robust clustering algorithm for categorical attributes, Inf. Syst., Volume 25, Number 5, 2000, ISBN 0306-4379, pp. 345–366, Elsevier Science Ltd., Oxford, UK, UK
28. Han J., Kamber, M. and Tung A. K. H., Spatial Clustering Methods in Data Mining: A Survey, in Geographic Data Mining and Knowledge Discovery, Taylor and Francis, 2001
29. M. Ester, H.-P. Kriegel, J. Sander, *Spatial Data Mining: A Database Approach*, *Advances In Spatial Databases* (Springer, Heidelberg, 1997), Lecture Notes in Computer Science, vol. 1262, pp. 47–66
30. K. Koperski, J. Han, N. Stefanovic, An Efficient Two-Step Method for Classification of Spatial Data, in *Proceedings of the Symposium on Spatial Data Handling (SDH 98)*, 1998, pp. 45–54
31. F. Qi, A.-X. Zhu, Int. J. Geogr. Inf. Sci. **17**(8), 771–795 (2003)
32. S. Gopal, W. Liu, X. Woodcock, in *Visualization Based On Fuzzy ARTMAP Neural Network For Mining Remotely Sensed Data*, ed. by H.J. Miller, J. Han. Geographic Data Mining and Knowledge Discovery (Taylor and Francis, London, 1996), pp. 315–336
33. V. Karasova, J.M. Krisp, K. Virrantaus, Application of Spatial Association Rules for Improvement of a Risk Model for Fire and Rescue Services, in *Proceedings of the 10th Scandinavian Research Conference on Geographical Information Science (ScanGIS2005)*, 2005
34. K. Koperski, J. Han, Discovery of Spatial Association Rules in Geographic Information Databases, SSD '95: in *Proceedings of the 4th International Symposium on Advances in Spatial Databases* (Springer, London, 1995), pp. 47–66, ISBN 3-540-60159-7
35. Y. Huang, S. Shekhar, H. Xiong, Discovering Colocation Patterns from Spatial Data Sets: A General Approach, IEEE Trans. on Knowl. and Data Eng. **16**(12), 2004. ISSN 1041-4347, pp. 1472–1485, IEEE Educational Activities Department, Piscataway, NJ
36. R.T. Ng, in *Detecting Outliers from Large Datasets*, ed. by Miller, Han. Geographic Data Mining and Knowledge Discovery (Francis and Taylor, London, 2001)
37. B. Thuraisingham, *Data Mining: Technologies, Techniques, Tools, and Trends* (CRC Press, FL, 1998), ISBN 0849318157
38. A.M. Lee, J.F. Fraumeni Jr., Arsenic and Respiratory Cancer In Man: An Occupational Study, vol. 43, JNCI, 1969, pp. 1045–1052
39. J.G. Han, K.H. Ryu, K.H. Chi, Y.K. Yeon, *Statistics Based Predictive Geo-spatial Data Mining: Forest Fire Hazardous Area Mapping Application*, APWeb, 2003, pp. 370–381
40. P. Duke, Geospatial Data Mining for Market Intelligence, Published online at b-eye Network on July 10, 2008
41. S. Gaffney, P. Smyth, Trajectory Clustering With Mixtures Of Regression Models, KDD '99: in *Proceedings of the 5th ACM SIGKDD International Conference On Knowledge Discovery And Data Mining* (ACM, New York, 1999), pp. 63–72, ISBN 1-58113-143-7, San Diego, CA, USA
42. J.-G. Lee, J. Han, K.-Y. Whang, Trajectory Clustering: A Partition-and-group Framework, SIGMOD '07: in *Proceedings of the 2007 ACM SIGMOD International Conference On Management Of Data*, 2007, Beijing, China (ACM, New York, USA), pp. 593–604, ISBN 978-1-59593-686-8
43. P. Kalnis, N. Mamoulis, S. Bakiras, in *On Discovering Moving Clusters In Spatio-Temporal Data*. Advances in Spatial And Temporal Databases, vol. 3633 (Springer, Heidelberg, 2005), Lecture notes in computer science, pp. 364–381

44. R.K. Oswald, W.T. Scherer, B.L. Smith, Traffic Flow Forecasting Using Approximate Nearest Neighbor Nonparametric Regression, 2000, Final project of ITS Center project: Traffic forecasting: non-parametric regressions, December
45. S. Shekhar, C.T. Lu, S. Chawla, P. Zhang, Data Mining and Visualization of Twin-Cities Traffic Data, 2001
46. H.-P. Kriegel, M. Renz, M. Schubert, A. Zuefle, Statistical Density Prediction in Traffic Networks, SIAM SDM, pp. 692–703, SIAM, 2008
47. H. Gonzalez, J. Han, X. Li, M. Myslinska, J.P. Sondag, Adaptive Fastest Path Computation On A Road Network: A Traffic Mining Approach, VLDB '07: in *Proceedings of the 33rd International Conference On Very Large Data Bases*, Vienna, Austria, 2007, pp. 794–805, ISBN 978-1-59593-649-3, VLDB Endowment
48. L. Deren, X. Song, S. Haigang, Z. Xiaodong, Change Detection Based On Spatial Data Mining, ISPRS Workshop on Updating Geo-spatial Databases with Imagery and The 5th ISPRS Workshop on DMGISs, Urumchi, Xingjiang, China, August 28–29, 2007
49. P. Zhang, M. Steinbach, V. Kumar, S. Shekhar, in *Discovery of Patterns in the Earth Science Data using Data Mining*, ed. by J. Zurada, M. Kantardzic. New Generation of Data Mining Applications (Wiley-interscience Publication, 2004)
50. V. Kumar, in *Data Mining for Scientific and Engineering Applications*, ed. by R.L. Grossman (Kluwer Academic Publishers, Norwell, MA, 2001), ISBN 1402000332
51. S. Grumbach, P. Rigaux, L. Segoufin, The DEDALE system for complex spatial queries, SIGMOD '98: in *Proceedings of the 1998 ACM SIGMOD International Conference On Management Of Data*, Seattle, Washington, ACM, New York, 1998, pp. 213–224, ISBN 0-89791-995-5
52. National Research Council NRC US, Geospatial Databases and Data Mining, in IT Roadmap to a Geospatial Future, By a Committee on Intersections Between Geospatial Information and Information Technology. National Academies Press, 2003
53. R. Hartmut Güting, M.H. Böhlen, M. Erwig, C.S. Jensen, N.A. Lorentzos, M. Schneider, M. Vazirgiannis, A Foundation For Representing And Querying Moving Objects, ACM Trans. Database Syst. **25**(1). ISSN 0362-5915 (ACM, New York, 2000), pp. 1–42
54. D. Mark, M. Egenhofer, S. Hirtle, B. Smith, UCGIS Emerging Research Theme: Ontological Foundations For Geographic Information Science, 2000
55. R. Agrawal, R. Srikant, Fast algorithms for mining association rules. in *Proceedings of the 20th International Conference on VLDB*, Chile, 1994, pp. 487–499

Data Mining and Discovery of Chemical Knowledge

Lu Wencong

In this chapter, the Data mining methods adopted are briefly introduced. The main focuses are on the successful applications of data mining methods in chemistry and chemical engineering. The discoveries of chemical knowledge cover the formation of ternary Intermetallic compounds, structure–activity relationships of drugs, and industrial optimization based on chemical data mining methods, especially by using statistical pattern recognition and support vector machine.

1 Data-Mining-Based Prediction on Formation of Ternary Intermetallic Compounds

1.1 Data-Mining-Based Prediction on Formation of Ternary Intermetallic Compounds Between Nontransition Elements

The prediction of the formation of ternary intermetallic compounds in ternary alloy systems is a very important and difficult task for the calculation of ternary alloy phase diagrams. In this work, a systematic method based on data mining was proposed to study the regularities of ternary intermetallic compound formation in an empirical way, by using pattern recognition techniques developed in our lab and Villars's system of atomic parameters [1]. Here, data-mining-based prediction on formation of ternary intermetallic compounds will be described and discussed based on four categories: (1) system (ternary intermetallic compounds) between nontransition elements; (2) system between transition elements; (3) system between one nontransition element and two transition elements; (4) system between two nontransition elements and one transition element.

L. Wencong (✉)
Shanghai University, 99 Shangda Road, BaoShan District, Shanghai, Peoples Republic of China
e-mail: wclu@shu.edu.cn

M.M. Gaber (ed.), *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*, DOI 10.1007/978-3-642-02788-8_11,

1.2 Methods of Investigation

Atomic Parameters Used for Data Mining

It is generally accepted that the physico-chemical behavior of alloy systems is related to three factors: geometrical factor, charge transfer factor, and energy band factor. The geometrical factor can be roughly described by some function of effective atomic radii, and the charge transfer factor should be described by the difference of electronegativities of the constituent elements, while the energy band factor is usually to be considered as some function of the number of valence electrons. In Villars's system of atomic parameters, the effective atomic radius is described as some function of R_{sp} (pseudo-potential radius sum), the charge transfer is described by X_{MB} (the Martynov–Batsaanov scale of electronegativity), and the valence electron number is described by VE (the number of elements given by Villars). In this work, the parameters given by Villars are used. But the VE value of Zn group elements (Zn, Cd, and Hg) used in this work is 2 (instead of 12) since the energy levels of the d-orbitals are so deep. It is unlikely that spd hybridization plays a significant role in intermetallic compound formation for Zn group elements.

Methods of Computation

In this work, atomic parameters R_{sp}, VE, and X_{MB} (abbreviated R, V, and X, respectively) and their functions (the ratio of R_{sp} and the difference of X_{MB} are usually used to describe the geometrical factor and charge transfer factor, respectively. Here, we denote $R_{mn} = R_m/R_n$, and $X_{mn} = X_m - X_n$, respectively) are used to span the multi-dimensional spaces. The representative points of ternary systems are "plotted" into these spaces. If the representative points of ternary compound forming systems are located in a definite zone in these spaces, various pattern recognition methods can be used to find the mathematical models to describe the zone of ternary compound formation. Linear projection methods making projection maps having good separation between the representative points of ternary compound forming systems and that of the systems without ternary compound formation are used to find the regularities of classification. Each projection map forms a rectangular frame or some lines describing the boundaries of two kinds of points (ternary compound forming and that without ternary compound). Based on a series of projection maps, a hyperpolyhedron will be defined in multi-dimensional space, as shown in Fig. 1, so the envelop of the zone of ternary compound formation can be roughly represented by this hyperpolyhedron. A series of inequalities describing the hyperplanes as the boundaries of the hyperpolyhedron can be used as the criterion of ternary intermetallic compound formation. The "Materials Research Advisor" software is used for the computation work [2]. By using this software, the inequalities describing the hyperpolyhedron can be obtained automatically.

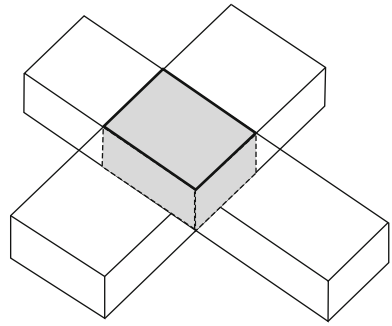


Fig. 1 Conceptual representation of the hyperpolyhedron model formed by intersection of the rectangular tunnel represented by rectangular frames or hyperplanes

Data Files for Training and Model Building

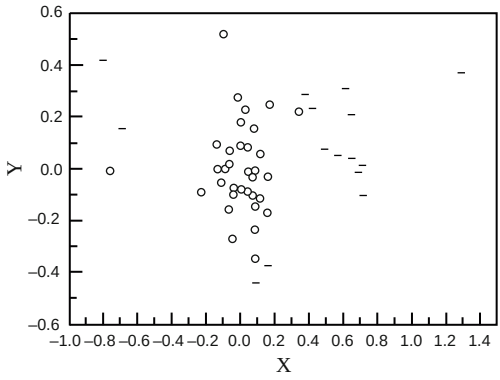
Villars's Data Bank for Ternary Alloy Systems has collected reliable known phase diagrams of ternary alloy systems. Using this data bank, the data sets available can be organized as four categories of ternary alloy systems mentioned in Sect. 1.1. All samples are defined as two classes: class 1 (symbolized as "O" in diagrams of this work) for samples of ternary compound formation and class 2 (symbolized as "-" in diagrams of this work) for no ternary compound (for the sake of self-consistency, the order of the three elements is defined as follows: the first element has the lowest valency and the third element has the highest valency. If two elements have the same valency, they should be arranged in ascending electronegativity values).

1.3 Results and Discussion

Intermetallic Compounds Between Nontransition Elements

If we confine our object of investigation to systems containing one common element, the atomic parameters (or their functions) of the other two elements can be used to span the multi-dimensional space to investigate the regularities of ternary compound formation. For example, 47 Mg-containing ternary systems can be used as the training set. By using R_{sp}, VE, and X_{MB} of the other two components as the features of data mining work, the linear projection map shown in Fig. 2 can be obtained. It can be seen that the separation of two kinds of representative points is rather clear-cut. The coordinates of this map are linear combinations of the atomic parameters, as shown in Fig. 2.

Based on the data mining work, it has been found that the problems can be solved easily if we treat the ternary systems containing metalloids and without metalloids separately. Figure 3 illustrates bivariate maps for the 117 systems not containing metalloids. Figure 3a is a plot with two radius ratio values $R_{31}0$ and R_{21}. From



$$X = 0.712R_2 + 0.732R_3 - 0.666V_2 - 0.443V_2 - 0.969V_3 + 2.037X_1 + 1.907X_2 + 4.260X_3 - 9.983$$

$$Y = 0.045R_1 + 0.995R_2 - 0.540V_1 + 3.263V_2 + 0.486V_3 + 2.043X_1 - 10.085X_2 - 2.030X_3 + 9.307.$$

Fig. 2 Formation of ternary compounds in Mg-containing systems. (*open circle*) Ternary compound formation; (*line*) no ternary compound

Fig. 3a it can be seen that most of the systems without ternary compound formation distribute near the point corresponding to nearly equal size of the three kinds of atoms. Figure 3b shows that the electronegativity difference of the first and third elements exhibits a strong influence on ternary compound formation. For these 117 training points all systems with $X_{31} \geq 0.65$ form ternary intermetallic compounds. If we cut away the subspace with $X_{31} \geq 0.65$, a subset consisting of 53 samples with $X_{31} < 0.65$ is formed. Using these 53 sample points as the training set, a linear projection map with good separability can be obtained, as shown in Fig. 4. By analyzing projection maps by this way, the criteria for ternary compound formation for systems containing no metalloid can be obtained.

Based on data mining work, it can be found that the electronegativity of the first element X_1 is an influential parameter for ternary compound formation for metalloid-containing alloy systems of nontransition metals. Systems with $X_1 < 1.26$ usually form ternary intermetallic compounds, while most of the systems with $X_1 > 1.26$ do not form ternary intermetallic compounds. Figure 5 illustrates the regularity of ternary compound formation in the subspace $X_1 > 1.26$. The criterion for ternary compound formation in this subspace can be expressed as follows:

$$24.48 \leq 9.156R_1 - 2.733R_2 - 1.140R_3 + 9.759X_1 + 1.631X_2 - 2.117X_3 \leq 25.33,$$

$$29.781 \leq 0.038R_1 + 0.380R_2 - 4.521R_1 - 1.319X_1 + 1.697X_2 + 17.11X_3 \leq 30.89.$$

Based on the earlier results, we can try to make a computerized prediction of ternary intermetallic compounds using the atomic parameter-pattern recognition method. For example, five newly discovered compounds, Sr-Hg–Pb, Sr-Hg–n, Li-Ca–Al, Li-Zn–Ga, and Na-Cd–Ga, are “predicted” to be ternary compound forming systems by the criteria found earlier. The predicted results are all in agreement with the experimental results.

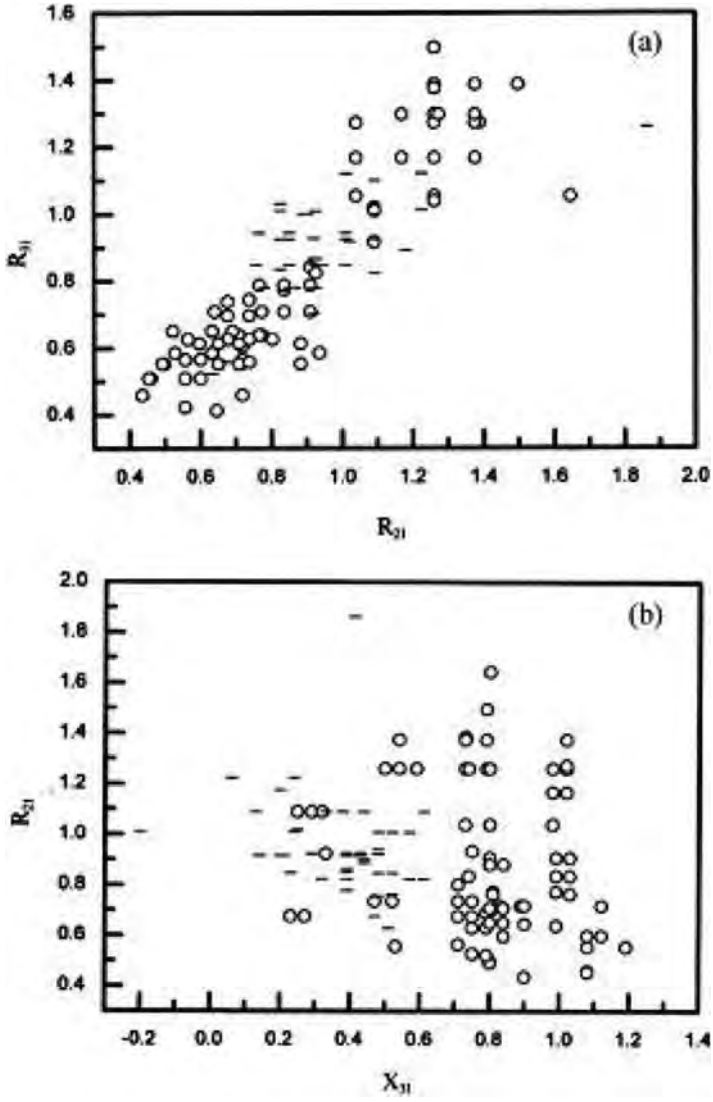
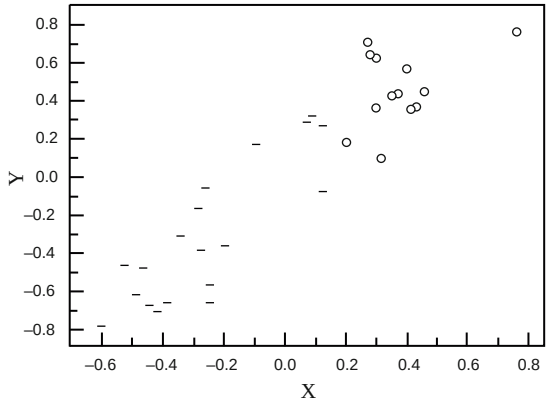


Fig. 3 Bivariate maps showing the influence of Rsp and X_{MB} on ternary compound formation. (*open circle*) ternary compound formation; (*line*) no ternary compound

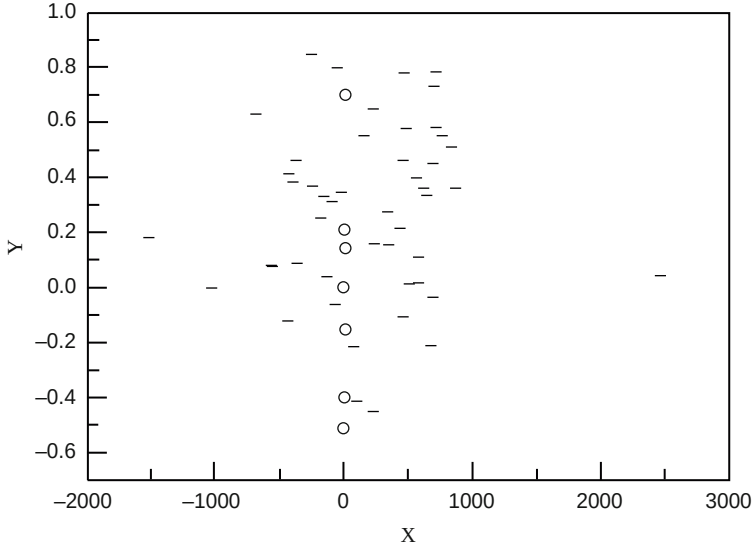
It is meaningful to discuss the physical background of the influences of these factors. From the regularities found, a higher electronegativity difference X_{31} favors ternary compound formation. Since the order of constituent elements is arranged with ascending valence and electronegativity, large X_{31} means that the sum of X_{21} and X_{32} is also large. A large electronegativity difference should induce strong interatomic charge transfer and partial ionicity, and should stabilize the lattice of ternary intermetallic compounds.



$$X = 0.296R_{31} - 0.264R_{21} + 0.990V_1 - 1.051V_2 + 0.284V_3 - 2.374X_{31} + 4.921X_{21} - 5.067;$$

$$Y = -0.334R_{31} + 1.078R_{21} + 1.292V_1 - 0.969V_3 + 2.638X_{31} + 0.128X_{21} - 1.335.$$

Fig. 4 Regularity of ternary compound formation of alloy systems with X31 < 0.65. (*open circle*) ternary compound formation; (*line*) no ternary compound



$$X = -2187R_1 + 276.5R_2 + 968.9R_3 - 523.8V_1 + 718.1V_2 + 29.5V_3 - 551.6X_1 - 3264.5X_2 - 2233.9X_3 + 12331.5;$$

$$Y = 0.854R_1 - 1.118R_2 + 0.665R_3 - 0.820V_1 + 1.435V_2 + 3.818V_3 + 5.243X_1 - 6.747X_2 - 34.22X_3 + 54.59.$$

Fig. 5 Regularity of ternary compound formation of metalloid-containing alloy systems. (*open circle*) ternary compound formation; (*line*) no ternary compound

It has been found that ternary alloy systems having nearly the same atomic radii (in other words, the systems with both R_{31} and R_{21} within the range 1.0 ± 0.15) usually do not form ternary intermetallic compounds. This can be understood as follows: closepacking often increases the stability of intermetallic compounds; systems with atoms of different sizes may have more possibility of stacking these atoms to form denser crystal lattices, and this favors ternary compound formation.

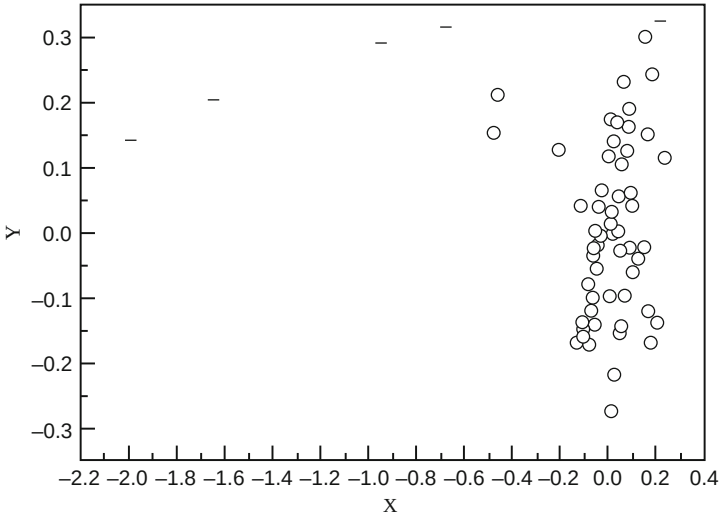
Intermetallic Compounds Between Transition Elements

In Sect. 830, the regularities of formation of ternary intermetallic compounds between nontransition elements have been found. In this section, the regularities of the formation of ternary intermetallic compounds between three transition elements are studied by similar strategies [3]

Figure 6 illustrates the regularity of formation of Fe-containing ternary compounds. The criteria of formation for Fe-containing systems can be expressed as follows:

$$0.077R_1 + 0.0625R_2 + 0.234R_3 + 0.110V_1 - 0.029V_2 - 0.336V_3 + 0.572X_1 + 0.314X_2 + 1.086X_3 \geq 3.281$$

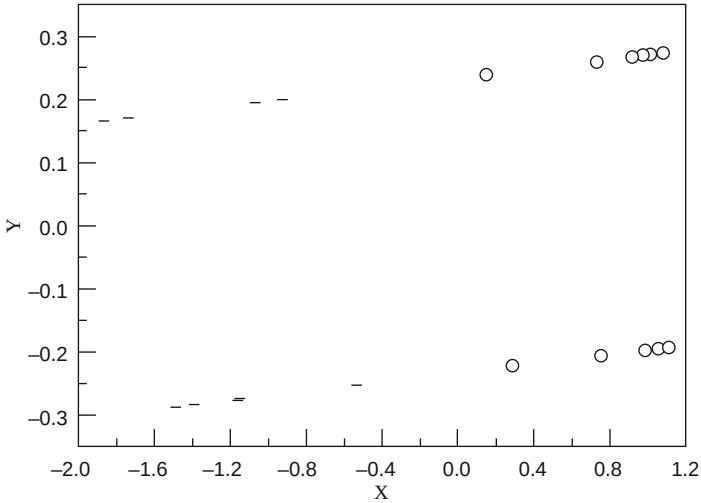
$$-0.696R_1 + 0.144R_2 - 0.112R_3 + 0.165V_1 - 0.048V_2 + 0.080V_3 + 0.079X_1 + 0.673X_2 - 0.093X_3 \leq 1.418$$



$X = 0.077R_1 + 0.062R_2 + 0.234R_3 + 0.110V_1 - 0.029V_2 - 0.336V_3 + 0.572X_1 - 0.313X_2 - 1.086X_3 - 3.759$

$Y = -0.069R_1 - 0.144R_2 - 0.112R_3 + 0.1647V_1 - 0.048V_2 + 0.080V_3 - 0.079X_1 + 0.693X_2 - 0.093X_3 - 1.119$

Fig. 6 Formation of Fe-containing ternary compounds. (*open circle*) ternary compound formation; (*line*) no ternary compound



$$X = -0.049R_1 - 0.285R_2 + 0.123R_3 + 0.192V_2 - 0.315V_3 - 26.849X_1 - 0.106X_2 - 0.505X_3 + 40.599$$

$$Y = 3.414R_1 - 0.011V_3 + 159.33X_1 - 0.018X_3 - 178.71$$

Fig. 7 Formation of Ag-containing ternary compounds. (*open circle*) ternary compound formation; (*line*) no ternary compound

Figure 7 illustrates the regularity of formation of Ag-containing ternary compounds. The criteria of formation for Ag-containing systems can be expressed as follows:

$$-0.049R_1 - 0.285R_2 + 0.123R_3 + 0.192V_2 + 0.315V_3 - 36.849X_1 - 0.106X_2 - 0.505X_3 \geq -40.453$$

Based on the data file including 215 system with V1 equal to 8-11, i.e. all the systems containing one, two, or three late transition elements, a bivariate map of rather good separability is shown in Fig. 8a. It can be seen that all the systems with $R_{31} > 1.30$ or $R_{21} > 1.31$ form ternary intermetallic compounds. But the 67 sample points in the subspace with $R_{31} < 1.30$ and $R_{21} < 1.31$ are still not classified clearly. In Fig. 13.18b, these 67 samples can be separated into two subspaces: the samples in the region with $VE(3) > 8$ are all systems with no ternary compound formation (on the borderline case of $VE(3) = 8$ there are two borderline cases, Fe-Co–Ni system and Fe–Co–Pt system, having disordered solid solution at high temperature, but forming CuAu or $AuCu_3$ type lattices by the decomposition of solid solution at low temperature).

The left-hand side region of Fig. 8b distributes two kinds of sample points. Since the systems containing Mn-group elements will be treated separately (Mn-group elements are the borderline cases between early and late transition elements) in this section. Here, we only take the sample points with $VE(3) \leq 6$ to build a subset for model building.

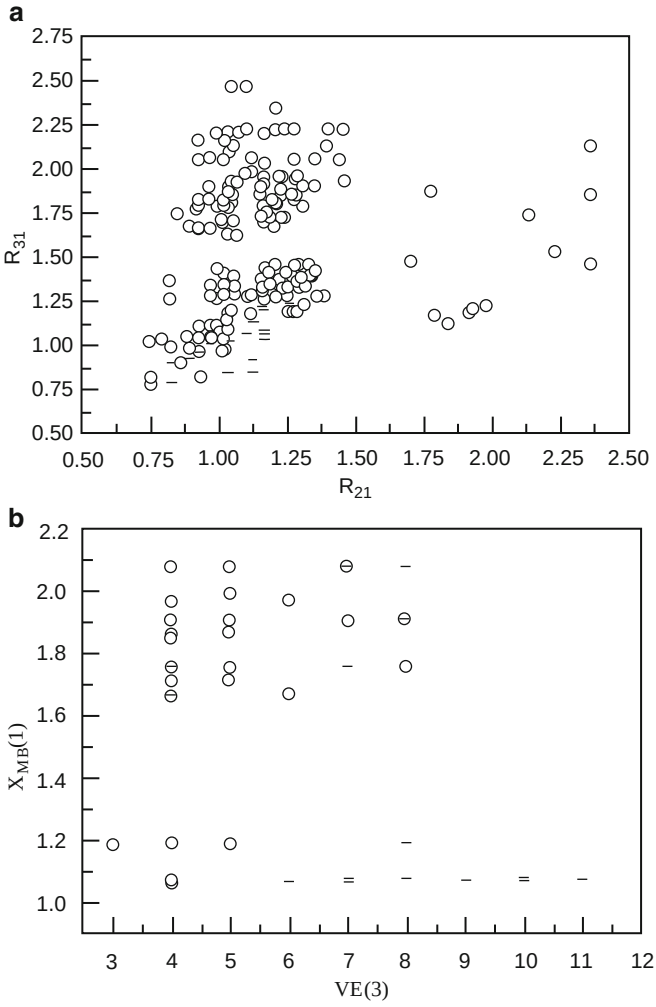
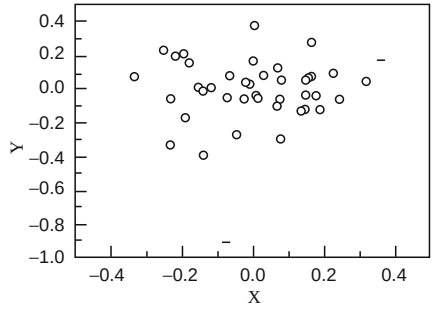


Fig. 8 Ternary compound formation of late transition element-containing alloy systems. (*open circle*) ternary compound formation; (*line*) no ternary compound

Figure 9 illustrates a linear projection map from a 8D space spanned by the VE, X_{MB} of three elements and two radius ratios $R_{\mathrm{sp}}(3)/R_{\mathrm{sp}}(1)$ and $R_{\mathrm{sp}}(2)/R_{\mathrm{sp}}(1)$. The criterion of ternary compound formation for these systems can be expressed as follows:

$$11 \geq V_1 \geq 8$$

$$6 \geq V_3 \geq 3$$



$$X = 0.166V_1 - 0.038V_2 + 0.172V_3 + 0.439X - 0.641X_2 - 0.796X_3 - 0.317R_{31} + 0.679R_{21} - 0.471$$

$$Y = -0.073V_1 - 0.034V_2 - 0.269V_3 + 0.278X_1 - 0.322X_2 + 0.414X_3 + 1.02R_{31} - 0484R_{21} + 0.845$$

Fig. 9 Formation of ternary intermetallic compounds of the systems with $3 \leq VE(3) \leq 6$ and $8 \leq VE(1) \leq 11$. (*open circle*) ternary compound formation; (*line*) no ternary compound

$$0.136 \leq 0.166V_1 - 0.038V_2 + 0.172V_3 + 0.439X_1 - 0.641X_2 - 0.796X_3 - 0.317R_3 + 10.679R_{21} \leq 0.789$$

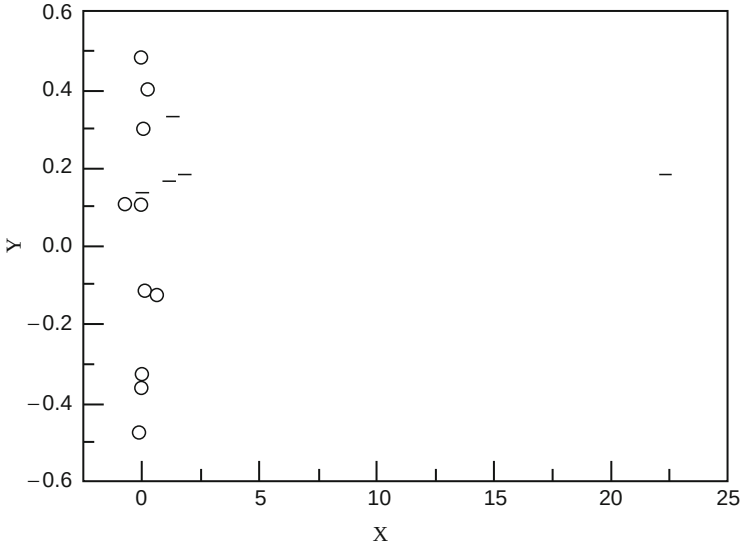
$$-1.236 \geq 0.073V_1 0.034V_2 - 0.269V_3 + 0.278X_1 - 0.322X_2 + 0.474X_3 + 1.021R_{31} - 0.484R_{21} \geq -0.471.$$

The regularities of ternary compound formation for the systems consisting of three early transition elements were investigated based on a subset data file containing 245 sample with $VE(1) \leq 6$. Figure 10 is a linear projection map of the representative points of the ternary compound forming systems and the systems without ternary compound formation for the systems consisting of three early transition elements.

Since the above-mentioned criteria for ternary compound formation are dealing with early transition elements and late transition elements, and the elements of Mn-group, Mn, W, and Re, are the borderline cases between early and late transition elements they should be treated separately. Figure 11 illustrates the distribution of all ternary alloy systems containing Mn, W, or Re. The criterion of ternary compound formation for Mn-group containing systems can be expressed as follows:

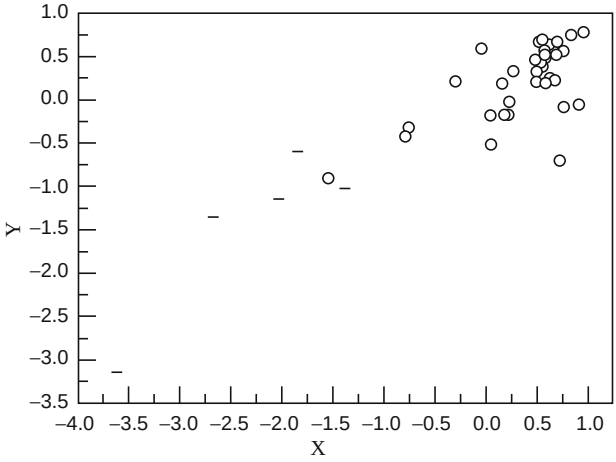
$$-1.564R_1 - 0.429R_2 + 0.043R_3 + 0.273V_1 + 0.050V_2 + 0.069V_3 - 1.745X_1 - 0.454X_2 + 1.695X_3 \geq 3.021.$$

To test the prediction ability of the model obtained, two newly discovered compounds of ternary intermetallic compounds between transition elements, Fe4NbTi and Ni4PtU5 [4, 5], were used as the objects for testing the prediction ability. Figure 12 illustrates the results of this "prediction". It is obvious that the representative points are located in the ternary compound-forming region and these systems should form ternary intermetallic compounds. This "prediction" result is in agreement with experimental facts.



$$X = -29.09R_{31} + 24.5R_{21} + 0.869V_1 - 0.823V_2 + 3.25V_3 - 18.1X_1 + 10.41X_{21} - 1.600$$
$$Y = 1.47R_{31} + 0.11R_{21} - 0.063V_1 - 0.239V_2 + 0.91V_3 - 2.61X_1 + 2.94X_{21} - 4.350$$

Fig. 10 Formation of ternary compounds between early transition elements. (*open circle*) ternary compound formation; (*line*) no ternary compound



$$X = -0.732R_1 + 0.197R_2 + 0.091R_3 - 0.156V_1 + 0.093V_2 - 0.056V_3 + 0.903X_1 - 1.732X_2 + 0.203X_3 - 4.798$$
$$Y = 0.449R_1 + 0.109R_2 - 0.181R_3 - 0.036V_1 - 0.1912 + 0.162V_3 + 0.370X_1 + 1.821X_2 - 2.00X_3 - 0.799$$

Fig. 11 Formation of ternary intermetallic compounds of Mn-group elements. (*open circle*) ternary compound formation; (*line*) no ternary compound

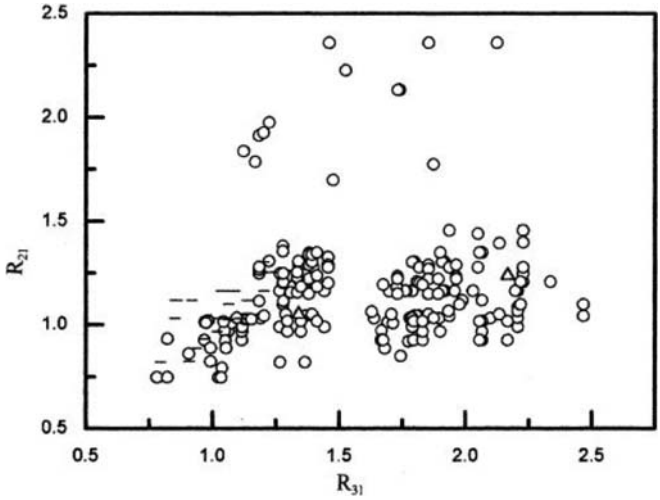


Fig. 12 Test of prediction ability for ternary compound formation between transition elements. (triangle) ternary compound formation; (line) no ternary compound; (triangle) samples for prediction

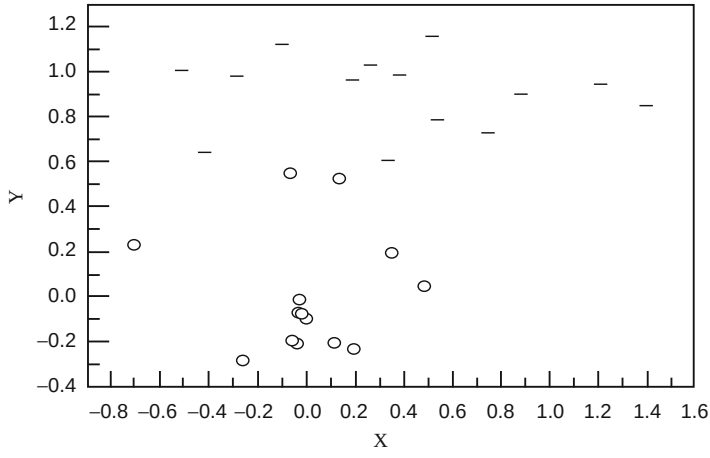
Intermetallic Compounds Between One Nontransition Element and Two Transition Elements

In Sects. 840 and 856, the regularities of intermetallic compound formation between nontransition elements, and that between transition elements have been discussed based on data mining work [3, 6]. In this section, the results of the investigation of ternary compound formation between two transition elements together with one nontransition element will be reported.

The atomic parameters and computational methods used in this work are similar to that in Sect. 840. In the data file for computation, the order of three element is $T - T' - M$, where T and T' are transition elements and the number of d electrons in unfilled shells of T is defined to be larger than that of T'. Here, M denotes some nontransition element.

Since transition elements can be classified as early transition elements (denoted by ET) and late transition elements (denoted by LT), the ternary alloy systems of $T - T' - M$ type can be classified into three groups: LT-LT–M, LT–ET–M systems, and ET–ET–M systems. In this work, the regularities of ternary compound formation of these three groups are investigated separately.

It has been found that the regularities of the formation of ternary compounds in the alloy systems having one common constituent element can be described by some projection maps obtained in pattern recognition work. For example, we can use pattern recognition technique to investigate the regularities of ternary compound formation of Ag-containing systems. Figure 13 illustrates a linear projection of the representative points from the multi-dimensional space spanned by R_{sp} and X_{MB} of elements.



$$X = 0.81R_1 - 0.11R_2 + 0.4V_1 - 0.096V_2 + 0.091X_1 - 0.572X_2 - 2.23$$
$$Y = -0.40\mathrm{R}_1 - 0.19R_2 - 0.22V_1 + 0.092V_2 + 0.66X_1 - 0.533X_2 + 1.42.$$

Fig. 13 Formation of ternary intermetallic compounds in Ag-containing alloy systems. (*open circle*) ternary compound formation; (*line*) no ternary compound

By this way, the criterion for ternary intermetallic compound formation can be expressed as follows:

$$0.691R2 - 0.002R3 + 1.868X2 + 0.743X3 \leq 3.481$$
$$-0.397R2 - 0.400R2 - 0.176X2 - 0.266X3 \leq -2.421$$

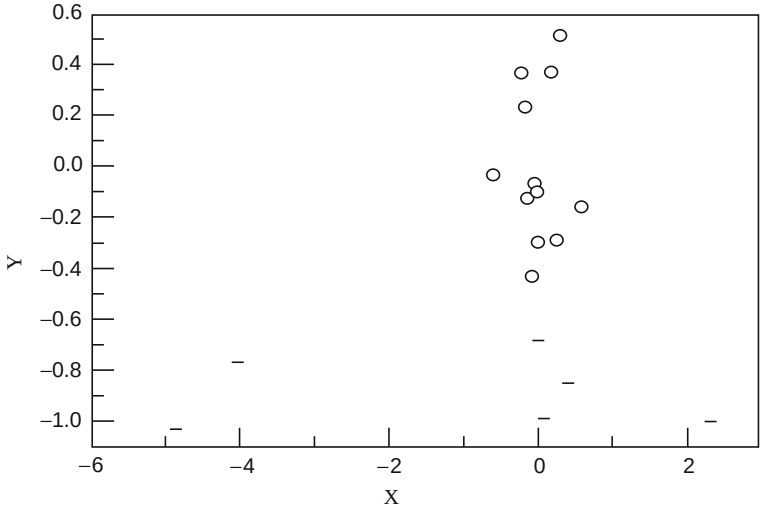
Here, R, V, X are the abbreviations of R_{sp}, VE and X_{MB}, respectively. It is also possible to investigate the regularities of a nontransition element combine with various pairs of transition elements. For example, the regularity of formation of Al-containing ternary compounds with two later transition elements can be shown by the linear projection map in Fig. 14. And the criterion for ternary compound formation can be expressed as follows:

$$1.661R12 + 0.0132V1 + 0.272V2 + 0.1039X12 \leq 4.89$$

Here, R_{mn} denotes R_m/R_n, and X_{mn} denotes $X_m - X_n$.

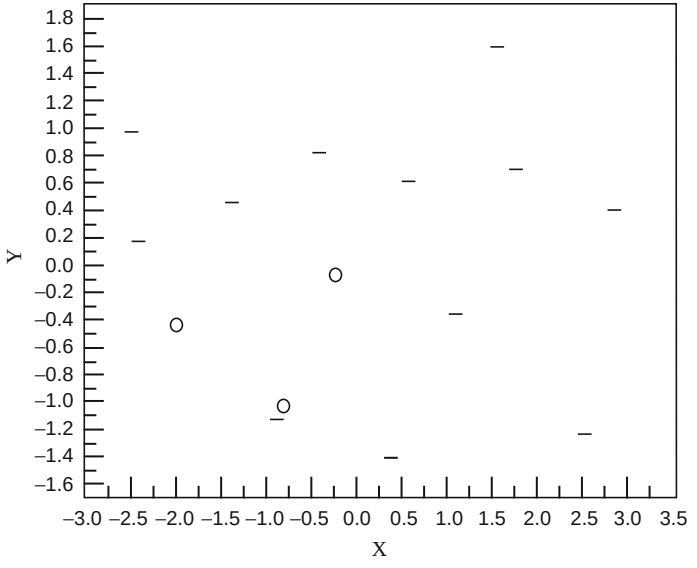
Since the behavior of ternary compound formation of the systems with two Cu-group elements seems different from other systems, the alloy systems with two Cu-group elements (Cu, Ag, Au) and one nontransition element are treated separately. Figure 15 illustrates the regularity of ternary compound formation of these systems. The criterion of ternary intermetallic compound formation for these systems can be expressed as follows:

$$14.23 \leq 1.602R2 - 1.263R3 + 0.4758V3 + 9.027X2 + 1.831X3 \leq 15.99$$



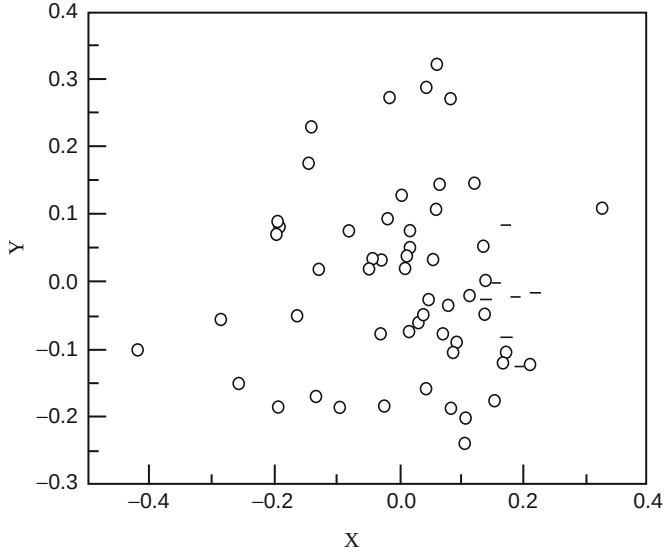
$$X = -2.64R_1 + 0.51R_2 + 0.22V_1 + 0.01V_2 + 6.16X_1 - 21X_2 - 3.948$$
$$Y = 0.10R_1 - 1.62R_2 + 0.14V_1 + 0.07V_2 + 0.30X_1 + 140X_2 - 1.170$$

Fig. 14 Formation of ternary intermetallic compounds in Al-containing alloy systems. (*open circle*) ternary compound formation; (*line*) no ternary compound



$$X = 1.6R_2 + 1.3R_3 + 0.48V_3 + 9.03X_2 + 1.83X_3 - 16.23$$
$$Y = 0.62R_2 + 4.25R_3 + 0.15V_3 + 3.49X_2 + 0.17X_3 - 13.98$$

Fig. 15 Formation of termary intermetallic compounds between two Cu-group Elements and one nontransition element. (*open circle*) temary compound formation; (*line*) no ternary compound



$X = 0.13R_1 - 0.18R_2 + 0.22R_3 + 0.08V_1 - 0.04V_2 - 0.04V_3 - 0.14X_1 - 0.25X_2 + 0.18X_3 - 0.24$

$Y = 0.20R_1 - 0.27R_3 - 0.04V_1 + 0.01V_2 + 0.57V_3 + 0.13X_1 + 0.13X_3 + 1.76X_3 + 1.085.$

Fig. 16 Formation of ternary intermetallic compounds between two late nontransition elements and one nontransition element. (*open circle*) ternary compound formation; (*line*) no ternary compound

$$12.95 \leq 0.6197R2 + 4.245R3 + 0.1496V3 + 3.491X2 + 0.1670X3 \leq 13.91$$

Figure 16 illustrates the regularity of ternary intermetallic compound formation of the systems containing two late transition elements and one nontransition element (but not including the systems with two Cu-group elements). It can be seen that most of these ternary systems form ternary intermetallic compounds, only a few of them do not form intermetallic compound. The criterion for the systems without ternary compound formation can be expressed as follows:

$$2.04 \leq R_1 \leq 2.38$$
$$2.11 \leq R_2 \leq 2.22$$
$$1.42 \leq R_3 \leq 2.09$$
$$10 \leq V_1 \leq 11$$
$$1.07 \leq X_2 \leq 2.04$$
$$1.67 \leq X_3 \leq 2.04$$
$$1.64 \leq X_3 \leq 2.14$$

$$-1.38R_1 - 9.24R_2 - 1.04R_3 - 0.34V_1 + 0.29V_2 + 0.66V_3 + 0.11X_1 - 3.69X_2 - 2.84X_3 \geq -21.94$$

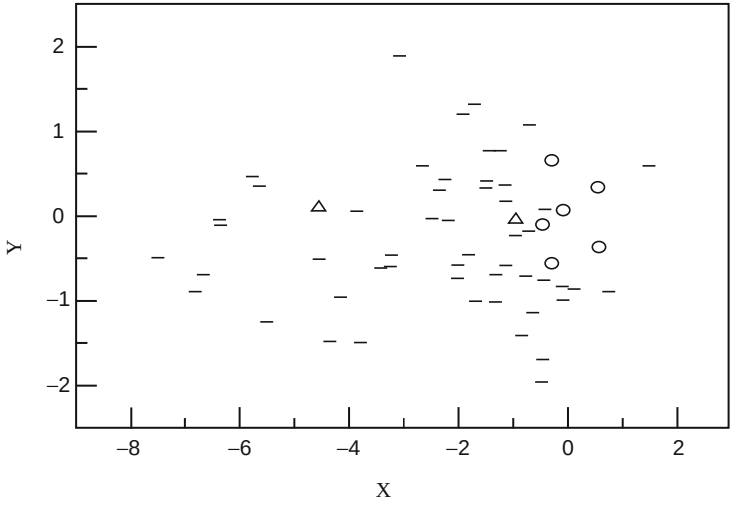
In Villars's Data Bank of Ternary Alloy Systems, it can be found that the number of ternary intermetallic compounds between one early transition element, one late transition element and one nontransition element is very large. Ternary compounds of more than 1,500 ternary alloy systems are stored in this Data Bank. At the same time, all ternary systems of this kind in this Data Bank have records of ternary intermetallic compound formation. On the other hand, there are some experimental records about the perfect stability of Fe–Cr or Ni–Cr alloy in liquid alkali metals [7]. It seems believable that the system of Fe–Cr or Ni–Cr do not form ternary intermetallic compounds with alkali metal such as Na and K. This fact can be understood as follows: the spd hybridization between the d-orbitals of transition elements and the s or p electrons of the nontransition element are the chief driving forces to enhance the ternary compound formation. So the energy levels of the s or p electrons of nontransition element must be deep enough to enhance the spd hybridization. Since the electronegativity of an element is related to the energy level of its valence electrons, it is reasonable to define a margin value of electronegativity of a nontransition element for the formation of ternary compound with an early transition element and a late transition element. For the XMB scale of electronegativity, this margin value seems near 1.45. Therefore, Al, Ga, In, Tl, Si, Ge, Sn, Pb, B, As, Sb, and Bi are the elements most easily to form ternary compounds with one early transition element and one later transition element, while alkali metals cannot form ternary compound of this type.

Based on this concept, it seems possible to use a simple argument for the ternary compound formation: if a nontransition metallic element or metalloid having $X_{MB} \geq 1.45$, it can combine with one early transition element (or a rare earth element) and one late element to form ternary intermetallic compound.

Two lately discovered ternary intermetallic compounds, $CuPd_{13}$ Si_{13} and $RhFe_5Al_4$ indicate that Cu–Pd–Si and Rh-Fe–Al systems are ternary compound-forming systems. These two systems, not included in the training set mentioned in this chapter, are used as the objects for "computerized prediction" based on the regularities found. Figure 17 illustrates the results of computerized prediction by the linear projection method. It can be seen that the representative points of these predicted systems are located in the zone of ternary compound formation. These results are in agreement with the experimental facts [8, 9].

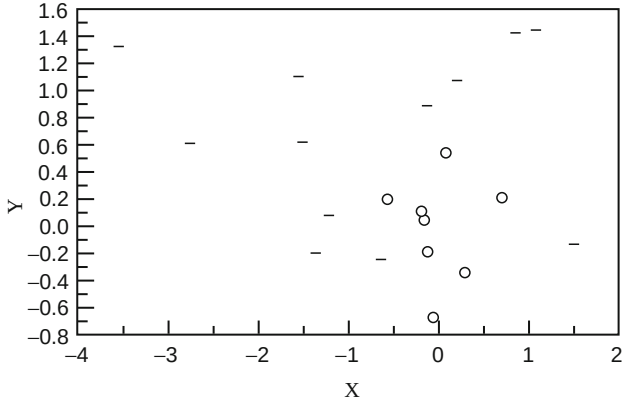
Intermetallic Compounds Between Two Nontransition Elements and One Transition Element

In our previous works, the regularities of the formation of ternary intermetallic compounds between nontransition elements, transition elements, or one kind of nontransition element with two kinds of transition elements have been studied by the atomic parameter-pattern recognition method, with good results [3, 6, 10]. In this section, we extend this strategy to investigate the regularities of the formation of ternary intermetallic compounds consisting of one kind of transition element and two kinds of nontransition elements.



$$X = -6.58R_{21} + 9.18R_{32} - 1.507V_1 - 0.196V_2 - 2.64V_3 + 5.67X_{21} + 9.956X_{32} + 22.44$$
$$Y = -3.02R_{21} - 1.13R_{32} - 0.10V_1 - 0.05V_2 + 1.39V_3 - 0.17X_{21} - 1.90X_{32} + 0.44.$$

Fig. 17 Computerized prediction of ternary compound by the linear projection method. (*line*) ternary compound formation; (*open circle*) no ternary compound; (*triangle*) predicted sample point

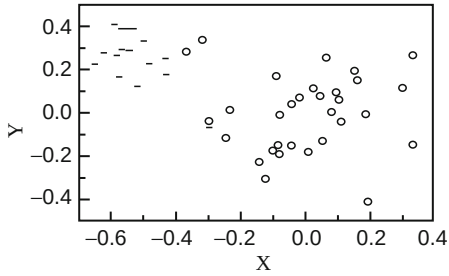


$$X = 2.16R_2 - 5.3R_3 - 1.2V_2 + 5.31V_3 + 5.67X_2 - 20.9X_3 + 17.8;$$
$$Y = -0.19R_2 + 2.61R_3 - 0.06V_2 - 0.35V_3 - 1.39X_2 + 4.14X_3 - 7.97.$$

Fig. 18 Regularity of formation of ternary compounds of Fe-containing systems. (*open circle*) ternary compound formation; (*line*) no ternary compound

Figure 18 illustrates a linear projection map showing ternary compound forming systems and systems without ternary compounds for Fe-containing alloys.

Figure 19 illustrates the regularities of ternary compound formation for early transition element containing systems by using the pattern recognition diagram.



$$X = 1.46R_{31} - 0.17R_{21} + 0.14V_1 - 0.27V_2 - 0.40V_3 + 0.73X_{31} - 1.13X_{32} + 2.77X_3 - 3.95.$$
$$Y = 1.80R_{31} - 0.05R_{21} + 0.03V_1 + 0.49V_2 - 0.23V_3 + 0.74X_{31} + 1.25X_{32} + 1.38X_3 + 0.45.$$

Fig. 19 Regularity of ternary compound formation for early transition element containing systems. (*open circle*) ternary compound formation; (*line*) no ternary compound

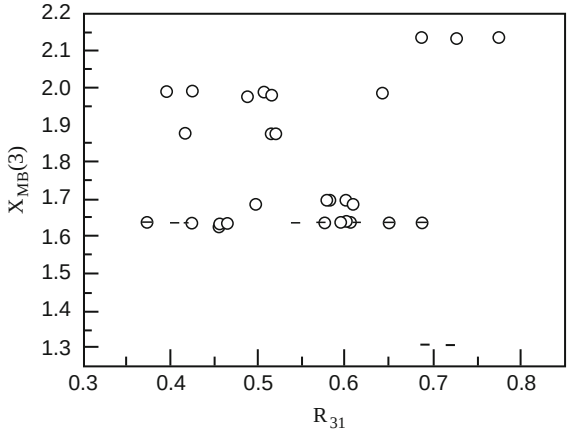
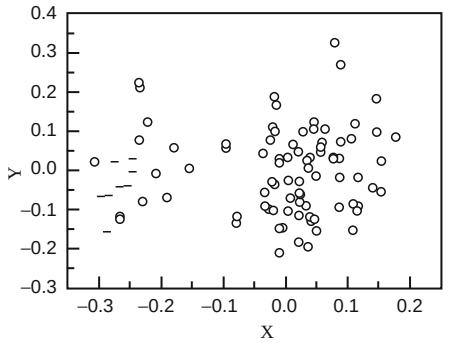


Fig. 20 Bivariate map showing the influence of $X_{\mathrm{MB}}(3)$ on the ternary compound formation. (*open circle*) ternary compound formation; (*line*) no ternary compound

From bivariate maps it can be seen that the most important factor dominating ternary compound formation is $X_{MB}(3)$, as illustrated in Fig. 20. This fact can be explained as follows: $X_{\mathrm{MB}}(3)$ is usually the higher value of the electronegativities of these two nontransition elements. A higher value of $X_{\mathrm{MB}}(3)$ corresponds to deeper energy levels of the *s* or *p* electrons in the atoms of nontransition element. The deeper energy levels of *s* or *p* electrons favor the *spd* hybridization between the atoms of transition element and nontransition elements. This *spd* hybridization will stabilize the chemical bond between the atoms of the first element and that of the third element. If the electronegativity of the second element is also rather high (for example, in the case of system V–Si–Sb), the *s* and *p* electrons of both nontransition elements will take part in the *spd* hybridization and make the ternary compound stable. On the other hand, if the electronegativity of the second element is rather low (for example, in the case of the system Ti-Li–Bi),



$X = 0.58R_{31} - 0.08R_{21} + 0.21V_1 + 0.02V_2 - 0.19V_3 + 0.24X_{31} + 0.14X_{32} + 0.46X_3 - 2.60$
$Y = 0.50R_{31} - 0.15R_{21} + 0.02V_1 + 0.07V_2 - 0.13V_3 + 0.03X_{31} + 0.06X_{32} + 0.09X_3 - 0.84.$

Fig. 21 Regularity of formation of ternary compounds containing late transition elements. (*open circle*) ternary compound formation; (*line*) no ternary compound

the electronegativity difference between the second and third elements will lead to charge transfer between the atoms of the second element and the third element, and the ionicity will stabilize the chemical bond between these atoms and thus make the ternary compound stable.

Figure 21 illustrates the regularities of formation of ternary intermetallic compounds containing late transition elements. The criterion of ternary compound formation can be roughly expressed as follows:

$$0.736R_{31} + 0.075R_{21} + 0.0242V_1 + 0.013V_2 - 0.242V_3 + 0.257X_{31} + 0.125X_{32} + 0.733X_3 \geq 3.066$$

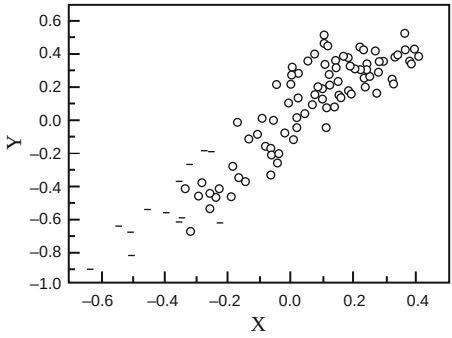
Regularities of Formation of Ternary Compounds Containing Cu-Group Elements

Figure 22 illustrates the regularity of the ternary intermetallic compound formation for Cu-group element-containing systems. It can be seen that the criterion of ternary compound formation can be roughly expressed as follows:

$$0.336R_{31} + 0.035R_{21} + 0.18V_2 - 0.55V_3 - 1.869X_{31} + 1.143X_{32} + 2.549 \geq 2.32$$

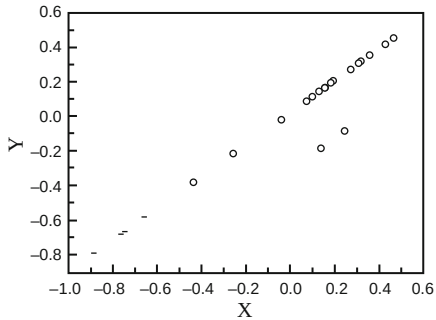
Figure 23 illustrates the regularity of formation of the ternary intermetallic compounds containing Mn and Re. It can be shown that the criterion of ternary compound formation can be expressed as follows:

$$2.06R_1 - 0.285R_2 + 0.128V_2 - 0.003V_3 - 51.38X_1 + 3.512X_3 \geq 94.95$$



$$X = 0.34R_{31} + 0.04R_{21} + 0.19V_2 - 0.55V_3 - 1.87X_{31} + 1.14X_{32} + 2.55X_3 - 265.$$
$$Y = -0.69R_{31} + 0.02R_{21} + 0.33V_2 - 0.05V_3 - 0.26X_{31} + 2.00X_{32} - 2.55X_3 + 3.32.$$

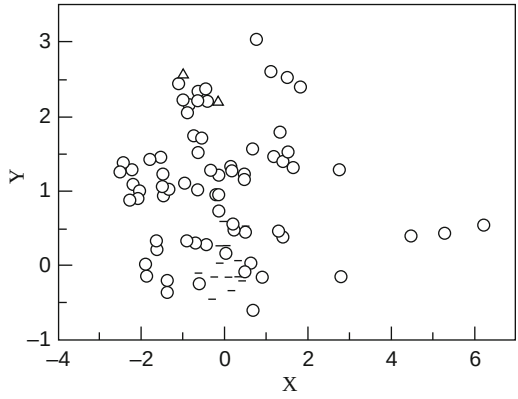
Fig. 22 Regularity of formation of ternary compounds containing Cu-group elements. (*open circle*) ternary compound formation; (*line*) no ternary compound



$$X = 1.77R_1 - 0.21R_2 + 0.51R_3 + 0.06V_2 - 0.43V_3 - 29.45X_1 - 0.47X_2 + 3.44X_3 - 51.28.$$
$$Y = -2.59R_1 - 0.20R_2 + 0.47R_3 + 0.06V_2 - 0.40V_3 - 57.78X_1 - 0.43X_2 + 3.19X_3 - 106.4.$$

Fig. 23 Regularity of formation of ternary intermetallic compounds of Mn-group elements. (*open circle*) ternary compound formation; (*line*) no ternary compound

Two lately discovered ternary intermetallic compounds, Au-Ca–Ga and $Au_3Ba_4Si_2O$, indicate that Au-Ca–Ga and Au-Ba–Si systems are ternary-compound forming systems. These two systems are not included into the training set and are used as the objects of computerized prediction. The results are shown in Fig. 24. It can be seen that these predicted objects are located in the zone of ternary compound formation. These results are in agreement with the experimental tests [11–13].



$$X = -0.78R_{21} + 1.98R_{32} - 0.62V_2 + 3.40V_3 + 12.4V_{12} + 14.0X_{23} - 2.41.$$
$$Y = 0.73R_{21} - 1.14R_{32} + 1.07V_2 + 1.18V_3 + 9.16V_{12} + 4.73X_{23} - 1.16.$$

Fig. 24 The results of test of prediction ability of pattern recognition method. (*open circle*) ternary compound formation; (*line*) no ternary compound; (*triangle*) sample points of predicted systems

2 Data-Mining-Based Prediction on Structure–Activity Relationships of Drugs

The research on structure–activity relationship (SAR) plays a very important role in the process of drug development. SAR researches can be classified into two categories: quantitative structure–activity relationships (QSAR) and qualitative structure–activity relationships (QSAR). In QSAR analysis, biological activity is quantitatively expressed as a function of physico-chemical properties of molecules. QSAR involves modeling a continuous activity for quantitative prediction of the activity of new compounds. QSAR aims to separate the compounds into a number of discrete types, such as active and inactive. It involves modeling a discrete activity for qualitative prediction of the activities of new compounds.

With the increasing demand for the research of SAR, the number of data mining methods introduced into SAR has increased. Since there are a lot of data mining methods including multiple linear or nonlinear regression (MLR or NLR), partial least-square regression (PLSR), artificial neural networks (ANN), genetic algorithms (GA), principal component analysis (PCA), multiple discriminate vector (MDV), nonlinear mapping (NLM), decision trees (DT), wavelet transform (WT), etc., one has to deal with the troublesome problem about model selection for a particular data set with finite number of samples and multiple descriptors. It is very important to select a proper model with good generalization ability, i.e., high-prediction accuracy or low-mean relative error for the activities of new compounds (unseen samples).

Generally speaking, the modeling problem is actually ill-posed in the sense of Hadamard [14]. So, how to choose the right balance between model flexibility and

overfitting to a limited training set is one of the most difficult obstacles for obtaining a model with good generalization ability [15]. Since the number of samples available in a real world data set is finite or relative small, the problem of overfitting in artificial neural network or nonlinear regression methods may become a rather serious problem in some cases. As an effective way to overcome the problem of overfitting, support vector machine (SVM) based on statistical learning theory (SLT) has been proposed by V.N. Vapnik [16]. Now, SVM has gained successful application in such research fields as vowel recognition [17], text recognition [18], time series forecasting [19], combinatorial chemistry [20], proteins [21, 22], drug design [23], etc. We also reported a series of applications of SVM in Chemistry [24].

The SVM methods include the support vector classification (SVC) and support vector regression (SVR) algorithms [25]. In this section, two cases studied are given to prove the significance of SVM application in the field of SAR. One case is QSAR problem investigated by using SVC model. The other is QSAR problem studied by using SVR model.

2.1 Methodology

Support Vector Classification [16, 25]

SVC has been recently proposed as a very effective method for solving pattern recognition problems. The geometrical interpretation of SVC is that it chooses the optimal separating surface, i.e. the hyperplane equidistant from two classes. This optimal separating hyperplane has many nice statistical properties as detailed in Vapnik [16].

Consider the problem of separating the set of training vectors belonging to two separate classes, $(y_1, x_1)..(y_n, x_n), x \in R^m, y \in -1, +1$ with a hyperplane $w^T x + b = 0$

If the training data are linearly separable, then there exists a pair (w, b) such that

$$y_i(w^T x_i + b) - 1 \geq 0, \quad i = 1, 2, \ldots, l$$

$$w^T x + b \geq +1 \quad for \ \text{all}\ x \in T$$
$$w^T x + b \leq -1 \quad for \ \text{all}\ x \in F$$

The decision rule is:

$$f_{w,b}(x) = sgn(w^T x + b),$$

where w is termed the weight vector and b the bias. Without loss of generality the pair (w, b) can be rescaled such that:

$$\min_{i=1,2,\ldots,l} |w^T x_i + b| = 1$$

The learning problem is hence reformulated as: minimize $||w||^2$ subject to the constraints of linear separability. This is equivalent to maximizing the distance, normal to the hyperplane, between the convex hulls of two classes. The optimization is now a quadratic programming (QP) problem:

$$\underset{w,b}{\text{Minimize}}\,\theta(w) = \frac{1}{2}||w||^2$$

subject to: $y_i(w^T x_i + b) \geq 1, \;\; i = 1, 2, ..., l$.

This problem has global optimum. The Lagrangian for this problem is:

$$L(w, b, \Lambda) = \frac{1}{2}||w||^2 - \sum_{i=1}^{l} \lambda_i [y_i (w^T x_i + b) - 1],$$

where $\Lambda = \lambda_1, \ldots, \lambda_l$ are the Lagrange multipliers, one for each data point.

Hence, we can write:

$$F(\Lambda) = \sum_{i=1}^{l} \lambda_i - \frac{1}{2}||w||^2 = \sum_{i=1}^{l} \lambda_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \lambda_i \lambda_j y_i y_j x_i^T x_j$$

note that the Lagrange multipliers are only nonzero when $y_i(w^T x_i + b) = 1$, vectors for these cases are called support vectors since they lie closest to the separating hyperplane. Then the optimal separating hyperplane is given by,

$$w^* = \sum_{i=1}^{l} \lambda_i^* x_i y_i$$

and the bias is given by:

$$b^* = -\frac{1}{2}(w^*)^T (x_s + x_r),$$

where x_r and x_s are any support vector from each class satisfying,

$$y_r = 1, y_s = -1$$

The hard classifier is then,

$$f(x) = sgn[(w^*)^T x + b^*].$$

In the case where a linear boundary is inappropriate, the SVC can map the input vector, x, into a high-dimensional feature space, F. By choosing a nonlinear mapping Φ, the SVC constructs an optimal separating hyperplane in this higher

dimensional space. Among acceptable mappings are polynomials, radial basis functions and certain sigmoid functions. Then, the optimization problem becomes

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \langle \Phi(x_i).\Phi(x_j) \rangle.$$

In this case, the decision function in SVC is as follows:

$$g(x) = sgn(f(x)) = sgn\left(\sum_{i \in SV} \alpha_i y_i \langle \Phi(x).\Phi(x_i) \rangle + b \right)$$
$$= sgn\left(\sum_{i \in SV} \alpha_i y_i K(x, x_i) + b \right),$$

where the x_i is the support vectors and $K(x, x_i)$ is called kernel function equal to the inner product of two vectors x_i and x_j in the feature space $\Phi(x)$. The elegance of using kernel function lied in the fact that one can deal with feature spaces of arbitrary dimensionality without having to compute the map $\Phi(x)$ explicitly. Any function that satisfies Mercer's condition can be used as the kernel function.

Support Vector Regression (SVR) [16, 25]

SVM can be applied to regression by the introduction of an alternative loss function and the results appear to be very encouraging. In support vector regression (SVR), the basic idea is to map the data X into a higher dimensional feature space F via a nonlinear mapping Φ and then to do linear regression in this space. Therefore, regression approximation addresses the problem of estimating a function based on a given data set $G = \{(x_i, d_i)\}_{i=1}^{l}$ (x_i is input vector, d_i is the desired value). SVR approximates the function in the following form:

$$y = \sum_{i=1}^{l} w_i \Phi_i(x) + b,$$

where $\{|phi_i(x)\}_{i=1}^{l}$ is the set of mappings of input features, and $\{w_i\}_{i=1}^{l}$ and b are coefficients. They are estimated by minimizing the regularized risk function $R(C)$:

$$R(C) = C \frac{1}{N} \sum_{i=1}^{N} L_\epsilon(d_i, y_i) + \frac{1}{2}||w||^2,$$

where

$$L_\epsilon(d, y) = \begin{cases} |d - y| - \epsilon & \text{for } |d - y| \geq \epsilon \\ 0 & \text{otherwise} \end{cases}$$

and ϵ is a prescribed parameter. $(C/N)\sum_i^N L_\epsilon(d_i, y_i)$ is the so-called empirical error (risk) measured by ϵ-insensitive loss function $L_\epsilon(d, y)$, which indicates that it does not penalize errors below ϵ. $(1/2)||w||^2$ is used as a measurement of function flatness. C is a regularized constant determining the tradeoff between the training error and the model flatness. Introduction of slack variables "ζ" leads to the following constrained function:

$$MaxR(w, \zeta^*) = \frac{1}{2}||w||^2 + C^* \sum_{i=1}^{n}(\zeta_i + \zeta_i^*),$$

$$\text{s.t.} w\Phi(x_i) + b - d_i \leq \epsilon + \zeta_i,$$

$$d_i - w\Phi(x_i) - b_i \leq \epsilon + \zeta_i,$$

$$\zeta, zeta^i \geq 0.$$

Thus, decision function becomes the following form:

$$f(x, \alpha_i, \alpha_i^*) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)K(x.x_i) + b,$$

where α_i, α_i^* are the introduced Lagrange multipliers, satisfying the equality $\alpha_i.\alpha_i^* = 0, \alpha_i \geq 0, \alpha_i^* \geq 0$; $K(x_i, x_j)$ is the kernel function; $b = -\frac{1}{2}\sum_l(\alpha_i - \alpha_i^*)[K(x_r - x_i) + K(x_s - x_i)]$, $i = 1, \ldots, l$£.

Based on the Karush-Kuhn–Tucker (KKT) conditions of quadratic programming, only a number of coefficients $(\alpha_i - \alpha_i^*)$ will assume nonzero values, and the data points associated with them could be referred to as support vectors.

Implementation of Support Vector Machine (SVM)

Referring to the literature [16, 26], the SVM software package including SVC and SVR was programmed in our lab [27]. The validation of the software was tested in some applications in chemistry and chemical engineering [24, 28]. All computations were carried out on a Pentium IV computer with a 2.66G Hz processor.

2.2 Using Support Vector Classification for Anti-HIV-1 Activities of HEPT-Analog Compounds

The pandemic form of sexually transmitted human immunodeficiency virus (HIV) that causes the acquired immunodeficiency syndrome (AIDS) has lead to the development of various non-nucleoside reverse transcriptase inhibitors (NNRTIs), which are less toxicand chemically more stable, have slower metabolizing rate and are

slower emitted from the human body than nucleoside reverse transcriptase inhibitors (NRTIs). Among the non-nucleoside analogues, 1-[(2-hydroxyethoxy)methyl]-6-(phenylthio)-thymine (HEPT) is a potent inhibitor of HIV-1 reverse transcriptase (RT), an enzyme, which is necessary for the catalytic formation of proviral DNA from viral RNA [29].

In this work, we apply the support vector classification (SVC) method to investigate the structure–activity relationship (SAR) of the HEPT-analogue compounds with anti-HIV-1 activities. We constructed the qualitative model on the basis of the SVC, with structural descriptors calculated using the HYPERCHEM software. The performance of SVC proved to be outstanding, which underlines the significance of this method.

Data set

In order to test the applicability of the method in quantitative structure–activity relationships we used a data set consisting quantitative activities for 36 deoxy HEPT analogues [30]. The general formula of the HEPT-analogue compounds studied in this work is shown in Fig. 25. Our data set can be sorted into two classes, “Class 1” and “Class 2”, containing the compounds with high and low anti-HIV-1 activities, i.e. those with $EC_{50} < 1$ and $EC_{50} > 1 \mu mol/L$, respectively.

Descriptors

Generally speaking, molecular structures can be described by various descriptors, including electronic parameters, geometric parameters and hydrophobic parameters. We defined 15 descriptors in this work. First we optimised 3D structures of the molecules by molecular mechanics as implemented in the HYPERCHEM software [31] using the MM+ force field with the Polak-Ribiere algorithm until the root-mean-square gradient became less than 0.1 Kcal/mol. Quantum chemical parameters were obtained for the most stable conformation of each molecule

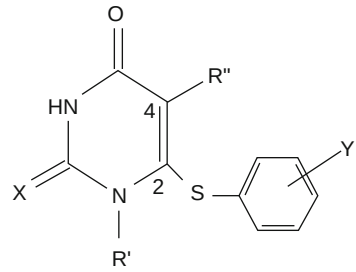


Fig. 25 Structures of the HEPT-analog compounds

by using the PM3 semiempirical molecular orbital method at the restricted Hartree-Fock level with no configuration interaction [32], other parameters were obtained by using the appropriate option of the HYPERCHEM software. The descriptors obtained are as follows: HOMO (highest occupied molecular orbital) energy, LUMO (lowest unoccupied molecular orbital) energy, ?E (difference between HOMO and LUMO), TE (total energy), HF (heat of formation), EE (total electronic energy), SA (surface area), MV (molecular volume), logP (partition coefficient), MR (molecular refractivity), MP (molecular polarisability), MD (molecular dipole) and MW (molecular weight), Q_2 and Q_4 (net charges on atoms 2 and 4, respectively).

Owing to the redundancy of some parameters, the selection of descriptors for a QSAR study is not straightforward. However, the selection of descriptors will contribute a lot to construct the actual model. Generally, the presence of irrelevant or redundant descriptors can cause the model to focus attention on the idiosyncrasies of individual samples and lose sight of the broad picture that is essential for generalization beyond the training set. Especially, this problem is compounded when the number of observations is also relatively small. In this work, we applied the entropy method [33] for the given data set of 15 measurements to choose a subset of descriptors. In the computation, the rate of separability R ($R = 1 - N_2/N_1$) was defined as a criterion to select the key descriptors that determined the anti-HIV-1 activities of the compounds. Here, N1 was the number of the samples with high activities predicted correctly, while N2 was the number of the samples with low activities misclassified. We supposed that the entropy of data set was measured by R value. Therefore, six selected descriptors (Q4, MV, LogP, MR, MP, and MW) were found suitable to construct the SVC model without degrading R value or increasing the entropy of data set. The results indicated that the anti-HIV-1 activities were affected by Q4 and MP as important electronic factors, which were the integrated embodiments of induction, resonance and hydrogen bond effect as well. At the same time, MV and MR were significant geometric factors; LogP and MW were the important hydrophobic factors, which were represented in the selected features and the derived model. Table 1 consists of anti-HIV-1 activities EC50 and the reduced subset of six descriptors. Further combinations of descriptors may exist, which may be useful for the classification of the data set used here, but the selected subset proved to be appropriate for a successful prediction of activities, therefore we did not look for further sets.

Selection of the Kernel Function and the Capacity Parameter C

Similarly to other multivariate statistical models, the performance of SVC is related to the number of dependent and independent variables as well as the combination of parameters used. In the computation of SVC, we have to deal with the capacity (or regularization) parameter C and the kernel function adopted. In this work, we

Table 1 Descriptors, structures and anti-HIV-1 activities for 36 HEPT-analogs studied in this work

No.	X	R'	R''	Y	EC_50 (μmol/L)	Q4	MV (Å^3)	LogP	MR	MP	MW
1	O	$CH_2OCH_2CH_2OMe$	Me	H	8.7	−0.4921	922.31	1.98	84.19	33.13	322.38
2	O	$CH_2OCH_2CH_2OC_5H_{11-n}$	Me	H	55	−0.4295	1143.2	3.58	102.7	40.47	378.49
3	O	$CH_2OCH_2CH_2OCH_2Ph$	Me	H	20	−0.4610	1145.4	3.76	108.8	42.79	398.48
4	O	CH_2OMe	Me	H	2.1	−0.4874	778.59	2.14	73.14	28.82	278.33
5	O	CH_2Pr	Me	H	3.6	−0.4386	857.01	3.15	81.16	31.86	290.38
6	O	$CH_2OCH_2CH_2SiMe_3$	Me	H	32	−0.4849	1026.7	1.65	93.96	38.48	364.53
7	O	CH_2OCH_2Ph	Me	H	0.088	−0.4782	1004.9	3.92	97.76	38.48	354.42
8	S	CH_2OEt	Me	H	0.026	−0.5121	911.29	4.00	90.78	35.65	322.44
9	S	CH_2OEt	Et	3,5-Me2	0.0044	−0.6025	1029.4	4.94	100.9	39.32	350.49
10	S	CH_2OEt	Et	3,5-Cl2	0.013	−0.5121	1002.0	5.04	100.4	39.51	391.33
11	S	$CH_2 - i - Pr$	Et	H	0.22	−0.5581	921.26	4.67	93.92	36.85	320.47
12	S	$CH_2OCH_2 - c - Hex$	Et	H	0.35	−0.4807	1121.8	5.56	111.8	44.06	390.56
13	S	CH_2OCH_2Ph	Et	H	0.0078	−0.5307	1076.9	5.44	110.6	43.48	384.51
14	S	CH_2OCH_2Ph	Et	3,5-Me2	0.0069	−0.4316	1191.5	6.37	120.7	47.15	412.56
15	S	$CH_2OCH_2C_6H_4(4-Me)$	Et	H	0.078	−0.4460	1129.9	5.91	115.7	45.31	398.54
16	S	$CH_2OCH_2C_6H_4(4-Cl)$	Et	H	0.012	−0.4780	1118.0	5.96	115.4	45.41	418.96
17	S	$CH_2OCH_2CH_2Ph$	Et	H	0.091	−0.5349	1128.8	5.69	115.4	45.31	398.54
18	S	CH_2OEt	i-Pr	H	0.014	−0.4564	959.74	4.34	95.33	37.49	336.47
19	S	CH_2OCH_2Ph	i-Pr	H	0.0068	−0.4761	1120.8	5.77	115.2	45.31	398.54
20	S	CH_2OEt	c-Pr	H	0.095	−0.4433	932.38	3.83	93.52	36.72	334.45
21	O	CH_2OEt	Et	H	0.019	−0.5970	890.71	2.88	82.49	32.49	306.38
22	O	CH_2OEt	Et	3,5-Me2	0.0054	−0.6257	997.32	3.82	92.57	36.16	334.43
23	O	CH_2OEt	Et	3,5-Cl2	0.0074	−0.4855	976.42	3.92	92.10	36.35	375.27
24	O	$CH_2 - i - Pr$	Et	H	0.34	−0.5363	932.75	3.30	86.91	34.33	320.41
25	O	$CH_2OCH_2 - c - Hex$	Et	H	0.45	−0.5732	1090.9	4.44	103.5	40.90	374.5
26	O	CH_2OCH_2Ph	Et	H	0.0059	−0.5893	1057.1	4.32	102.4	40.32	368.45
27	O	CH_2OCH_2Ph	Et	3,5-Me2	0.0032	−0.5956	1160.5	5.25	112.4	43.99	396.5
28	O	$CH_2OCH_2CH_2Ph$	Et	H	0.096	−0.5897	1109.9	4.57	107.1	42.15	382.48
29	O	CH_2OEt	i − Pi	H	0.012	−0.5153	929.99	3.21	87.04	34.33	320.41
30	O	CH_2OCH_2Ph	i-Pi	H	0.0027	−0.6302	1091.7	4.65	106.9	42.15	382.48
31	O	CH_2OEt	c-Pi	H	0.1	−0.5675	901.01	2.71	85.24	33.55	318.39
32	O	H	Me	H	250	−0.4319	655.81	1.70	62.39	24.52	234.27
33	O	Me	Me	H	150	−0.4319	699.62	1.94	67.29	26.35	248.3
34	O	Et	Me	H	2.2	−0.4913	748.51	2.28	72.04	28.19	262.33
35	O	Bu	Me	H	1.2	−0.4990	856.21	3.15	81.16	31.86	290.38
36	O	$CH_2OCH_2CH_2OH$	Me	H	7	−0.4770	864.37	1.70	79.44	31.30	308.35

applied the leaving-one-out cross-validation (LOOCV) procedure to find the suitable capacity parameter C and the appropriate kernel function for the SVC model. The better is the model, the smaller is P_w, the number of wrong predictions using the LOOCV method. Therefore, we employed P_w as a criterion to determine the appropriate kernel function and the optimal capacity parameter C. Figure 26 depicts P_w versus the capacity parameter C from 0.1 to 250 with different kernel functions including linear, radial basis, polynomial and sigmoid functions. It was found that the radial basis kernel function with $C > 50$ provides the best performance for the SVC.

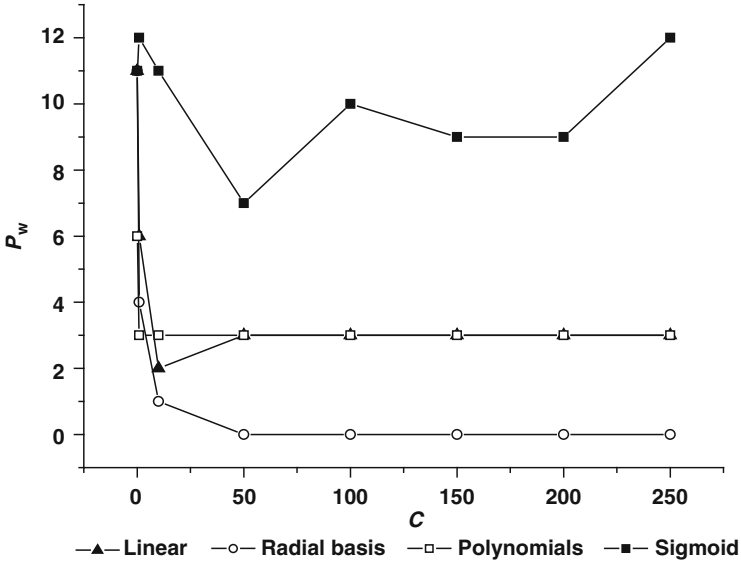


Fig. 26 Dependence of P_w on C using LOOCV with different kernel functions

Modeling of SVC

According to the results obtained in Sect. 902, the optimal SVC model for discriminating between high and low activities of compounds could be constructed using the radial basis kernel function with a capacity parameter $C = 100$:

$$g(x) = sgn\left(\sum_{i \in SV} \alpha_i y_i exp\{-\frac{||x - x_i||^2}{\sigma^2}\} + b\right),$$

where $\sigma = 1$ and $b = 0.977$ with $y_i = 1$ for the samples of "Class 1" and $y_i = -1$ for the samples of "Class 2." x is a sample vector with unknown activity to be discriminated, xi is one of the support vectors. Based on this SVC model, the samples were discriminated as those of high anti-HIV-1 activities ($EC_{50} < 1\mu mol/L$), if $g(x) \geq 0$. Using SVC model for the classification of anti-HIV-1 activities of the HEPT- analogue compounds, the accuracy of classification was 100% by using the radial basis kernel function with the capacity parameter $C = 100$. Figure 27 illustrates the trained and predicted results of classes of samples with SVC model.

Results of cross validation test

In the leaving-one-out cross-validation (LOOCV) test, the first compound is omitted, then a classification function is developed using the remaining ones, finally the omitted compound is classified. In a second step, the first compound is included and

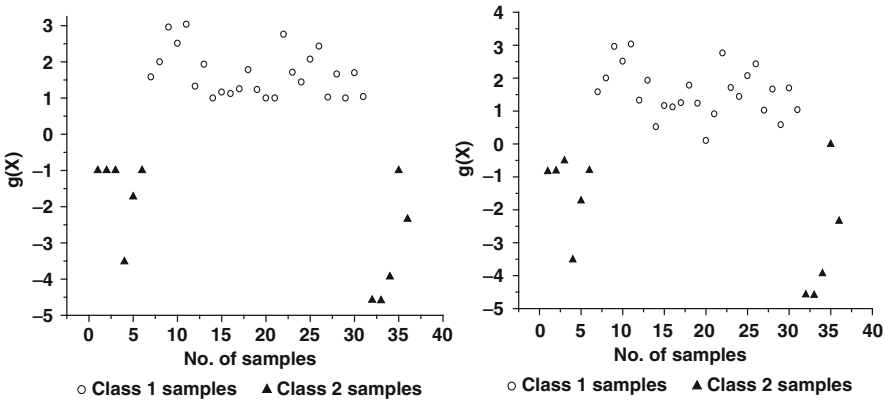


Fig. 27 The g(x) function for trained (*left*) and predicted (*right*) results for the HEPT derivatives

the second one is removed, then the procedure continues until the last compound is removed. Figure 27 (right) illustrates the positioning of classes of samples combining the LOOCV test with SVC. It is seen that none of the samples is inappropriately classified. These results underline the robustness of the SVC model since the prediction accuracy of the LOOCV test is 100% as compared to that of a stepwise discriminant analysis, which achieved only a prediction accuracy of 94% [30].

2.3 Predeicting Aldose Reductase Inhibitory Activities of Flavones Using Support Vector Regression

Diabetes mellitus is a major and growing public health problem throughout the world, with an estimated worldwide prevalence in 2000 of 150 million people, expected to increase to 220 million people by 2010 [34]. Although the pathology of diabetes is not clear, several pathogenic processes may be involved in the development of diabetes. Diabetic complications including cataract, neuropathy, and retinopathy are proved to be involved the accumulated sorbitol in diabetic cases. In healthy people, glucose is metabolized through the Embden- Meyerhoff pathway. In cases of diabetes mellitus, with the increased levels of glucose in insulin-insensitive tissues the Aldose Reductase (AR) in the polyol pathway facilitates the conversion of glucose to sorbitol [35, 36] Thus, if the inhibitors of AR (ARI) are potent enough and nontoxic, they not only prevent but also lead to the cure of the complications arising out of diabetes mellitus. Many of flavones were proved to be effective in inhibiting of AR activity. Matsuda and co-workers studied the AR inhibitory activity of a large number of flavones and related compounds from traditional antidiabetic remedies [37].

The study on quantitative structure–activity relationships (QSAR) of the Aldose Reductase (AR) inhibitory activity of flavones has important significance for

predicting inhibitory activity of flavones. Yenamandra S. Prabhakar and co-workers studied QSAR of the Aldose Reductase (AR) inhibitory activity of 48 flavones using Free-Wilson, Combinatorial Protocol in Multiple Linear Regression (CP-MLR), and Partial Least Squares (PLS) procedures [38].

In this work, support vector regression (SVR) was applied to QSAR study on the Aldose Reductase Inhibitory Activity of Flavones, to obtain a proper model with high accuracy of prediction for the activities of new compounds.

Data Set

The general formula of the flavones studied in this work is shown in Fig. 28. The structural information of 48 flavones and their reported rat lens aldose reductase (AR) inhibitory activity ($-logIC_{50}$, here IC_{50} is the concentration, in moles per liter, required to produce 50% inhibition) are listed in Table 2. In this work, 227 parameters were available as molecular descriptors: 158 parameters come from the reference [38], 50 were calculated by topology software including Auto correlation Topological Index, Extended adjacency index, AM index, Molecular connectivity index, Balaban index, Wiener index, Path/Walks Descriptors, Flexibility index, Xu index, MC index, Schultz Molecular Topological Index and some EI index, and other 19 parameters including electronic, geometric and some physicochemical property parameters were calculated by HyperChem Release 7 software.

Selection of Kernel Function and the Molecular Descriptors

To build an appropriate model with good predicting ability, a suitable kernel function was selected from linear kernel function, polynomial kernel function, and radial basis kernel function. Root mean squared error (RMSE) in LOOCV

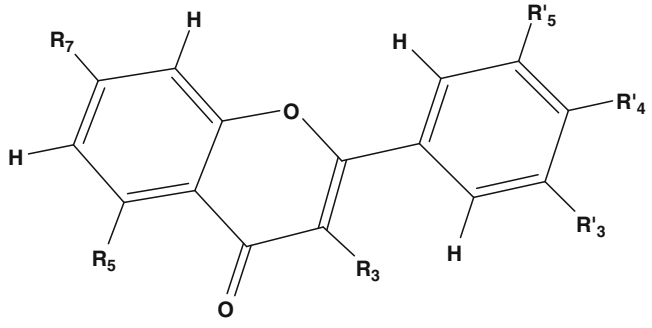


Fig. 28 General structure of the flavones (R3 and R7 may be H, OH, OMe, or O-glycoside and R5, R3, R4, and R5 may be H, OH, or OMe) associated with ratlens aldose reductase inhibitory activity

Table 2 Observed Aldose Reductase Inhibitory Activity of Flavones

Compd.	R3	R5	R7	R'3	R'4	R'5	$-logIC_{50}$ Exp.	$-logIC_{50}$ Pred.
1	H	H	OH	H	H	H	5.00	4.90
2	H	OH	OH	H	H	H	5.07	5.09
3	H	H	OH	H	OH	H	5.42	5.75
4	H	H	H	OH	OH	H	6.43	6.04
5	H	H	OH	OH	OH	H	6.52	6.07
6	H	OH	OH	H	OH	H	5.66	5.69
7	H	OH	SU1	H	OH	H	4.64	5.02
8	H	OH	SU2	H	OMe	H	5.33	4.56
9	H	OH	OH	OH	OH	H	6.35	6.13
10	H	OH	OH	OH	OMe	H	5.07	5.39
11	H	OH	OMe	OH	OMe	H	4.92	5.10
12	H	OH	OMe	OMe	OMe	H	4.14	4.44
13	H	OH	SU1	OH	OH	H	6.00	5.28
14	H	OH	SU3	OH	OH	H	5.51	5.88
15	H	OH	SU1	OH	OMe	H	4.64	4.52
16	OH	OH	OH	H	OH	H	5.00	5.25
17	SU3	OH	OH	H	OH	H	5.29	5.47
18	OH	OH	OH	OH	OH	H	5.66	5.64
19	OH	OH	OMe	OH	OH	H	5.57	5.38
20	OH	OH	OH	OH	OMe	H	4.96	4.92
21	OMe	OH	OMe	OH	OH	H	6.09	5.57
22	OH	OH	OMe	OH	OMe	H	5.22	4.63
23	OMe	OH	OMe	OH	OMe	H	4.47	4.84
24	OH	OH	OMe	OMe	OMe	H	4.14	3.90
25	OMe	OH	OMe	OMe	OMe	H	4.60	4.17
26	SU1	OH	OH	OH	OH	H	5.35	5.13
27	SU4	OH	OH	OH	OH	H	5.52	5.10
28	SU5	OH	OH	OH	OH	H	6.82	5.99
29	SU6	OH	OH	OH	OH	H	6.75	7.03
30	SUI	OH	SU1	OH	OH	H	4.08	4.41
31	SU2	OH	OH	OH	OH	H	5.05	5.66
32	SU2	OH	OMe	OH	OH	H	4.68	5.33
33	SU2	OH	OMe	OH	OMe	H	4.39	4.62
34	SU2	OH	OMe	OMe	OMe	H	4.06	3.97
35	OH	H	OH	OH	OH	H	5.43	5.65
36	OH	OH	OH	OH	OH	OH	4.54	5.19
37	OH	OH	OH	OH	OMe	OH	4.72	4.98
38	OH	OH	OMe	OH	OH	OH	4.68	4.84
39	OMe	OH	OMe	OH	OH	OH	4.92	5.10
40	OH	OH	OMe	OH	OMe	OH	4.62	4.62
41	OH	OH	OMe	OMe	OMe	OH	4.36	3.96
42	SU5	OH	OH	OH	OH	OH	5.42	5.45
43	SU5	OH	OH	OH	OMe	OH	5.42	5.13
44	SU5	OH	OMe	OH	OMe	OH	4.32	4.90
45	SU5	OH	OH	OMe	OMe	OH	4.68	4.56
46	SU5	OH	OMe	OMe	OMe	OH	4.15	4.28
47	SU5	OH	OMe	OMe	OMe	OMe	4.15	3.95
48	SU7	OH	OH	OH	OH	OH	7.09	6.65

In these compounds, the substituent groups corresponding to the SUgar moieties have been abbreviated as SU suffixed with a number as SU1:for O-β-D-glucopyranosyl; SU2 for O-β-D-glucopyranosyl(6 $\rightarrow$ 1)-O-α-L-rhamnopyranosyl; SU3 for :O-β-D-glucopyranosiduronic acid; SU4 for: O-β-D-galacotopyranosyl; SU5 for O-α-L-rhamnopyranosyl; SU6 for O-α-L-arabinopyranosyl; SU7 for O-(2-galloyl)-α-L-rhamnopyranosyl [38]

(leaving-one-out cross-validation) of SVR was used as the criteria of optimal set for QSAR analysis. In general, the smaller value of RMSE obtained, the better set of parameters gained. The RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(p_i - e_i)^2}{n}},$$

Where e_i is the experimental value of i sample, P_i is the predicted value of i sample, n is the number of the whole samples.

The changing tendency of RMSE in LOOCV of SVR with the regularization parameter C and epsilon under three kernel functions including linear kernel function, polynomial kernel function, and radial basis kernel function are given in Figs. 29 and 30.

Based on Figs. 29 and 30, it was found that the linear kernel function was suitable to build SVR model with minimum of RMSE among the three kernel functions.

Two hundred and twenty-seven descriptors were available as molecular descriptors in this work, the presence of irrelevant or redundant descriptors can cause the model to focus attention on the idiosyncrasies of individual samples and lose sight of the broad picture that is essential for generalization beyond the training set generally. Thus, a major focus in descriptors selection is the search for a set of molecular descriptors that give the best possible correlation between molecular structures and their inhibitory activities. In this work, either of descriptor i or j was deleted if the value of r_{ij} (correlation coefficient between descriptors i and j) was large than 0.9. After that, there were 56 descriptors left. Then, Genetic algorithm based on LOOCV of SVR was introduced for descriptors selection. Finally, the optimal model was obtained with the minimum of RMSE 0.39214 ($C = 10$, epsilon $= 0.15$) and seven selected descriptors: dChivps9 (Chi valence path differences of lengths "orders" 9), ESHaaCH (E-State of atom type HaaCH), EsssCH2

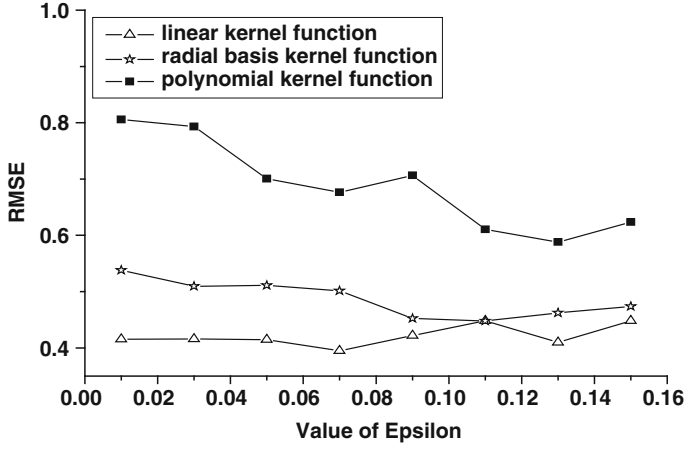


Fig. 29 RMSE in LOOCV vs. the epsilon with $C = 10$

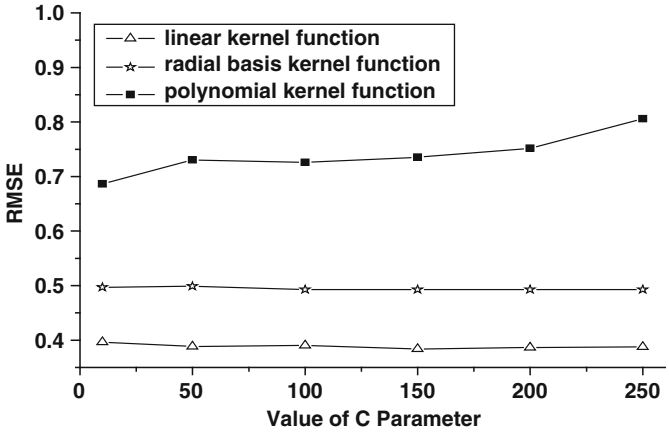


Fig. 30 RMSE in LOOCV vs. the C with epsilon = 0.15

(E-State of atom type ssCH2), n2Pag (count of alpha-gamma vertices), degree2 (degree number of connections=2), $I'3$ (defined indicator for R3-substituent position of 2-aryl-benzpyran-4-one; "1" for OH and "0" otherwise), $I'4$ (defined Indicator for R4-substituent position of 2-aryl-benzpyran-4-one; "1" for OH and "0" otherwise).

Optimization of SVR parameters

Figure 31 illustrates the parameters (C and epsilon) optimized in SVR model using seven descriptors with linear kernel function.

From Fig. 31, the minimum value of RMSE was found in SVR model with C = 210 and epsilon = 0.06 under the linear kernel function.

Results of LOOCV of SVR and Discussion

Based on seven descriptors selected in Sect. 919, Figs. 32 and 33 illustrate $[-logIC_{50}]$(Exp.) vs. the trained $[-logIC_{50}]$(Cal.) and predicted $[-logIC_{50}]$ (Pred.) of LOOCV in SVR models with C = 210 and epsilon = 0.06 under the linear kernel function, respectively.

From Fig. 32, the value of r (correlation coefficient) between $[-logIC_{50}]$(Exp.) and $[-logIC_{50}]$(Cal.) is 0.887, ARE (the average of relative error) is 0.0557 and AAE(the average of absolute error) is 0.287.

From Fig. 33, the value of r (correlation coefficient) between $[-logIC_{50}]$(Exp.) and $[-logIC_{50}]$(Pred.) is 0.866, ARE is 0.0622 and AAE is 0.319.

In this work, r and F (see Table 3) were obtained to estimate the performance of trained models [26]. R_c^2v (see Table 4) were obtained to estimate the performance of

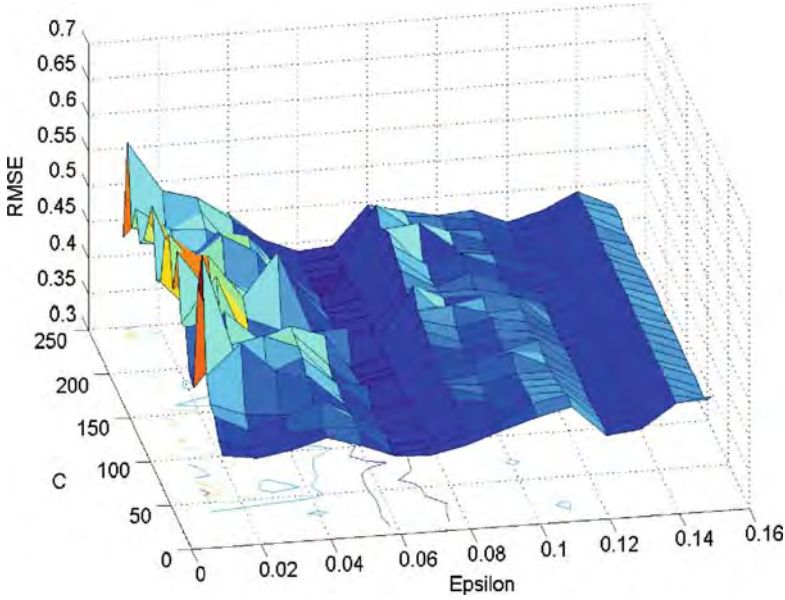


Fig. 31 RMSE in LOOCV vs. C and epsilon with linear kernel function

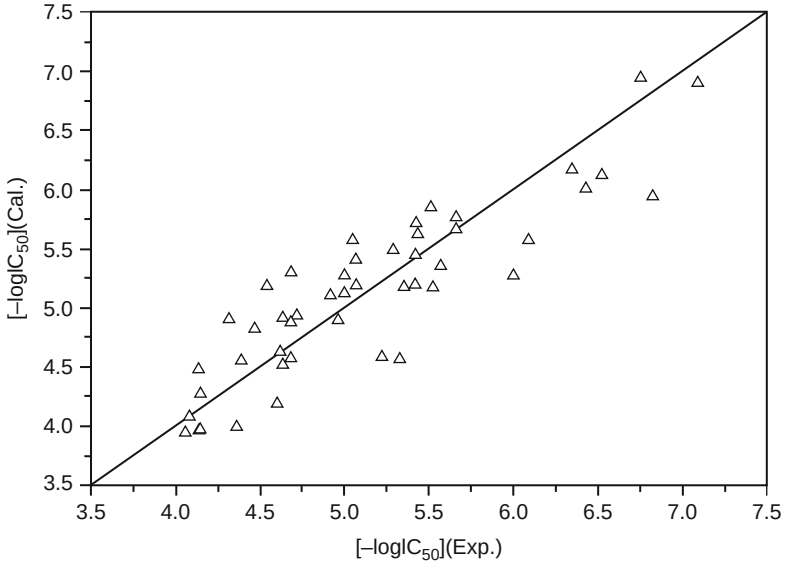


Fig. 32 $[-logIC_{50}]$(Exp.) vs. $[-logIC_{50}]$(Cal.) in SVR trained model using linear kernel function

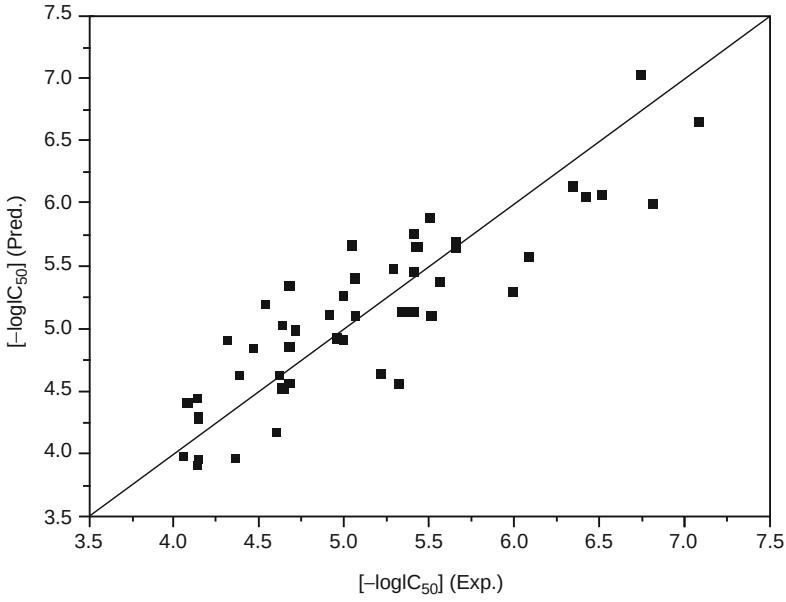


Fig. 33 $[-logIC_{50}]$(Exp.) vs. $[-logIC_{50}]$(Pred.) in LOOCV of SVR model using linear kernel

Table 3 r, ARE and AAE are obtained by trained model with seven descriptors

Model	PLS	CP−MLR	SVR
r	0.887	0.890	0.887
ARE	0.0560	0.0554	0.0557
AAE	0.288	0.284	0.287
F	21.08	21.77	21.08

Table 4 R^2_{cv}, ARE, AAE, and $RMSE$ obtained by LOOCV of SVR with seven parameters

Model	PLS	CP-MLR	SVR
R^2_{cv}	0.634	0.723	0.750
ARE	0.0699	0.0656	0.0622
AAE	0.366	0.338	0.319
$RMSE$	0.467	0.407	0.385

prediction models. Where F is the F-ratio between the variances of calculated and observed activities, $R^2_c v$ is cross-validated correlation coefficient from the LOOCV procedure, F and $R^2_c v$ are defined as follows:

$$F = \frac{r^2(n-k-1)}{k(1-r^2)}$$

$$R_{cv} = \sqrt{1 - \frac{\text{PRESS}}{\sum(y_{\text{act}} - y_{\text{mean}})^2}}$$

$$\text{PRESS} = \sum(y_{\text{pred}} - y_{act})^2$$

In equations above, n is the number of compounds, r the correlation coefficient, k the number of used parameters, y_{act} the actual value of y, y_{mean} the mean value of y, and y_{pred} is the value of prediction.

Based on Table 3, it can be found that the trained results of different models have no large differences. Based on Table 3, however, the value of R_c^2v in SVR model is the maximum among three models. It indicates that the prediction ability of SVR model was the best of all.

$r = 0.882$, $F = 20.10$, $R_c^2v = 0.691$ were reported in Yenamandra S. Prabhakar's best model [18], while $r = 0.887$, $F = 21.08$, $R_c^2v = 0.750$ were obtained in SVR model. Therefore, it can be concluded that SVR model with seven selected descriptors in this work has better predicting ability than Yenamandra S. Prabhakar's best model.

2.4 Conclusion

The SVM has been introduced as a robust and highly accurate data mining technique. The studies on qSAR of HEPT-analog compounds indicated that the predictive power of SVC outperforms that of the stepwise discriminant analysis. The studies on QSAR of Flavones proved that the prediction performance of SVR model were better than those of MLR or PLS models. Therefore, it can be concluded that the SVC is a good approach for the approximation of qualitative classification such as qSAR problems and that the SVR is a very promising tool for the approximation of quantitative regression such as QSAR problems, both with great potential in the research of drugs.

The training and optimization using SVM are easier and faster compared with other machine learning techniques, because there are fewer free parameters and only support vectors (only a fraction of all data) are used in the generalization process. Although no single method can be called uniformly superior to others, SVM is a very useful data-mining tool, which is especially suitable for finding relationships for a small size of data set. This allows generalising the conclusions drawn on the basis of such relationships. However, it should be mentioned that the success of SVM does not mean the fact that the classical data processing methods would be useless in SAR research. On the contrary, a clever strategy is to integrate SVM with other data processing methods together to make problem-solving for SAR.

3 Data-Mining-Based Industrial Optimization System for Chemical Process

Many chemical reaction systems in industrial production involve chemical reactions, heat transfer, mass transfer and fluid flow. The comprehensive modeling of these processes is very important for the industrial optimization. During the

last decades, process optimization and monitoring have been developed into an important branch of research and development in chemical engineering. The first-principle models have been used in the optimization of chemical processes [39–41]. However, those models are usually very difficult to obtain because of the complexity of the processes, and are also difficult to implement for on-line optimization since optimizations based on mechanistic models are usually very time-consuming.

In fact, large volumes of data in chemical process operation and control are collected in modern distributed control and automatic data logging systems. By using data mining, the useful information hidden in these data can be extracted not only for fault diagnosis but also for the optimal control with the objective of saving energy, increasing yield, reducing pollution, and decreasing production cost. However, compared with the more accurate laboratory data, the data from industrial processes have special characteristics: (1) the data usually have a higher noise/signal ratio. Even in a modern plants, there are uncontrollable factors affecting the production processes, such as the composition fluctuation of raw materials, shifts of production requirements, environmental fluctuation, operation accidents, influence of the unsteady state in the starting step and the transition from one operational mode to another mode; (2) the process data usually have a very nonuniform distribution, most of the data points are concentrated within the limited region based on the operation rules, and the data points outside of this region are thinly scattered; and (3) the features or variables of the data sets are usually not independent on each other, but more or less relevant to each other. Therefore, the data are usually in low quality for data mining. In recent years, linear data mining techniques, such as partial least squares (PLS) [42, 43], principal component analysis (PCA) [44, 45], and Fisher discriminant analysis (FDA) [46, 47], were frequently used in chemical process optimization and fault diagnosis. It has also been reported that nonlinear data mining techniques, such as artificial neural networks (ANN) [48–50] and genetic algorithms (GA) [51–53], can be applied to this field. And even the newly developed technology such as support vector machine (SVM) [54–56] method, which can be used for both linear and nonlinear cases, has been successfully employed in process industries. Despite the great progress of data mining methods, difficulties still exist in the method fusion, feature selection, automatic model, model validation, model updating, multi-model building, and on-line monitoring. Moreover, many newly proposed data mining technologies for processes optimization and monitoring either lack effective validation modules or depend on one validation algorithm, thus leading to a low identification reliability or sensitivity.

In recent years, we developed a Data Mining Optimization System (DMOS) for chemical process optimization and monitoring, which integrated database, pattern recognition, artificial intelligence, statistical learning, and domain knowledge. In Sect. 3.1, the principles and functions of the implemented methods are described briefly. Then both off-line and on-line DMOS software are introduced in Sect. 1.2. Finally, a case study of process optimization of ammonia synthesis is provided in Sect. 1.3.

3.1 Methodology

It is generally accepted that different data mining technologies are suitable for modeling different types of data sets, without any generalized method fitting for all cases. Therefore, both linear and nonlinear data mining technologies including K-nearest Neighbor (KNN), PLS, PCA, FDA, ANN, GA, SVM are implemented in the DMOS. Some novel techniques developed in our lab, such as hyper-polyhedron (HP) [57], are also implemented in the DMOS.

The frequently used data mining technologies can be referred to the corresponding references [58–66]. PLS is widely adopted for modeling linear relationships between multivariate measurements in chemometrics. The central idea of PCA is to reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much as possible of the variation present in the data set. FDA is a linear dimensionality reduction technique in terms of maximizing the separation between several classes. KNN is one of the best performers in a large number of classification problems. ANN is an adaptive system that can be used to model complex relationships between inputs and outputs or to find patterns in data. ANN has been attracting great interest in modeling nonlinear relationships. GA is an advanced optimization technique that mimics the selection processes occurring in nature. In the DMOS, a GA-based algorithm is used for searching optimal parameters in an n-dimension space. SVM has become an increasingly popular technique for machine learning activities including classification, regression, and outlier detection. As compared with other algorithms used in data mining, SVM exhibits an outstanding advantage of generalization in processing small data sets.

3.2 DMOS: A Software System Based on Data Mining for Chemical Process Optimization and Monitoring

Two kinds of tasks exist in complex chemical processes monitoring and optimization: (1) the off-line performance analysis and optimization based on long-term historical data, conducted by supervisors and engineers and (2) the online evaluation, optimization, and monitoring conducted by operators [67]. The supervisors and engineers monitor the long-range performance of a plant to identify opportunities for improvement and causes for poor operation, and a substantial amount of data involving a long term is usually used in the analysis. The operators require instant information to keep the process within acceptable working conditions. If unacceptable working performance appears, rapid response or intervention is needed to restore the unit back to acceptable working performance. Analysis of large volumes of multivariate data is overwhelming so a computer automated data analysis tool can convert this data to useful information.

To meet demands of these two types of tasks, the DMOS software has two operating modes: the off-line version and on-line version.

Description of the Off-line DMOS Version

The schematic diagram of the off-line DMOS version is given in Fig. 34. The three main components of the program are data warehouse, data mining algorithms, and custom-built system.

The component of data warehouse is used to save and update the process data from a distributed control system (DCS), related database, or process records. The component of data mining algorithms integrates all the data mining methods that can be used for data sampling, data evaluation, data structure analysis, features selection, correlation analysis, coefficient analysis, and model building, visualization analysis, and optimization scheme generating.

It should be mentioned that preprocessing data correctly is extremely important, because all subsequent analyses depend on the efficacy of the preprocessing procedure. Solid process knowledge and good engineering judgment are involved in most aspects of data preprocessing, such as process parameters selection, selection of the number of samples, and outlier point identification. The main goal of data mining is to obtain an accurate optimization model with good performance of prediction on production process.

The visualization analysis module provides an important graphical user interface to intuitively reveal the optimal results. The optimization scheme generating module reports all kinds of optimal results to engineers or operators according to given demands. The optimization model obtained by off-line DMOS software can be exported to the on-line DMOS software.

Figure 35 is a data processing flowchart of the off-line DMOS version. It has been proved to be a powerful information processing flow in many processes optimal cases [68].

Customized system provides a platform on which users can develop specialized on-line or off-line DMOS systems for their specific chemical process optimization unit. To accommodate different process optimization problems, we design different workflows adopting different data mining techniques, executing processes and result reports.

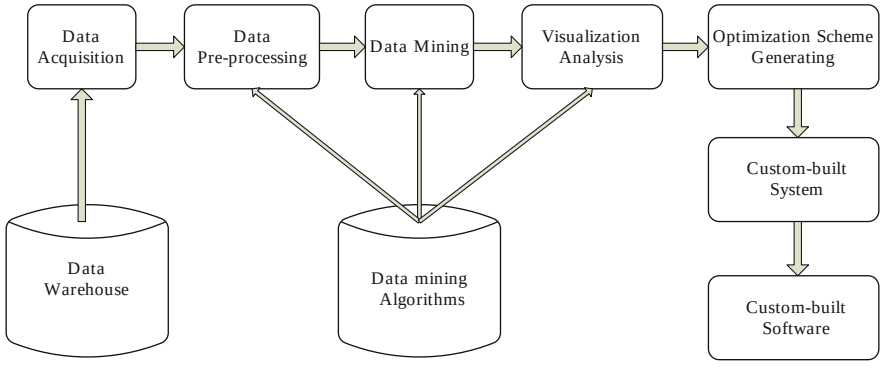


Fig. 34 The schematic of the off-line DMOS version

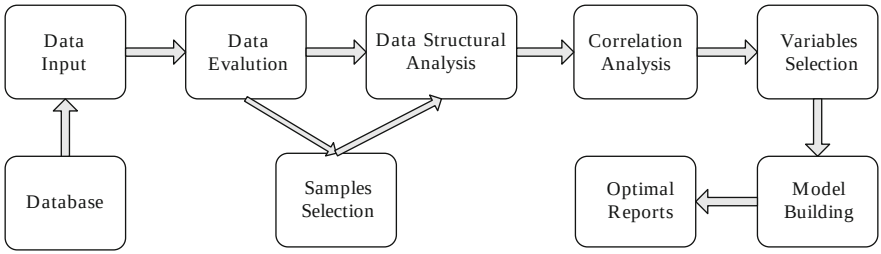


Fig. 35 Data processing flowchart of the off-line DMOS version

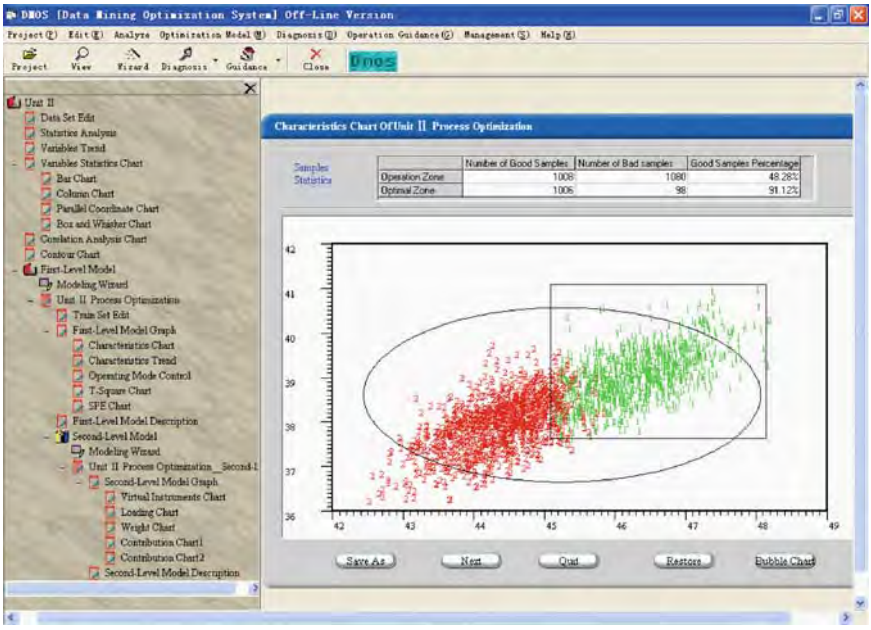


Fig. 36 The main operation window interface of off-line DMOS version

The graphical user interface (GUI) is programmed in MS Visual Basic, and can be installed and run on any PC with a window operating system. The main window of the GUI is given in Fig. 36. It consists of a menu bar, a project view window, and a visual zone. At the top of the window is the menu bar with eight items, which can be activated by means of the mouse pointer. The main menus cover the related operations such as project, editing, analysis, optimization model, diagnosis, operation guidance, management, and help. Correspondingly, the pop up sub-menus are available to perform some project related operations such as new project, open project, project view, data exporting, save graph, project exporting, model exporting, and exiting the project. Figure 37 shows the sub-window of the virtual instruments used for

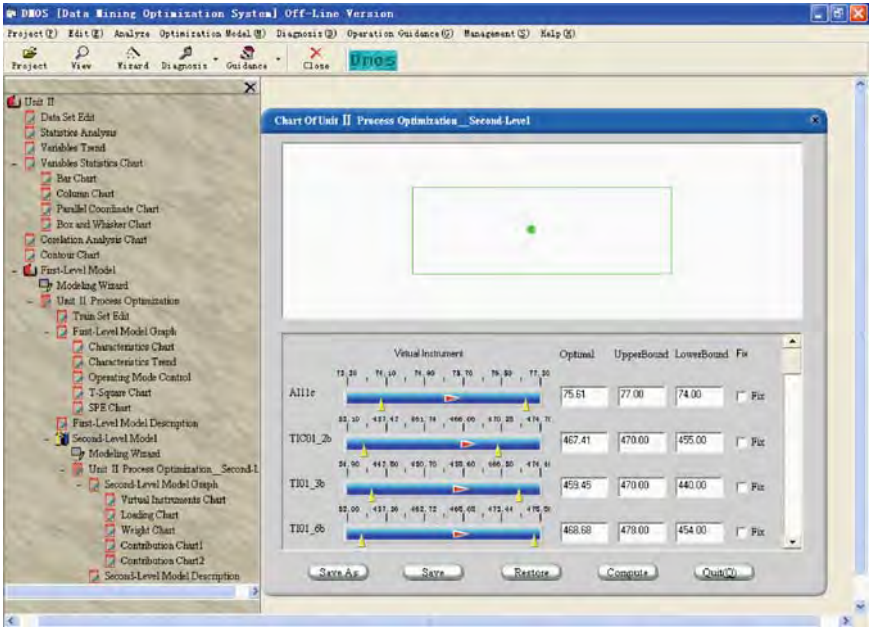


Fig. 37 The sub-window of virtual instruments for parameters adjustment of off-line DMOS version

some special parameters adjustment. Using this window, the operator can determine the direction and size of the special technical parameter adjustment.

Description of the on-line DMOS version

The on-line DMOS version consists of two different user interfaces. One is developed for the administrator and the other for the operator. The administrator interface offers all the required components for importing of the models, making of the rules and other general configuration tools. The operator interface has all the information required for on-line monitoring of the processes, which can be made transparent for the operators. Through this window it is possible to monitor the current state of a process and even to predict the future trend without concerning about transfer of data or other modifications. Figure 38 gives the sub-window of the virtual instruments used for some special parameters adjustment.

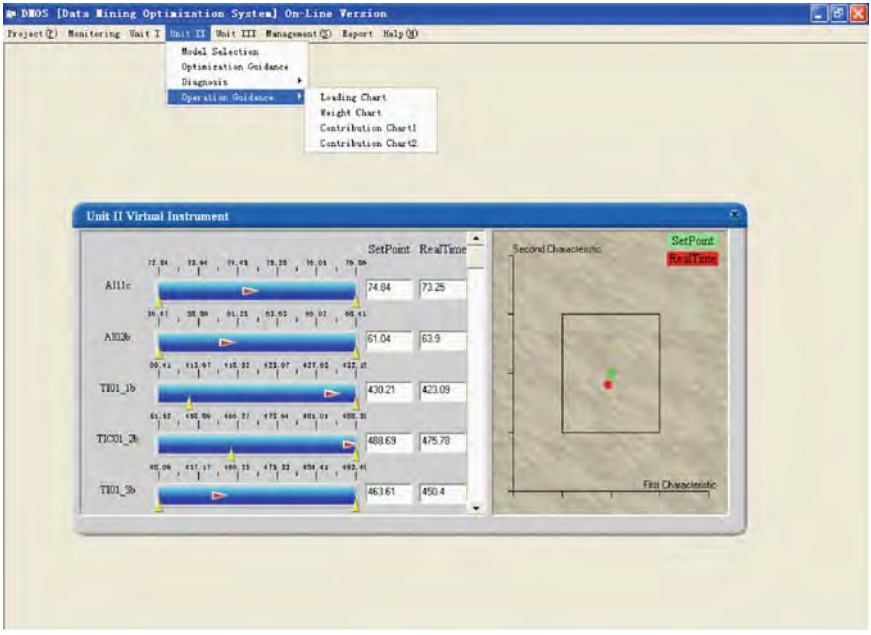


Fig. 38 The sub-window of virtual instruments for parameters adjustment of on-line DMOS version

Case Study of DMOS: Optimization of Ammonia Synthesis Process

The Haber–Bosch process for ammonia synthesis is considered to be one of the most significant scientific discoveries in the early twentieth century [69]. Ammoniais has been applied to various sectors, such as fertilizer productions, textile processing, water purification, explosive manufacturing, etc.; and the large scale production and application as fertilizer is by far the most important factor for the great increase in food production of the twentieth century. As a consequence of the increasing world population, there is no reason to suspect that the significance of NH_3 will become less important in the future.

In this work, the data set including 2088 samples each with 38 features and one target were collected from the DCS system of the ammonia synthesis unit in Yunnan Yunwei Group Co., Ltd. (China). The target of optimization is to increase the flow of fresh synthesis gas (FI, m^3/s). Feature selection techniques were used to search the chief factors determining the target.

It is generally accepted that different data mining methods are suitable for different data structure, and no generally applicable method for all cases. Therefore, before modeling we had better know the data structure, i.e., the geometrical or topological relationship between the optimal zone (the zone of good sample points) and the operation zone (the zone corresponding to the conditions of present operation). Here, we classify the data structures into two topological types: one-side type and

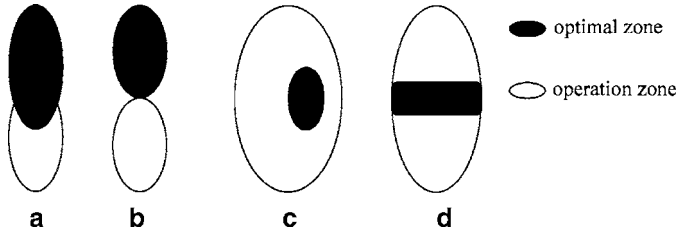


Fig. 39 Topological types of data structure: (**a**, **b**) one-side type and (**c**, **d**) inclusion type

Table 5 The target and the features used in optimization model

FI (m^3/s)	Flow of fresh synthesis gas	TIC02(K)	Temperature in the second-stage reactor
AI(%)	Hydrogen content of fresh synthesis gas	TIC03(K)	Temperature in the third-stage reactor
TI01(K)	Inlet temperature in the first-stage reactor	TI03(K)	Temperature in the third-stage reactor
TIC01(K)	Outlet temperature in the first-stage reactor	TI04(K)	Inlet temperature in the waste heat boiler
TI02(K)	Temperature in the second-stage reactor	TI05(K)	Outlet temperature in the water cooler

inclusion type. Figure 39 illustrates the characteristics of these two types. After the differentiation of different topological types, we can use different methods for the feature selection and the modeling of different topological data structures.

Preliminary statistical analysis indicated that the type of data set was one-side structure, so two methods were adopted for feature selection: one was based on a voting method (providing the results of PCA, PLS, and KNN methods), and the other was based on the hyper-polyhedron method. Fortunately, both methods gave the same result. As a result, nine features from 38 features were selected to model the flow of fresh synthesis gas: AI (%), TI01 (K), TIC01 (K), TI02 (K), TIC02 (K), TIC03 (K), TI03 (K), TI04 (K), and TI05 (K). The descriptions of the target and the nine features are listed in Table 5.

The process optimization results show that the optimal operation conditions of ammonia synthesis lie in a near linear optimization zone. Therefore, PCA method is applied to the optimization of ammonia synthesis unit.

The PCA model is built based on all process data. Figure 40 gives the PCA score plot to visualize the clusters in the data set. The distributed region of data cluster with class "1" (with target value more than $9.45\,m^3/s$) is the optimal zone, which can be described by following two inequalities in original feature space:

$$\begin{aligned} -452.002 < {} & 0.074(\mathrm{AI}) - 0.144(\mathrm{TIC01}) - 0.048(\mathrm{TI02}) - 0.119(\mathrm{TI03}) - 0.023(\mathrm{TI01}) \\ & -0.060(\mathrm{TIC02}) - 0.130(\mathrm{TIC03}) - 0.069(\mathrm{TI04}) - 0.096(\mathrm{TI05}) < -445.285 \end{aligned} \quad (1)$$

$$75.384 < -0.287(\mathrm{AI}) + 0.022(\mathrm{TIC01}) - 0.039(\mathrm{TI02}) - 0.022(\mathrm{TI03}) - 0.189(\mathrm{TI01})$$

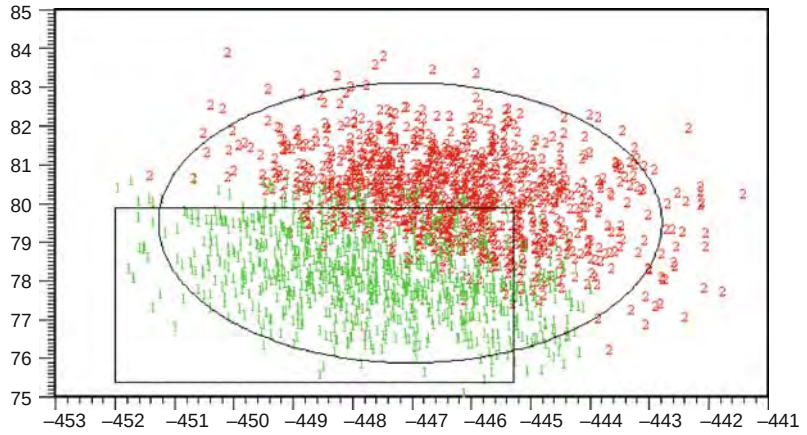


Fig. 40 Seven PCA score plot of ammonia synthesis process data

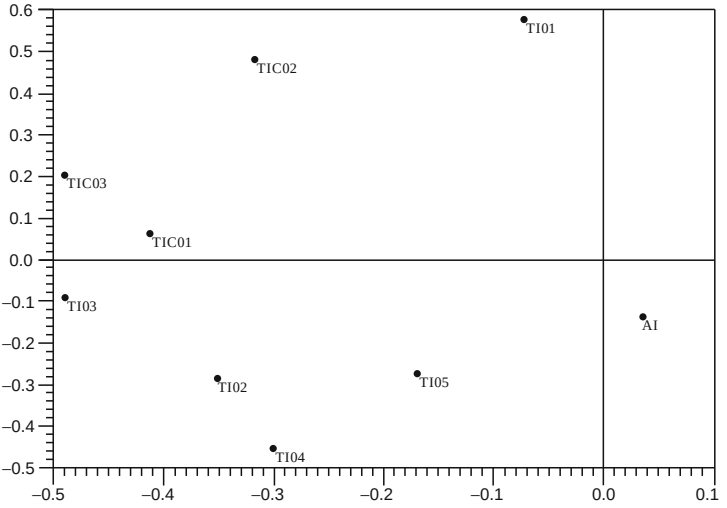


Fig. 41 Loading chart of PCA model

$$+0.091(\mathrm{TIC02}) + 0.054(\mathrm{TIC03}) - 0.104(\mathrm{TI04}) - 0.154(\mathrm{TI05}) < 79.890 \qquad (2)$$

By keeping the operation conditions within the optimal zone, the flow of fresh synthesis gas can be increased significantly.

Figure 41 is the variable loadings of all the nine features on the principal components. The most important technical parameters are TI01, TI03, and TI04 based on the principal component loadings and the scores. The higher inlet temperature in the waste heat boiler (TI04), the higher temperature in the third stage reactor (TI03), and

the lower inlet temperature in the first stage reactor will contribute to the increased flow of fresh synthesis gas.

The PCA model is validated by carrying out an experimental investigation and comparing the factual class of sample points of industrial process with the model-predicted class. A test set with 80 samples derived from continuous industrial process data that were excluded from the training set, was used to validate the PCA model. The results show that the accuracy of the prediction for the sample class was 95.00%. So, the model based PCA meets the requirement of ammonia synthesis process optimization. As a result, the model can be exported to on-line DMOS software, which can realize the on-line monitoring and optimization of ammonia synthesis process.

Discussions and Conclusions

Data-mining-based optimization and monitoring for chemical processes are a great challenge both for research and practice, because it may be dangerous to use a wrong mathematical model in practical operation of a chemical plant. The high noise/signal ratio data acquired from practical operations may sometimes lead to wrong conclusions in process modeling. The following rules should be obeyed in modeling for chemical processes: (1) it is necessary to use the knowledge of domain experts in data processing. The domain experts can judge whether the mathematical model is reasonable or not, or whether it is dangerous or not; (2) it is necessary to use as many as methods of data mining available and then make knowledge fusion based on the overall data mining results. The linear projection techniques such as FDA, PCA, and PLS methods are very useful, because linear projections can reveal the simple relationships between target and features. At the same time, nonlinear data mining techniques, such as ANN and GA, should also be used to increase the reliability of the mathematical models. The mathematical models should also be validated by some predefined schemes. For example, if a mathematical model not only fits well the data of training set, but also gives good prediction results by an independent validation test, this mathematical model should be considered as a robust one; (3) the industrial optimization task is to find an optimal operation zone in the high-dimensional space spanned by influencing parameters. It is not necessary to include all good sample points into the optimal operation zone for optimal control, but the optimal operation zone must be large enough to make the control practically feasible. To make optimal control more reliable, the optimal operation zone should separate as far as possible from the operation zone where bad sample points distributed. For fault diagnosis, it is also necessary to find a reliable operation zone to avoid the fault. Therefore, the optimal operation zone for industrial optimization or the safe operation zone to avoid fault should be selected from subspaces where good sample points are distributed and separated far away from the bad sample region; and (4) sometimes the geometrical form of the optimal operation zone will become too complicated for data fitting. In such cases, it is usually helpful to divide the hy-

perspace into subspaces, so that the geometry of optimal region can be simplified and modeling can be conducted.

In the present work, a comprehensive and graphical computer software, the data mining optimization system (DMOS), for chemical process optimization and monitoring has been developed. The DMOS can be used both for off-line and on-line purposes. The user-friendly interface and a great bank of data mining methods implemented in DMOS make the operation easy, convenient, and adaptable. The DMOS has some exciting characteristics such as method fusion, feature selection, automatic model, model validation, model updating, multi-model building, and on-line monitoring, which contribute to solve the optimization and monitoring problems of complex chemical processes. The successful application of the DMOS in ammonia synthesis process promises its great potential in chemical process optimization and monitoring.

References

1. P. Villars, J. Less-Common Met. **92**, 215 (1983)
2. N. Chen, R. Chen, W. Lu, C. Li, P. Qin, L. Yao, D. Zhu, in *Proceeding of the International Conference on MultiSource-Multisensor Information Fusion*, Las Vegas, USA, 1998
3. N. Chen, W. Lu, R. Chen, Regularities of formation of ternary intermetallic compounds part 1. Ternary intermetallic compounds between nontransition elements. J. Alloys Compd. **289**, 120 (1999)
4. E. GalvaodaSilva, J. Magn. Magn. Mater. **138**, 63 (1994)
5. J.I. Espaso, Ph. Galez, L. Paccard, C. Lester, D. Paccard, J. Allemand, J.C. Gomez Sal, K.A. McEwen, K.S. Knight, Solid State Commun. **92**, 389(1994)
6. N. Chen, W. Lu, P. Qin, Regularities of formation of ternary intermetallic compounds part 2. Ternary intermetallic compounds between transition elements, J. Alloys Compd., **289**, 126 (1999)
7. R.N. Lyon, **Liquid-Metals Handbook** (The Atomic Energy Commission, Washington, DC, 1952)
8. Ya.R. Bilyalov, M.M. Alexanderova, E.V. Tatyanin, J. Alloys Compd. **189**, L5 (1992)
9. S.A. Maklouf, T. Nakamura, M. Shiga, J. Magn. Magn. Mater. **135**, 257 (1994)
10. W. Lu, N. Chen, C. Li, Regularities of formation of ternary intermetallic compounds part 3. Ternary intermetallic compounds between one transition and two nontransition elements. J. Alloys Compd. **289**, 131 (1999)
11. G. Gordier, T. Friedrich, R. Hansleit, A. Grauel, U. Tegel, C. Schank, X. Geibel, J. Alloys Compd. **201**, 197 (1993)
12. G. Gordier, P. Woll, J. Less-Common Metals **169**, 291 (1991)
13. N. Chen, W. Lu, R. Chen, Regularities of formation of ternary intermetallic compounds part 4. Ternary intermetallic compounds between two transition and one nontransition elements. J. Alloys Compd. **292**, 129 (1999)
14. J. Hadamard. *Lectures on the Cauchy Problem in Linear Partial Differential Equations* (Yale University Press, Yale, 1923)
15. A.I. Belousov, S.A. Verzakov, J. von Frese, A flexible classification approach with optimal generalization performance: support vector machines. Chemom. Intell. Lab. Syst. **64**, 15 (2002)
16. V.N. Vapnik. **Statistical Learning Theory** (Wiley, New York, USA, 1998)
17. V. Wan, W.M. Campbell, Support vector machines for speaker verification and identification, in *Neural Networks for Signal Processing X: Proceedings of the 2000 IEEE Signal Processing Workshop*, vol. 2 (IEEE, 2000) p. 775

18. T. Joachims, Learning to classify text using support vector machines. Dissertation, Universitaet Dortmund, February 2001
19. T. Van Gestel, J.A.K. Suykens, D.E. Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, B. De Moor, J. Vandewalle, Financial time series prediction using least squares support vector machines within the evidence framework, IEEE Trans. Neural Networks **12**(4), 809 (2001)
20. M. Trotter, B. Buxton, S. Holden, Support vector machines in combinatorial chemistry. Meas. Control. **34**(8), 235 (2001)
21. Y.D. Cai, X.J. Liu, Y.X. Li, X.B. Xu, K.C. Chou, Prediction of beta-turns with learning machines. Peptides **24**(5), 665 (2003)
22. Y.D. Cai, K.Y. Feng, Y.X. Li, K.C. Chou, Support vector machine for predicting alpha-turn types. Peptides **24**(4), 629 (2003)
23. R. Burbidge, M. Trotter, B. Buxton, S. Holden, Drug design by machine learning: support vector machines for phamaceutical data analysis. Comput. Chem. **26**(5), 5 (2001)
24. N.Y. Chen, W.C. Lu, G.-Z. Li, J. Yang, *Support Vector Machine in Chemistry* (World Scientific Singapore, 2004)
25. S. Gunn, Support Vector machines for classification and regression, ISIS Technical Report, University of Southampton, 14 May 1998
26. Platt J. C., Fast training of Support Vector Machines using Sequential Minimal Optimization, Microsoft Research, http://www.research.microsoft.com/ jplatt
27. W. Lu, N. Chen, C. Ye, G. Li, Introduction to the algorithm of support vector machine and the software chemSVM. Comput. Appl. Chem. (in Chinese) **19**(6), 697 (2002)
28. N. Chen, W. Lu, C. Ye, G. Li, Application of support vector machine and kernel function in chemometrics. Comput. Appl. Chem. (in Chinese) **19**(6), 691 (2002)
29. S. Gayen, B. Debnath, S. Samanta, T. Jha, Bioorg. Med. Chem. **12**, 1493 (2004)
30. C.N. Alves, J. C. Pinheiro, A.J. Camargo, M.M.C. Ferreira, A.B.F. da Silva, J. Mol. Struct.-Theochem. **530**, 39 (2000)
31. HYPERCHEM Release 7.0 for Windows Molecular Modeling System, Hypercube Inc. (2002)
32. J.J.P. Stewart, J. Comput. Chem. **10**, 209 (1989)
33. N.Y. Chen, P. Qin, R.L. Chen, W.C. Lu, *Pattern Recognition Applied to Chemistry and Chemical Industry* (Science Press, Beijing, 1999, in Chinese)
34. P.Z. Zimmet, K.G.M.M. Alberti, J. Shaw, Global and societal implications of the diabetes epidemic. Nature **414**, 782 (2001)
35. H. Terashima, K. Hama, R. Yamamoto, M. Tsuboshima, R. Kikkawa, I. Hatanaka, Y. Shigeta, Effect of new aldose reductase inhibitors on various tissues in vitro. J. Pharmacol. Exp. Therap. **229**, 226 (1984)
36. S.S.M. Chung, E.S.M. Ho, K.S.L. Lam, S.K. Chung, Contribution of polyol pathway to diabetes-induced oxidative stress. J. Am. Soc. Nephrol. **14**, S233 (2003)
37. H. Matsuda, T. Morikawa, I. Toguchida, M. Yoshikawa, Structural requirement of the flavonoids and related compound for aldose reductase inhibitory activity. Chem. Pharm. Bull. **50**, 788 (2002)
38. Y.S. Prabhakar, M.K. Gupta, N. Roy, Y. Venkateswarlu, A high dimensional QSAR study on the aldose reductase inhibitory activity of some flavones: topological descriptors in modeling the activity. J. Chem. Inform. Model. **46**(1), 86 (2006)
39. P. Dufour, D.J. Michaud, Y. Toure , P.S. Dhurjati, A partial differential equation model predictive control strategy: application to autoclave composite processing. Comput. Chem. Eng. **28**, 545 (2004)
40. A. Toumi, S. Engell, M. Diehl, H.G. Bock, J. Schloder, Efficient optimization of simulated moving bed processes. Chem. Eng. Process. **46**, 1067 (2007)
41. G. Maria, Model-based heuristic optimized operating policies for D-glucose oxidation in a batch reactor with pulsate addition of enzyme. Comput. Chem. Eng. **31**, 1231 (2007)
42. P. Nomikos, J.F. MacGregor, Multi-way partial least squares in monitoring batch processes. Chemometr. Intell. Lab. **30**, 97 (1995)
43. J. Flores-Cerrillo, J.F. MacGregor, Multivariate monitoring of batch processed using batch-to-batch information. AIChE J. **50**, 1219 (2004)

44. P. Nomikos, J.F. MacGregor, Monitoring batch processes using multiway principal component analysis. AIChE J. **40**, 1361 (1994)
45. B.M. Wise, N.B. Gallagher, The process chemometrics approach to process monitoring and fault detection. J. Proc. Cont. **6**, 329 (1996)
46. L.H. Chiang, E.L. Russell, R.D. Braatz, Fault diagnosis in chemical processes using Fisher discriminant analysis, discriminant partial least squares, and principal component analysis, Chemometr. Intell. Lab. **50**, 243 (2000)
47. Q.P. He, S.J. Qin, J. Wang, A new fault diagnosis method using fault directions in fisher discriminant analysis. AIChE J. **51**, 555 (2005)
48. N. Bhat, T.J. McAvoy, Use of neural nets for dynamic modeling and control of chemical process systems. Comput. Chem. Eng. **14**, 573 (1990)
49. C.A.O. Nascimento, R. Giudici, R. Guardani, Neural network based approach for optimization of industrial chemical processes. Comput. Chem. Eng. **24**, 2303 (2000)
50. A. Chouai, M. Cabassud, M.V. Le Lann, C. Gourdon, G. Casamatta, Use of neural networks for liquid–liquid extraction column modeling: an experimental study. Chem. Eng. Process. **39**, 171 (2000)
51. I.P. Androulakis, V. Venkatasubramanian, A genetic algorithm framework for process design and optimization. Comput. Chem. Eng. **15**, 217 (1991)
52. K.H. Low, E. Sorensen, Simultaneous optimal design and operation of multipurpose batch distillation columns. Chem. Eng. Process. **43**, 273 (2004)
53. B. McKay, M. Willis, G. Barton, Steady-state modeling of chemical process systems using genetic programming. Comput. Chem. Eng. **21**, 981 (1997)
54. L.H. Chiang, M.E. Kotanchek, A.K. Kordon, Fault diagnosis based on Fisher discriminant analysis and support vector machines. Comput. Chem. Eng. **28**, 1389 (2004)
55. A. Kulkarni, V.K. Jayaraman, B.D. Kulkarni, Support vector classification with parameter tuning assisted by agent-based technique. Comput. Chem. Eng. **28**, 311 (2004)
56. A.M. Jade, V.K. Jayaraman, B.D. Kulkarni, A.R. Khopkar, V.V. Ranade, A. Sharma, A novel local singularity distribution based method for flow regime identification: gas–liquid stirred vessel with Rushton turbine. Chem. Eng. Sci. **61**, 688 (2006)
57. X.H. Bao, W.C. Lu, L. Liu, N.Y. Chen, Hyper-polyhedron model applied to molecular screening of guanidines as Na/H exchange inhibitors. Acta Pharmacol. Sin. **24**, 472 (2003)
58. K. Pearson, On lines and planes of closest fit to systems of points in space. Philos. Mag. **2**, 559 (1901)
59. R.O. Duda, P.E. Hart, D.G. Stork, *Pattern Classification* (Wiley, New York, 2001)
60. T.M. Cover, P.E. Hart, Nearest neighbor pattern classification. IEEE Trans. Inf. Theory, **IT-13**, 21 (1967)
61. A.F. Murray, *Application of Neural Networks* (Kluwer, Boston, 1995)
62. D.E. Goldberg, *Genetic Algorithms In Search, Optimization, And Machine Learning* (Addison-Wesley, Reading, MA, 1989)
63. V. Vapnik, *The Nature Of Statistical Learning Theory* (Springer, New York, 1995)
64. X.B. Ji, W.C. Lu, L. Liu, N.Y. Chen, Using sphere-linear map for the prediction of classes of antifungal activities of triazole-derivatives. Comput. Appl. Chem. **22**, 449 (2005)
65. P. Geladi1, B.R. Kowalski, Partial least-squares regression: a tutorial. Anal. Chim. Acta. **185**, 1 (1986)
66. W.C. Lu, X.H. Bao, L. Liu, J. Kong, L.C. Yan, N.Y. Chen, Hierarchical projection method for study of formation of binary complex compounds in MBr–M'Br2 system. Comput. Appl. Chem. **19**, 473 (2002)
67. X.Z. Wang, C. McGreavy, Automatic classification for mining process operational data. Ind. Eng. Chem. Res. **37**, 2215 (1998)
68. N.Y. Chen, W.C. Lu, R.L. Chen, C.H. Li, P. Qin, Chemometric methods applied to industrial optimization and materials optimal design. Chemometr. Intell. Lab. **45**, 329 (1999)
69. V. Smil, Detonator of the population explosion, Nature **400**, 415 (1999)

“This page left intentionally blank.”

Data Mining and Discovery of Astronomical Knowledge

Ghazi Al-Naymat

1 Introduction

Spatial data is essentially different from transactional data in its nature. The objects in a spatial database are distinguished by a spatial (location) and several non-spatial (aspatial) attributes. For example, an astronomy database that contains galaxy data may contain the x, y and z coordinates (spatial features) of each galaxy, their types and other attributes.

Spatial datasets often describe *geo-spatial* or *astro-spatial* (astronomy related) data. In this work, we use a large astronomical dataset containing the location of different *types* of galaxies. Datasets of this nature provide opportunities and challenges for the use of data mining techniques to generate interesting patterns. One such pattern is the *co-location* pattern. A co-location pattern is a group of objects (such as galaxies) each of which is located in the neighborhood (within a given distance) of another object in the group.

A *clique* is a special type of co-location pattern. It is described as a group of objects such that *all* objects in that group are co-located with each other. In other words, given a predefined distance, if a group of objects lie within this distance from every other object in the group, they form a clique. Figure 1 shows eight different objects $\{A1, A2, A3, B1, B2, B3, B4, C1\}$. The set $\{B1, B2, A3\}$ is a clique. However, $\{B1, B2, A3, C1\}$ is not, because $C1$ is not co-located with $B2$ and $A3$, therefore $\{B1, B2, A3, C1\}$ is a co-location pattern only.

In this chapter, we consider *maximal cliques*. A maximal clique is a clique that does not appear as a subset of another clique in the same co-location pattern (and therefore the entire dataset, as each object is unique). For example, in Fig. 1, $\{A1, A2, B4\}$ forms a maximal clique as it is not a subset of any other clique. However, $\{A3, B2, B3\}$ is not a maximal clique since it is a subset of the clique $\{A3, B1, B2, B3\}$ (which in turn *is* a maximal clique). The second column of Table 1 shows all the maximal cliques in Fig. 1.

G. Al-Naymat (✉)
School of Information Technologies, The University of Sydney, Sydney, Australia
e-mail: ghazi@it.usyd.edu.au

M.M. Gaber (ed.), *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*, DOI 10.1007/978-3-642-02788-8_12,

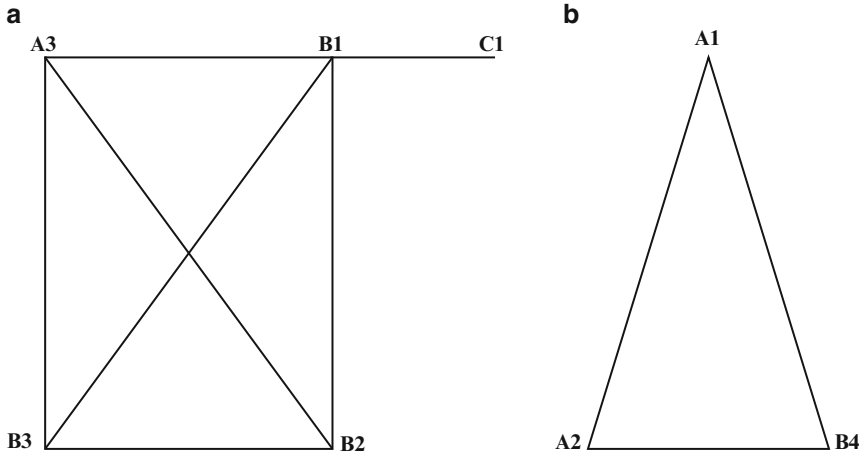


Fig. 1 An example of clique patterns

Table 1 Representing maximal cliques of Fig. 1 as complex relationships

ID	Maximal cliques	Raw maximal cliques	Non-complex relationships	Complex without negative relationships	Complex with negative relationships
1	{A3, B1, B2, B3}	{A, B, B, B}	{A, B}	{A, B, B+}	{A, B, B+, −C}
2	{B1, C1}	{B, C}	{B, C}	{B, C}	{−A, B, C}
3	{A1, A2, B4}	{A, A, B}	{A, B}	{A, A+, B}	{A, A+, B, −C}

In our dataset, each row corresponds to an object (galaxy) and contains its type as well as its location. We are interested in mining relationships between the *types* of objects. Examples of object types in this dataset are "early-type" galaxies and "late-type" galaxies. To clarify, we are not interested in the co-locations of *specific* objects, but rather, the co-locations of their *types*. Finding complex relationships between such types is useful information in the astronomy domain – this will be shown in Sect. 1.2. In Fig. 1, there are three types: $\{A, B, C\}$.

In this chapter, we focus on using maximal cliques to allow us to mine interesting *complex spatial relationships* between the object types. A *complex spatial relationship* includes not only whether an object type, say A, is present in a (maximal) clique, but:

- Whether *more than one* object of its type is present in the (maximal) clique. This is a *positive type* and is denoted by $A+$.
- Whether objects of a particular type are not present in a *maximal clique*; i.e., the absence of types. This is a *negative type* and is denoted by $-A$.

The inclusion of *positive* and/or *negative types* makes a relationship *complex*. This allows us to mine patterns that state, for example, that A occurs with multiple Bs but not with a C. That is, the presence of A may imply the presence of

multiple Bs and the absence of C. The last two columns of Table 1 show examples of (maximal) complex relationships. We propose an efficient algorithm (GridClique) to extract all maximal clique patterns from a large spatial data – the Sloan Digital Sky Survey (SDSS) data.

We are not interested in *maximal* complex patterns (relationships) in themselves, as they provide only local information (i.e., about a maximal clique). We are however interested in *sets* of object types (including complex types) that appear across the entire dataset (i.e., among many maximal cliques). In other words, we are interested in mining *interesting complex spatial relationships* (sets), where "interesting" is defined by a global measure. We use a variation of the $minPI$ (minimum Participation Index) measure [1] to define interestingness (1)

$$minPI(P) = \min_{t \in P}\{N(P)/N(\{t\})\}. \tag{1}$$

Here, P is a set of complex types we are evaluating, and $N(\cdot)$ is the number of maximal cliques that contain the set of complex types. We count the occurrences of the pattern (set of complex types) only in the maximal cliques. This means that if the $minPI$ of a pattern is above α, then we can say that whenever any type $t \in P$ occurs in a maximal clique, the entire pattern P will occur at least as a fraction *alpha* of those maximal cliques. Using $minPI$ is superior to simply using $N(P)$ because it scales by the occurrences of the individual object types, thus reducing the impact of a non-uniform distribution on the object types.

In this work we focus on *maximal cliques* because:

- The process of forming complex positive relationships makes sense. Suppose we extract a clique that is not maximal, such as $\{A1, B4\}$ from Fig. 1. We would not generate the positive relationship $\{A+, B\}$ from this, even though each of $\{A1, B4\}$ are co-located with $\{A2\}$. So we get the correct pattern only once we have considered the maximal cliques.
- Negative relationships are possible. For example, consider the maximal clique in row 1 of Table 1. If we did not use *maximal* cliques, then we would consider $\{B1, B2, B3\}$, and from this we would *incorrectly* infer that the complex relationship $\{B, B+, -A\}$ exists. However, this is not true because A *is* co-located with each of $\{B1, B2, B3\}$. *Therefore, using non-maximal cliques will generate* incorrect *negative patterns.*
- Each maximal clique will be considered as a single instance (transaction) for the purposes of counting. In other words, we automatically avoid multiply counting the same objects within a maximal clique.
- Mining maximal cliques reduces the number of cliques by removing all redundancy. It is possible to mine for maximal cliques directly. And because negative types cannot be inferred until the maximal clique is mined, it does not make sense to mine cliques that are not maximal.

The previous reasons demonstrate the value of our proposed algorithm GridClique given it mines all maximal clique patterns from large spatial data and puts them into a format which can be mined easily using association rule mining techniques.

1.1 Problem Statement

The problem that we are considering in this chapter consists of two parts:

1. Given a large spatial dataset (astronomy dataset), extract all maximal clique patterns. More specifically, extract all co-location patterns that are not subsets of any other co-location patterns.
2. Given the set of maximal cliques, find all interesting and complex patterns that occur among the set of maximal cliques. More specifically, find all *sets* of object types, including *positive* and *negative* (i.e., complex) types that are interesting as defined by their $minPI$ being above a threshold.

To solve the above problem efficiently, we propose a heuristic based on a divide-and-conquer strategy. Our proposed heuristic is GridClique [2] as it will be described in Sect. 3.2. After obtaining the maximal clique patterns the problem, therefore, becomes an *itemset mining* task. To achieve this very quickly, we use the GLIMIT algorithm [3].

Including negative types makes the problem much more difficult, as it is typical for spatial data to be sparse. This means that the absence of a type can be very common. Approaches relying on an Apriori style algorithm find this very difficult; however, this is not a problem for our approach.

1.2 Contributions

In this chapter, we make the following contributions:

1. We introduce the concept of *maximal cliques*. We describe how the use of *maximal cliques* makes more sense than simply using cliques, and we show that they allow the use of negative patterns.
2. We propose a heuristic GridClique on the basis of a divide-and-conquer strategy, to mine maximal clique patterns that will be used in mining complex co-location rules (MCCRs).
3. We propose a general procedure that splits the maximal clique generation, complex pattern extraction and interesting pattern mining tasks into modular components.
4. We contribute to the astronomy domain by proving existing facts when analyzing the complex association rules that are generated by our proposed approach. For example, we managed to answer questions such as the one depicted in Fig. 2.

A preliminary version of this work was presented in [2]. However, in this chapter, we used GridClique to do more analysis using the new release of SDSS Data (DR6).[1] We described the data preparation stage in detail. A performance

[1] http://www.sdss.org/dr6.

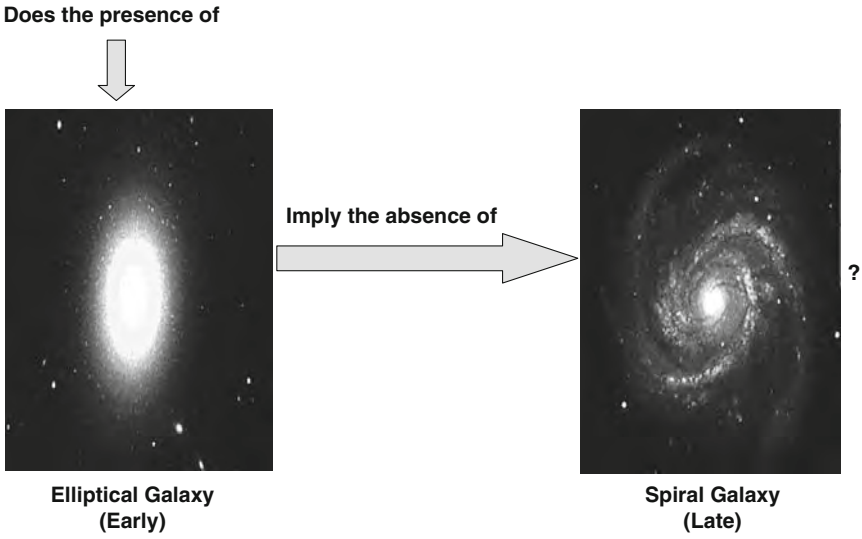


Fig. 2 An interesting question which can be answered by our method

comparison between GridClique and Naïve approach is also presented here to show the efficiency of the GridClique algorithm. GridClique applied to gain more knowledge than what presented in the preliminary version such as reporting very interesting rules that reflect the relationship between the defined galaxy types. Example of some interesting rules was shown and introduced to the astronomy domain. In this study, we show the complete mining process by adding the data preparation stage to the proposed framework. Table 2 provides a description for all used notations.

1.3 Chapter Outline

This chapter is organized as follows. Section 2 illustrates the important concepts and notations used in this work. In Sect. 3, we give detailed description about our approach for extracting knowledge from astronomical dataset. Section 4 contains our experiments and the analysis of the results. A summary of the chapter is given in Sect. 5.

2 Basic Definitions and Concepts

Consider a set of objects O with fixed locations. Given an appropriate distance measure $d : O \times O \rightarrow R$ we can define a graph G as follows; let O be the vertices and construct an edge between two objects $o_1 \in O$ and $o_2 \in O$ if $d(o_1, o_2) \leq \tau$, where τ is a chosen distance. A *co-location pattern* is a connected subgraph.

Table 2 Description of the notations used

Symbol	Description
MCCR	Mining complex co-location rule
parsec	Unit of length used in astronomy. It stands for "**par**allax of one arc **sec**ond"
Mpc	An abbreviation of "megaparsec," which is one million parsecs, or 3,261,564 light years
arcmin	Unit of angular measurement. Sizes of objects on the sky, field of view of telescopes, or practically any angular distance "Arc of Minutes"
minPI	Minimum participation index
maxPI	Maximum participation ratio
SDSS	Sloan digital sky survey
z	RedShift
zWarning	Parameter used to guarantee that the corrected RedShift values are used
zConf	RedShift confidence
X	X-coordinate
Y	Y-coordinate
Z	Z-coordinate
U	Ultraviolet
R	Red light magnitude
r-band	r-Band Petrosian magnitude
H_o	Hubble's constant
LRG	Luminous Red Galaxies
O	Spatial objects with fixed locations
o_i	The ith spatial object
G	Undirected graph
C_M	Maximal clique
P	A set of complex types
$N(\cdot)$	Number of maximal cliques that contain the set of complex types
T	Set of transactions
$Card(\cdot)$	Cardinality of a set
$\lceil \cdot \rceil$	Function that gives the ceiling of a value
$\lfloor \cdot \rfloor$	Function that gives the floor of a value
EucDist	Euclidean distance

Definition 1 (Clique). A clique $C \in O$ is any fully connected subgraph of G. That is, $d(o_1, o_2) \leq \tau \ \forall \{o_1, o_2\} \in C \times C$.

As we have mentioned in Sect. 1, we use maximal cliques so that we can define and use complex patterns meaningfully and to avoid double counting.

Definition 2 (Maximal Clique). A maximal clique C_M is a clique that is not a subset (sub-graph) of any other clique. So that $C_M \not\subset C \ \forall C \in O$.

Definition 3 (Clique Cardinality). A cardinality (*Card*) is the size of a clique and it is given in (2). For example, if we have a clique $C = \{o_1, o_2, o_3\}$, then $Card(C) = 3$.

$$Card(C) = |\{o \in O : o \in C\}|, \tag{2}$$

where $|\cdot|$ denotes the number of elements in a set.

Generally, cardinality of a set is a measure of the "number of elements of the set."

3 Mining Complex Co-location Rules

The MCCRs process is illustrated through the flowchart in Fig. 3. It consists of four stages. First, the data preparation process, which starts by downloading the spatial data from the SDSS repository and then putting it in the desired format. Section 3.1 provides details about this stage. Second, the GridClique algorithm finds all *maximal clique patterns*, and strips them of the object identifiers – producing raw maximal cliques (Table 1). The GridClique algorithm is described in Sect. 3.2. One pass is then made over the raw maximal cliques in order to extract complex relationships. We describe this in Sect. 3.3. This produces maximal complex cliques, each of which is then considered as a *transaction*. An *interesting itemset mining algorithm*, using *minPI* as the interestingness measure, is used to extract the interesting complex relationships. We describe this in Sect. 3.4.

Figure 3 shows that the clique generation and complex relationship extraction are local procedures, in the sense that they deal only with individual maximal cliques. In contrast, the interesting pattern mining is global – it finds patterns that occur across the entire space. We consider subsets of maximal cliques only in the last step – after the complex patterns have been extracted.

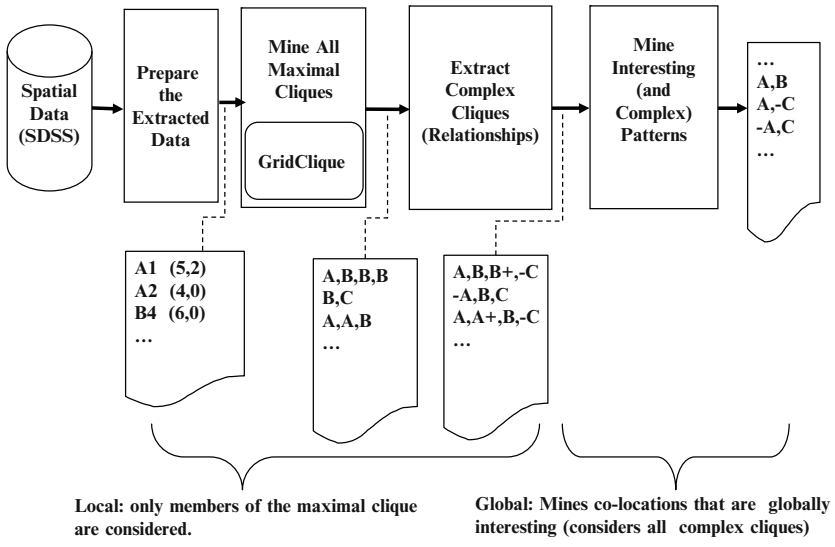


Fig. 3 The complete process of mining complex co-location rules (MCCRs)

3.1 Data Preparation

3.1.1 Data Extraction

This section describes the method for extracting attributes from the SDSS Data Release 6 and using them to categorize galaxy objects. A view called *SpecPhoto* which is derived from a table called *SpecPhotoAll* is used. The latter is a join between the *PhotoObjAll* and *SpecObjAll* tables. In other words, *SpecPhoto* is view of joined *Spectro* and *PhotoObjects* which contains only the clean spectra.

The concern is to extract only the galaxy objects from the SDSS using the parameter (object type = 0). The total number of galaxy type objects stored in the SDSS catalog is 507,594. To ensure the accuracy for calculating the distance between objects and the earth, which leads to calculate the X, Y, and Z coordinates for each galaxy, number of parameters are used, such as $zConf < 0.95$ (the rigid objects) and $zWarning = 0$ (correct RedShift). Therefore, the number of galaxy objects is reduced to 365,425.

SDSS release DR6 provides a table called *Neighbors*. This table contains all objects that are located within 0.5 arcmin, this makes it not useful in this research because there is no ability to choose any other distance greater than 0.5 arcmin to form the neighborhood relationship between objects. For example, in our experiments $(1, \ldots, 5)$ Mpc (distances) are used as the thresholds to check if objects are co-located or not.[2] Table 3 lists the extracted fields from the SDSS (DR6) that have been used during the preparation process.

The raw data was obtained from SDSS (DR6) [4]. This data is extracted from the online catalog services using several SQL statements. The catalog offers a very elegant interface that allows users to extract easily the preferred

Table 3 The SDSS schema used in this work

No	Field name	Field description
1.	specObjID	Unique ID
2.	z	Final RedShift
3.	ra	Right ascension
4.	dec	Declination
5.	cx	x of Normal unit vector
6.	cy	y of Normal unit vector
7.	cz	z of Normal unit vector
8.	primTarget	prime target categories
9.	objType	object type : Galaxy = 0
10.	modelMag_u	Ultraviolet magnitude
11.	modelMag_r	Red Light magnitude

[2] See http://csep10.phys.utk.edu/astr162/lect/distances/distscales.html for details.

Fig. 4 The front end of the SDSS SkyServer. It provides an SQL search facilities to the entire stored data

data (Fig. 4). The catalog provides other tools that can be used to browse all tables and views in the SDSS data. This catalog is accessible from the SDSS website.[3]

The first SQL statement used to extract the first 65,000 objects (galaxies) is as follows:

```
Select top 65000 specObjID, cx, cy,
                 cz, primTarget, z,
                 dbo.fObjTypeN(objType) as objType,
                 (299792.485 * z/71) as Mpc,
                 modelMag_u - modelMag_r as 'U-R'
                 modelMag_r
        From specPhoto
Where z>0 and zConf>0.95 and zWarning=0
      and objtype =0
Order by specObjID;
```

[3] http://cas.sdss.org/dr6/en/tools/search/sql.asp.

The second SQL statement used to extract the second 65,000 objects starting from the last object's ID (119286486972497000) is as follows:

```
Select top 65000 specObjID, cx, cy,
                 cz, primTarget, z,
                 dbo.fObjTypeN(objType) as objType,
                 (299792.485 * z/71) as Mpc,
                 modelMag_u - modelMag_r as 'U-R'
                 modelMag_r
        From specPhoto
Where z>0 and zConf>0.95 and zWarning=0
      and objtype =0
      and specObjID>119286486972497000
Order by specObjID;
```

The second SQL statement is different from the first one by adding the a condition $specObjID > \{lastID\}$. The reason behind extracting just 65,000 objects is to be able to handle them using Microsoft Excel 2003, which was used to clean some records.

Data Transformation

The extracted data needs to be transformed into the right format before start mining it. Transforming the data makes it accessible using $Oracle10_g$, where we uploaded the data into a normalized database. The use of the Oracle helps us in (1) manipulating the extracted data and report some statistics about it, (2) eliminating the undesired fields that we had to extract when we initially downloaded the data from the SDSS repository, and (3) calculating the distance between galaxy objects and the earth. Few tables were created to store the extracted data. We created number of stored procedures to categorize galaxy objects and put the data into the right format.

New Attributes Creation

Having all necessary fields extracted, the next step is to calculate for each galaxy the exact value of the X, Y, and Z coordinates that are not stored in the SDSS data. The following steps show the process of calculating the coordinates of each galaxy:

1. Calculating the distance between the earth and galaxy objects using Hubble's law and redshift z value (3).

$$D \approx \frac{c \times z}{H_o}, \tag{3}$$

   where c is the speed of light, z is the object RedShift, and H_o is the Hubble's constant. Currently, the best estimate for this constant is $71\,\mathrm{km\,s^{-1}\,Mpc^{-1}}$ [5,6].

2. Considering the extracted unit vectors cx, cy, and cz and multiplying them by D that obtained form the previous step. Equations (4), (5) and (6) used to calculate the final value of X, Y and Z coordinates, respectively.

$$X = D \times cx. \tag{4}$$

$$Y = D \times cy. \tag{5}$$

$$Z = D \times cz. \tag{6}$$

Galaxies Categorization

Our purpose is to mine complex co-location rules in astronomy data. Therefore, we have to add to our prepared data the galaxy types. More specifically, to use the association rule mining technique on the prepared data the galaxy type is used. If we go back to the supermarket example $\{bread \rightarrow cheese\}$ rule, instead we use, for example, $\{Galaxy\ type\ A \rightarrow Galaxy\ type\ B\}$. This rule can be interpreted as the presence of galaxy type A implies the presence of galaxy type B.

Galaxy types did not exist in the SDSS data. Therefore, we used several parameters to find the galaxy object types. This was performed as follows:

1. The difference between Ultraviolet U and Red light magnitude R, is used to categorize galaxy objects into either "Early" or "Late." If the difference $\geq$2.22 the galaxy is considered to be "Early," otherwise "Late."
2. The value of the r-band *Petrosian* magnitude indicates whether a galaxy is "Main" (close to the earth) or "Luminous Red Galaxies" LRG (far away from the earth). If *r-band* $\leq$17.77 the galaxy object is "Main," otherwise it is LRG [7].

Consequently, four galaxy types were found as a combination of the above mentioned types. These categories are: *Main-Late*, *Main-Early*, *LRG-Late*, and *LRG-Early*.

3.2 Mining Maximal Cliques

The stage of mining maximal cliques is the process of transforming the raw spatial data into transactional type data that can be used by any association rule mining techniques. This is a crucial step and it is performed using the our proposed algorithm (GridClique).

GridClique Algorithm

The GridClique algorithm uses a heuristic based on a divide-and-conquer strategy to efficiently extract maximal clique patterns from large spatial dataset (SDSS). This is

Table 4 An example of two-dimensional dataset

Object type	X-coordinate	Y-coordinate
A1	2.5	4.5
A2	6	4
A3	2	9
B1	1.5	3.5
B2	5	3
B3	5	4
C1	2.5	3
C2	6	3
D1	3	9
D2	7	1.5

achieved by dividing the space into a grid structure based on a predefined distance. The use of the grid structure plays a vital role for reducing the search space. Our heuristic treats the spatial objects (galaxies) as points in a plane and it uses grid structure when mining the maximal clique patterns. We use the Euclidean distance as the distance measure because it is very efficient to compute.

The aim of the GridClique algorithm is to extract maximal clique patterns that exist in any undirected graph. It is developed using an index structure through grid implementation. Table 4 contains 10 objects and their X and Y coordinates; this information will be used to explain the functionality of the GridClique algorithm. The SDSS is a three-dimensional dataset, but in our example we use two-dimensional dataset for simplicity. Algorithm 1 displays the pseudocode of the GridClique algorithm.

The 10 spatial objects in Table 4 are depicted in Fig. 5. The figure will be used as an example when we explain our algorithm. The objects are used as spatial points and are placed in the plane using their coordinates. The edges in each subgraph are generated between every two objects on the basis of the co-location condition. That is, if the distance between any two objects is $\leq d$, where d is a predefined threshold, an edge will be drawn between the two objects (Fig. 5a). The GridClique algorithm works as follows:

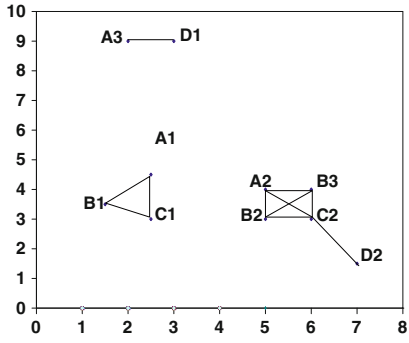
1. It divides the space into a grid structure which contains cells of size $d \times d$ (Lines 1–12). The grid space is structured where each cell has a key (GridKey). This key is a composite of X, Y, and Z coordinates (Line 5). We used a special data structure (hashmap) which is a list that stores data based on an index (key) to speed up retrieving the stored data. The number of cells depends on the maximum value of X and Y coordinates in the dataset. Having coordinates for each cell helps in placing the points in their corresponding cells. The total number of cells is calculated using (7).

$$\text{Number of cells} = \lceil \frac{\max(X)}{d} \rceil \times \lceil \frac{\max(Y)}{d} \rceil, \tag{7}$$

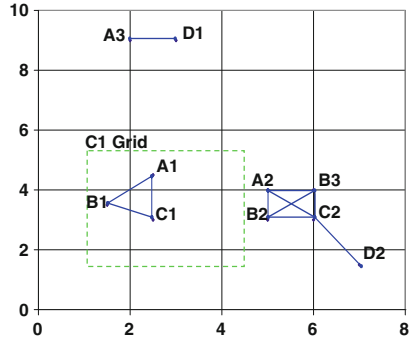
Algorithm 1 GridClique algorithm

Input: Set of points $(P_1, \cdots, P_n)$, Threshold d
Output: A list of maximal clique patterns.

{**Generating grid structure.**}
GridMap $\leftarrow \phi$
PointList $\leftarrow \{P_1, \cdots, P_n\}$
for all $P_i \in$ *PointList* **do**
  Get the coordinates of each point Pk_x, Pk_y, Pk_z
  Generate the composite key (GridKey=(Pk_x, Pk_y, Pk_z)).
  if *GridKey* $\in$ *GridMap* **then**
    GridMap $\leftarrow P_i$
  else
    GridMap $\leftarrow$ new GridKey
    GridMap.GridKey $\leftarrow P_i$
  end if
end for
{**Obtaining the neighborhood lists.**}
for all $p_i \in$ *GridMap* **do**
  p_i*.list* $\leftarrow \phi$
  NeighborGrids $\leftarrow$ (the 27 neighbor cells of p_i)
  NeighborList $\leftarrow \phi$
  if $NeighborGrids_i.size() > 1$ **then**
    for all $p_j \in NeighborGrids_j$ **do**
      if EucDist (p_i, p_j) $\leq d$ **then**
        $p_i.list \leftarrow p_j$ (p_i, p_j are neighbors)
      end if
    end for
  end if
  NeighborList $\leftarrow p_i$*.list*
end for
{**Pruning neighborhood list if at least one of its items violates the maximal clique definition.**}
TempList $\leftarrow \phi$
MCliqueList $\leftarrow \phi$
for all $Record_i \in$ *NeighborList* **do**
  RecordItems $\leftarrow Record_i$
  for all $p_i \in$ *RecordItems* **do**
    for all $p_j \in$ *RecordItems* **do**
      if *EucDist*(p_i, p_j) $\leq d$ **then**
        Templist $\leftarrow p_j$ (p_i, p_j are neighbors)
      end if
    end for
  end for
  MCliqueList $\leftarrow$ *Templist*
end for



(a) Spatial objects in a plane



(b) Grid structure

1	2	3	4	5	6	7	8	9	10
A1	B1	C1	D1	A2	B2	C2	D2	A3	B3
B1	A1	A1	A3	B2	A2	A2	C2	D1	C2
C1	C1	B1		C2	B3	B2			B2
				B3	C2	B3			A2
						D2			

(c) Neighborhood lists

1	4	5	8
A1	D1	A2	D2
B1	A3	B2	C2
C1		C2	
		B3	

(d) Maximal clique patterns

Fig. 5 An example to illustrate the process of extracting maximal clique patterns from two-dimensional dataset

where $\max(X)$ and $\max(Y)$ are the maximum of X and Y coordinates, respectively. The function $\lceil \cdot \rceil$ is the ceiling of any value.

In our example data, the maximum values of X and Y coordinates are 7 and 9, respectively. The predefined distance $d = 2$. Using (7), the number of cells will be 20. After structuring the grid space, the algorithm then places the points into their corresponding cells (Lines 6–11). This is performed by considering the X and Y coordinates of the corresponding cell as the $\lfloor X \rfloor$ and $\lfloor Y \rfloor$ coordinates of the placed point (Fig. 5b).[4] For example, if we consider object {A1}, its X and Y coordinates are 2.5 and 4.5, respectively; to place it in the grid space, its corresponding cell will be the one which has $GridKey = (2, 4)$.

2. GridClique finds (Lines 13–25) each object's neighbors and adds them to a list – this list is the neighborhood list. The algorithm checks the neighborhood list

[4] $\lfloor \cdot \rfloor$ is a function that gives the floor of a value.

members with respect to the definition of maximal clique which is all members (objects) are co-located with each other. In other words, the distance between every two objects is $\leq d$. For each object, the algorithm considers nine cells[5] (NeighborGrids in Line 15) to check their members if they are co-located with the object being checked.
Considering the example in Fig. 5, a list for each object is created. Our concern is to find only co-location patterns that have at least two objects (i.e., cardinality $\geq$2), because one object does not give co-location information. Therefore, there is no need to count objects that do not have connections (i.e., a relationship) with at least one other object. However, in our example all objects share relationships with others. For example, object {A1} has a relationship with objects {B1,C1}. It can be seen that these objects share the same neighborhood (co-located) – a neighborhood list will be generated for object {A1}. Figure 5c shows the neighborhood list for each object.

3. It prunes any neighborhood list that contains at least one object that violates the maximal clique definition (Lines 26–38). For example, list 7 is pruned because one of its members $\{A2\}$ is not co-located with $\{D2\}$. The shaded lists (2, 3, 6, 9, and 10) in Fig. 5c are pruned for the same reason. The $MCliqueList$ (Line 27) is a special data structure called *hashset*. This set does not allow repetition – this has helped us to report only distinct maximal clique patterns.

As a result of the previous steps, a list of distinct maximal clique patterns will be reported. For example, {A1, B1, C1} is a maximal clique, and so forth for lists 4, 5, and 8 (Fig. 5d).

GridClique Algorithm Analysis

This section discusses the GridClique algorithm's completeness, correctness, and complexity.

Completeness: All points in neighborhood lists appear as set or subset in maximal clique lists. After acquiring the entire neighbors for each point, another check among these neighbors is conducted to assure that all points are neighbors with each other. Intuitively, doing that results to have repeated neighborhood lists. Therefore, this ensures finding all maximal cliques in any given graph. The repeated neighborhood lists will be pruned using the *hashset* data structure.

Correctness: Every subset of a maximal clique appears in the neighborhood list. Thus, all maximal cliques that appear in maximal clique lists will not be found as a subset in another maximal clique, since this is the definition of maximal clique. Figure 6 displays an undirected graph and the neighborhood list and the existing maximal clique patterns. As can be seen, the pair $\{A, D\}$ does not appear in the neighborhood list, because the distance between A and D does not satisfy the co-location condition. Therefore, the pair $\{A, D\}$ will not be included in the maximal

[5] In two-dimensional space the number of neighbor cells is 9; however, in three-dimensional space it is 27.

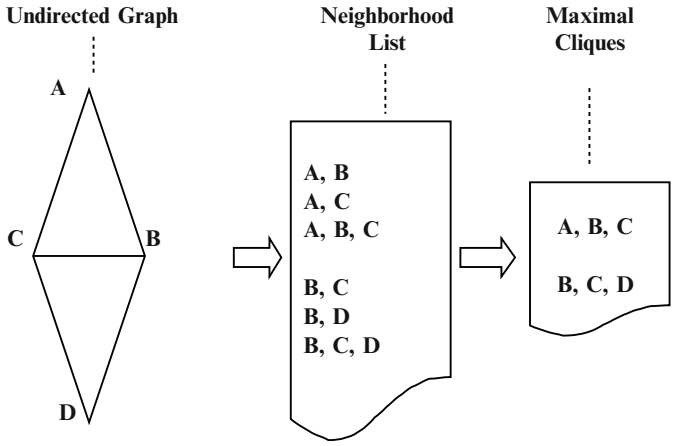


Fig. 6 Undirected graph contains two maximal cliques

cliques list. In other words, any subset of any maximal clique that appears in the neighborhood list will not appear as an independent maximal clique. In this way, the correctness of the proposed algorithm is shown.

Complexity: Firstly, assume there are N points and c cells, and assume that all points are uniformly distributed. Hence, on average there is N/c points per cell. Assume each cell has l neighbors. Then, to create the neighborhood list of one point, $l(N/c)$ points need to be examined to check if they are within distance d. Since the total number of points is N, the cost is $O(N^2 l/c)$. And since $c \gg l$, an assumption, that this part of the algorithm is sub-quadratic, can be stated.

Secondly, the pruning stage for the neighborhood lists. Again assume that on average the length of each neighborhood list is k. Then, for each neighborhood list, k points have to be checked against the co-location condition – the cost is $O(k^2)$. The total cost for this step is $O(Nk^2)$.

Ultimately, the total cost is the cost of putting the points in cell ($O(N)$), the cost of creating the neighborhood lists $O(N^2 l/c)$, and the cost of pruning the lists $O(Nk^2)$. Therefore, the complexity of the algorithm is $O(N(Nl/c + k^2 + 1))$.

3.3 Extracting Complex Relationships

A relationship is complex if it consists of *complex types* as defined in Sect. 1.

Extracting a complex relationship R from a maximal clique C_M is straightforward – we simply use the following rules for every type t:

1. If C_M contains an object with type t, $R = R \cup t$ (non-complex relationship).
2. If C_M contains more than one object of type t, $R = R \cup t+$ (positive relationship).
3. If C_M does not contain an object of type t, $R = R \cup -t$ (negative relationship).

Table 5 Spatial relationships with real-life examples from the astronomy domain [8]

Relationship	Notation	Description	Example
Non-complex	$A \rightarrow B$	Presence of B in the neighborhood of A	Sa type spiral galaxies $\rightarrow$ Sb type spiral galaxies
Positive	$A \rightarrow A+$	Presence of many instances of the same feature in a given neighborhood	Elliptic galaxies tend to cluster more strongly. E $\rightarrow$ E+
Negative	$A \rightarrow -B$	Absence of B in the neighborhood of A	Elliptic galaxies tend to exclude spiral galaxies. E $\rightarrow -$S
Complex	$A+ \rightarrow -C, B$	Combination of two or more of the above relationships	Clusters of elliptic galaxies tend to exclude other types of galaxies. E+ $\rightarrow -$S

If R includes a positive type $A+$, it will *always* include the basic type A. This is necessary so that maximal cliques that contain $A+$ will be counted as containing A when we mine for interesting patterns.

As mentioned earlier, the negative type makes sense only if we use *maximal cliques*. The last three columns of Table 1 show the result of applying Rule 1, Rule 1 and Rule 2, and all three rules, respectively. Table 5 provides four relationship types supported with real-life examples from the astronomy domain.

3.4 Mining Interesting Complex Relationships

In *itemset mining*, the dataset consists of a set of transactions T, where each transaction $t \in T$ is a subset of a set of *items* I; i.e., $t \subseteq I$. In our work, the set of complex maximal cliques (relationships) becomes the set of transactions T. The items are the object types – including the complex types such as $A+$ and $-A$. For example, if the object types are $\{A, B, C\}$, and each of these types is present and absent in at least one maximal clique, then $I = \{A, A+, -A, B, B+, -B\}$. An interesting itemset mining algorithm mines T for interesting itemsets. More details about the mining process can be found in [9].

4 Experiments and Results

4.1 Experimental Setup

All experiments were carried out on "Windows XP Pro" operated laptop with a 2.0 GHz Pentium 4M processor and 2 GB main memory. The data structures and algorithms were implemented in Java.

We used a real life three-dimensional astronomy dataset from the SDSS.[6] We extracted galaxies with special features giving a total of 365,425 objects (Sect. 3.1). The distance threshold used for generating the maximal cliques was 1, 2, 3, 4, and 5 Mpc; however, we mostly used 1 Mpc.

A total of 121,506 maximal cliques (transactions) were generated in 39.6 s. This is quite a large dataset. We processed these in a number of ways as described in Sect. 3.3.

4.2 Results

This section reports the results obtained from the use of the GridClique algorithm and the process of mining co-location rules.

Galaxy Types in Large Maximal Cliques

In this experiment we applied the GridClique algorithm on the "Main" galaxies extracted from SDSS data to generate maximal cliques with neighborhood distance as 4 Mpc. We selected the cliques with the largest cardinality (22). Figure 7a and 7b show the distribution of "Late" and "Early" type galaxies in the reported cliques, respectively. These results show that large cliques consist of more "Early" type galaxies (Elliptic) than "Late" type galaxies (Spiral). This conforms with the fact which says "Elliptic galaxies tend to cluster more strongly than any other galaxy objects" [10].

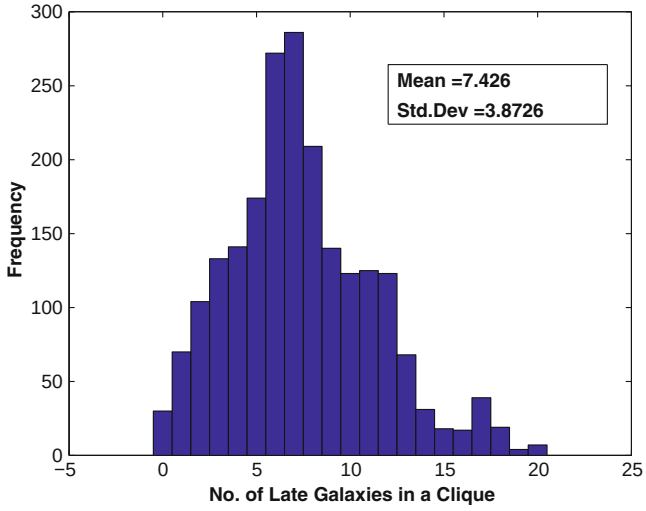
Cliques Cardinalities

Figure 8 shows the clique cardinalities in the "Main" galaxies. It shows that cliques with cardinality between 2 and 5 (small cliques) are more frequent than large cliques. In other words, in the universe there are no large clusters of galaxies. We mean by large clusters, large number of galaxy objects that are in the same neighborhood of each other. Although in this experiment we used very large threshold (4 Mpc), we obtained a large number of small cliques.

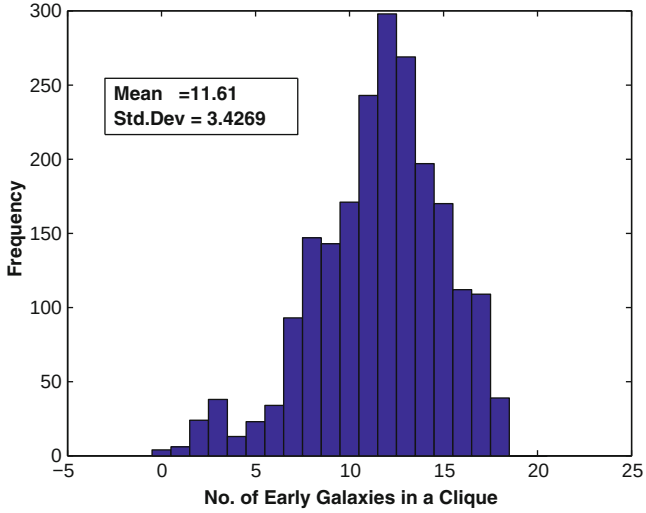
GridClqiue Performance

Since the previously proposed algorithms, which enumerate maximal clique patterns, are not specifically designed to mine the SDSS, a Naïve algorithm was

[6] http://cas.sdss.org/dr6/en/tools/search/sql.asp.



(a) MainLate



(b) MainEarly

Fig. 7 The existence of galaxies in the universe

implemented on the basis of brute force approach to obtain bench marks – this allows us to check the completeness of our algorithm. In this section, we show the effect of two factors on the GridClqiue algorithm, namely, distance and number of spatial objects. This section gives a comparison between the GridClique and Naïve algorithms as well.

Figure 9a shows the runtime of the GridClique algorithm with various numbers of objects (galaxies) and distance values. It illustrates that the runtime increases

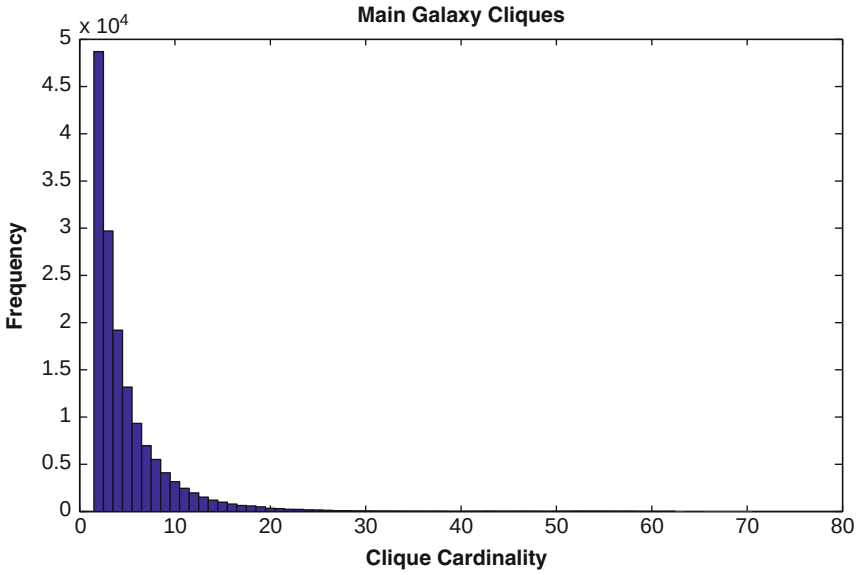
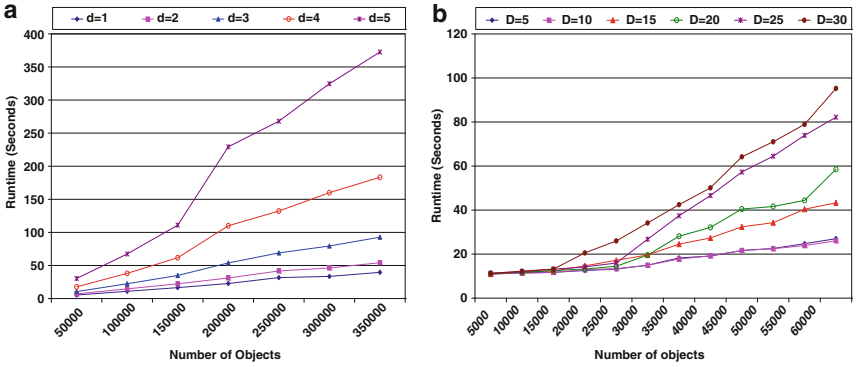


Fig. 8 Cliques cardinalities for Main galaxies using threshold = 4 Mpc



(a) The runtime of GridClique using different distances and number of objects. The distance and the number of objects were changed in increments of 1 Mpc and 50,000, respectively.

(b) The runtime of GridClique using different large distances and small number of objects. The distance and the number of objects were changed in increments of 5 Mpc and 5,000, respectively

Fig. 9 GridClique runtime

slightly as the number of objects and distance increase. The distance and the number of objects were changed in increments of 1 Mpc and 50,000, respectively.

To explain further, when the distance increases the grid size increases. By increasing number of objects at the same time, it allows more objects to appear in the

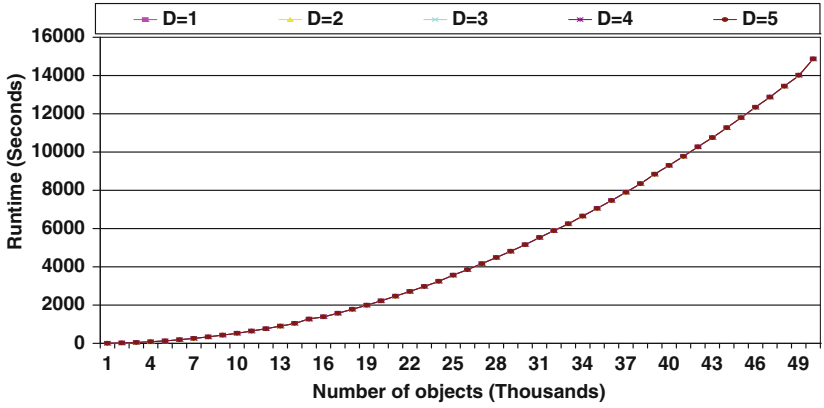


Fig. 10 The runtime of the Naïve algorithm

same cells or in the neighbor cells of the grid. In other words, increasing the number of objects increases the cell density. Hence, the two factors (distance and number of objects) affect the runtime of the GridClique algorithm.

In Fig. 9b, we show the performance of the GridClique algorithm using large distances. We changed the distance and the number of objects in increments of 5 Mpc and 5,000, respectively. This experiment shows that although the distances is large, the GridClique algorithm run time trends are similar to those when the distance is small, because the sparse nature of the astronomy data.

Figure 10 shows the effects of two factors (number of objects and the distance) on the Naïve algorithm runtime. It is clear that the algorithm is not affected when using different distances. However, its runtime increases exponentially as the number of objects increases.

We have carried out an experiment to compare the GridClique performance with the Naïve. Figure 11 shows that GridClique outperforms the Naïve algorithm with a difference of several order of magnitudes! We have used a distance of 1 Mpc; the number of objects was changed in increments of one thousand objects. The run time is given in logarithmic scale.

Interesting Rules from SDSS

By applying the *itemset mining* algorithm on the maximal cliques which are generated by GridClique, very interesting rules have been discovered. Table 6 lists sample of these interesting rules. Rules 1 and 2 show that while single instances of "Early" and "Late" galaxies are found in the neighborhood of each other, their clusters are found in isolation.

Rules 3 and 5 answer the question mentioned in Fig. 2. That means, the existence of a cluster of elliptical galaxies "Early" repels the existence of spiral "Late" ones.

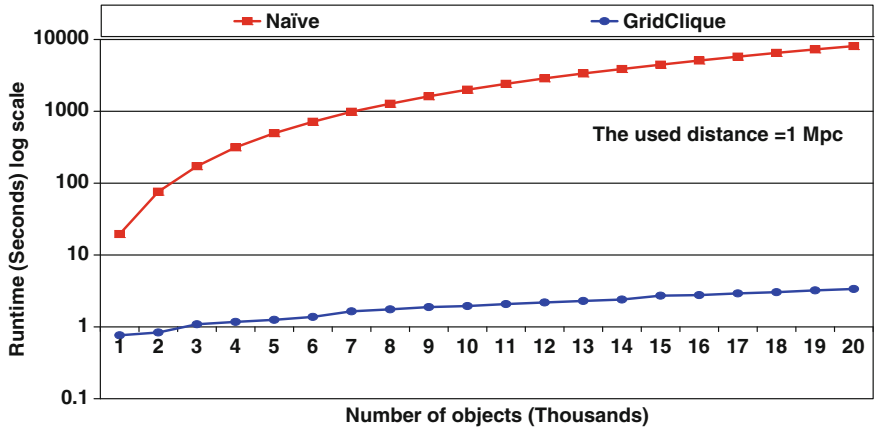


Fig. 11 Runtime comparison between the GridClique and the Naïve algorithms. The used distance was 1 Mpc

Table 6 Sample of association rules produced by our MCCRs technique, where the antecedents and consequents of these rules are galaxy-type objects

Rule number	Neighborhood distance = 1 Mpc Minimum confidence = 0.75
1	LrgEarly → LrgLate
2	LrgLate → LrgEarly
3	LrgEarly+ → −LrgLate
4	LrgLate+ → −LrgEarly
5	MainEarly+ → −MainLate
6	MainLate+ → −MainEarly

5 Summary

In this chapter, we demonstrated the problem of MCCRs in order to discover astronomical knowledge. That is, a systematic approach to mine complex spatial co-location pattern in SDSS data. We defined the term *maximal clique* in the context of mining complex spatial co-location. Maximal clique is fundamental to our work.

The MCCRs approach consists of two mining steps. First, it enumerates efficiently maximal clique patterns. In order to achieve the first mining step we have proposed a heuristic (GridClique) based on a divide-and-conquer strategy that considers all spatial objects as points in a plane. Then, it divides the plane into grid structure which helps in reducing the search space. The reported maximal cliques are considered to be the transactional data. MCCRs then uses the transactional data for mining interesting co-location rules using an association rule mining technique. The achieved results conformed to real facts in the astronomy domain and this has weighed up our proposed method favorably.

Acknowledgment I would like to thank A.P. Sanjay Chawla, Dr. Florian Verhein and Dr. Bavani Arunasalam for their valuable support in this work.

References

1. S. Shekhar, Y. Huang, *Discovering Spatial Co-location Patterns: A Summary of Results. Lecture Notes in Computer Science*, vol. 2121 (Springer, Berlin, 2001)
2. G. Al-Naymat, in *Proceedings of the 6th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, pp. 126–133, Doha, Qatar (2008)
3. F. Verhein, S. Chawla, in *IEEE International Conference on Data Mining*, pp. 655–666, Los Alamitos, CA, USA (2006). IEEE Computer Society
4. Sloan Digital Sky Survey (SDSS) (2005), http://cas.sdss.org/dr3/en/help/download/. Accessed 5 Aug 2005
5. D. Spergel, M. Bolte, W. Freedman, *Proc. Natl. Acad. Sci. U.S.A.* **94**, 6579 (1997)
6. M. Haynes, S. Churchman, *Hubble's Law* (2005), http://map.gsfc.nasa.gov/
7. V. Martin, E. Saar, *Statistics of the Galaxy Distribution* (Chapman and Hall/CRC, Boca Raton, FL, 2002)
8. B. Arunasalam, S. Chawla, P. Sun, in *Proceedings of the Fifth SIAM International Conference on Data Mining (SDM)*, pp. 173–182 (2005)
9. F. Verhein, G. Al-Naymat, in *The 2007 International Workshop on Spatial and Spatio-temporal Data Mining (SSTDM) in cooperation with The 2007 IEEE International Conference on Data Mining (ICDM)*, pp. 679–684, Los Alamitos, CA, USA (2007). IEEE Computer Society
10. J. Gray, D. Slutz, A. S. Szalay, A. R. Thakar, J. vandenBerg, P. Z. Kunszt, C. Stoughton, in *Data mining the SDSS skyserver database*, Tech. Rep. MSR-TR-2002-01, Microsoft Research (2002)

“This page left intentionally blank.”

Part IV
Future Trends

“This page left intentionally blank.”

On-board Data Mining

Steve Tanner, Cara Stein, and Sara J. Graves

Networks of remote sensors are becoming more common as technology improves and costs decline. In the past, a remote sensor was usually a device that collected data to be retrieved at a later time by some other mechanism. This collected data were usually processed well after the fact at a computer greatly removed from the in situ sensing location. This has begun to change as sensor technology, on-board processing, and network communication capabilities have increased and their prices have dropped.

There has been an explosion in the number of sensors and sensing devices, not just around the world, but literally throughout the solar system. These sensors are not only becoming vastly more sophisticated, accurate, and detailed in the data they gather but they are also becoming cheaper, lighter, and smaller. At the same time, engineers have developed improved methods to embed computing systems, memory, storage, and communication capabilities into the platforms that host these sensors. Now, it is not unusual to see large networks of sensors working in cooperation with one another. Nor does it seem strange to see the autonomous operation of sensor-based systems, from space-based satellites to smart vacuum cleaners that keep our homes clean and robotic toys that help to entertain and educate our children.

But access to sensor data and computing power is only part of the story. For all the power of these systems, there are still substantial limits to what they can accomplish. These include the well-known limits to current Artificial Intelligence capabilities and our limited ability to program the abstract concepts, goals, and improvisation needed for fully autonomous systems. But it also includes much more basic engineering problems such as lack of adequate power, communications bandwidth, and memory, as well as problems with the geolocation and real-time georeferencing required to integrate data from multiple sensors to be used together.

Given the limitations of current systems, what is driving the push to develop sensor networks and autonomous systems? What place does data mining have in such environments? What techniques and solutions are people using to work around

S. Tanner (✉)
University of Alabama in Huntsville, AL, USA
e-mail: stanner@itsc.uah.edu

M.M. Gaber (ed.), *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*, DOI 10.1007/978-3-642-02788-8_13,

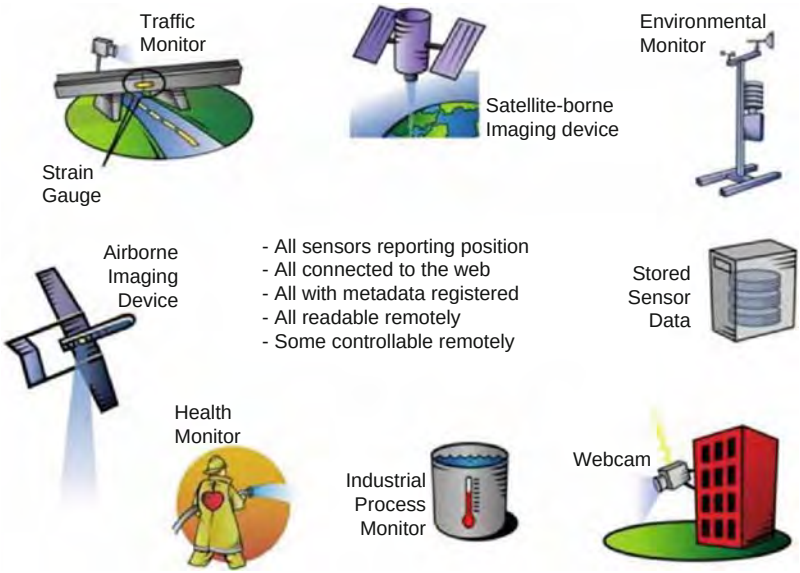


Fig. 1 The Open Geospatial Consortium's Sensor Web Enablement (SWE) specifications enable developers to make all types of sensors, transducers, and sensor data repositories discoverable, accessible, and useable via the Web (Courtesy of Open Geospatial Consortium, Inc.)

the problems encountered in the real world? At this point in time, the answers are almost all pragmatic in nature rather than absolute or rigorously and mathematically proved. Like most robotic and remote sensing systems, on-board data mining is more of an engineering endeavor than an academic one.

For on-board data processing efforts, it is important to work closely with domain scientists to verify that the algorithms accurately identify the phenomena of interest to them. They can also help with identifying false alarms or missed detections, and weighing the costs of each. Also, on-board algorithms don't have to duplicate all of the details that would result from the complete processing of an image on the ground. For example, in an Earth-monitoring application, rather than giving detailed results about a dust storm, the on-board algorithm can simply detect the dust storm and mark the data as high priority for downlink. Then, further processing can be done on the ground. Using the simplest processing techniques possible is a good approach, given the constraints. Complexity should be added only as necessary and feasible [11].

1 Problems Encountered with On-Board Mining

Before delving into current and planned efforts in on-board data mining, it is useful to consider the types of problems and issues that are likely to arise with such systems. What technical roadblocks have stood in the way of widespread use of data

analysis and mining in on-board systems? The simple answer is severe resource constraints. On-board processing by definition means working in a constrained environment of some sort. The constraints vary considerably but generally fall into several broad categories: power, bandwidth, computation, and storage. In addition, the on-board environment introduces two problems associated with the data itself that the system must overcome: noise, and incorrect and incomplete measurements. What's more, the data must be georeferenced in some way, so that analysis algorithms can correlate data coming either from different sensors or from temporally separate readings. Typically in non-on-board systems, the data cleaning, and georeferencing steps are completed at a central location as part of post-processing, and result in several different data products, which are often not available until days, weeks, or even months after the data is initially gathered. This luxury is not available for on-board applications, which must deal with the noisy data results and with tagging the data through the use of a georeferencing scheme to make the data immediately usable.

1.1 Power

Perhaps the most obvious problem in on-board data mining applications is power. Many on-board mining applications are associated with sensor platforms that are physically removed from any consistent or sizable power source. This has a profound impact on how much data analysis can take place on-board the platform. Even if there is sufficient computational capability and storage, there may not be enough energy to drive the computers. This is a very different environment from desktop, cluster, or grid computation, where power is rarely a concern, and when it is, it is addressed well beforehand.

In the case of space-based systems, the lack of adequate power is due to being located in a remote and extremely hostile environment. Most space-based systems have batteries and some sort of recharging capability (e.g. fuel cell or solar), but it is rarely enough to drive all of the on-board systems all of the time. In the case of solar power generation on satellites, great care must be taken to schedule tasks such that they are compatible with the day–night orbital path of the platform – something which few data mining researchers have to contend with.

For many ground-based systems, the problem is often one of size/weight constraints. Often, the sensors must be small and disconnected from a power grid, making the use of battery power a necessity. This may be due to the remote locations of placement (e.g. for environmental monitoring in remote geographic areas). It may also be due to the need for stealth, as in the case of surveillance and military sensors, or at least discretion, as is often the case for security cameras.

The implication of this dearth of energy is the need to husband power consumption very aggressively. For data mining, that may impact the type of algorithms used, how they are used, or when they are used. Often software developers will need to consider the efficiency of a process in light of the sensor data size to determine how

much energy a given algorithm will need. In some cases, the data mining plans may need to either use a lower resolution data feed (for example, sampling every other pixel instead of every pixel) or a lower sampling rate (sampling every other image instead of every image). In the most adaptive systems, the scheduler may be able to decide how to approach power consumption in a dynamic way – for example using higher power consuming algorithms during the daylight hours and a simpler, lower power version at night.

1.2 Bandwidth

Bandwidth is a significant obstacle in on-board data mining. Communication from a sensor platform, either with other sensors or with central downlink locations, is an expensive endeavor. It is expensive in terms of time, required hardware (and thus weight), power, and computational capabilities. Most developers of sensors and sensor platforms strive to limit either the frequency of communications or the amount of data sent, or both. For many sensor systems, there simply isn't enough bandwidth to send all of the gathered information, or just as limiting, there isn't enough power to drive the data through the network connections. This is especially true of wireless sensor networks, where passing information to a central processing center requires multiple hops from sensor platform to sensor platform.

This limits the amount of data mining that can be accomplished, since data mining algorithms may not have access to all of the potential data available. Data mining in pursuit of data fusion using multiple sources becomes problematic. However, data mining may actually be able to play a vital role in resolving this bandwidth problem. If the proper algorithms are employed, quick data analysis on-board a platform may be able to filter out unnecessary data, leaving only valuable data to vie for the limited communication resources.

In the case of earth science satellites, often simple filters are employed to eliminate poor sensor measurements. Cloud masks are an example of this, for instruments that can neither measure through clouds nor gather useful data about the clouds themselves. However, developing real-time cloud mask filters is nontrivial. Sun glint, snow packs, and other environmental factors make this a difficult task.

Data mining classification algorithms are a prime example of techniques that may be able to reduce the amount of data needed to be transmitted. These can even be used to deal with concept drift and environmental changes, for example through the use of a dynamically changing ensemble of classifiers, each tuned to a different environmental condition.

One must consider the impact of data mining on the need for bandwidth, and care must be taken not to overwhelm the network capacity. The most appropriate mining algorithms then are ones that reduce either the amount of data or the dimensionality. Luckily, these are often the very algorithms used in the initial data processing for data mining.

Fig. 2 Many commercial venders are offering a variety of small, inexpensive wireless sensor platforms and sensor packages. This small platform from Crossbow Technology, Inc. contains an accelerometer, temperature, humidity, and light sensors – perhaps useful for environmental monitoring applications (Image courtesy of Crossbow Technology, Inc.)

1.3 Computation

Computational limitations, combined with the need for timely results, are probably the biggest hurdle to overcome in the deployment of on-board data mining applications. Simply put, there may not be enough computing cycles to thoroughly analyze the gathered data on-board the sensor platform. This means, sacrifices must often be made in the amount or type of analysis that can fit within the computational constraints of the on-board system. Two typical approaches are to limit the resolution of the data to be analyzed or to limit the frequency of the analysis.

Consider a frame-based camera system linked with an on-board image process analysis application, in which the camera generates a series of M images per second, each of size $n \times n$ pixels. For the sake of simplicity, consider the use of a straightforward convolution algorithm as one step in the data mining process. A typical convolve algorithm uses a mask of some sort to filter the 2D image in one pass – so its computational requirement is approximately $(n^2 \times M) \times C$, where C is a constant based on the complexity of the algorithm. If there isn't enough computational power to meet this requirement, the developer has two basic choices: either limit the resolution of the images (reduce the size of n) or limit the number of images to process (reduce the size of M). The choice will depend on the application's overall goals and requirements. Which sacrifice has the least impact on the final outcome – lower resolution or fewer frames? In an application that monitors for intrusion detection, perhaps the frame rate isn't particularly important, but higher resolution images, which may help with identification of the intruders, are. In an

application that is part of a rocket propulsion health management system, perhaps the frame rate and quick response to millisecond changes is the overriding concern – in this case, changes are of a fast but relatively coarse nature.

1.4 Storage and Memory

The idea of on-board data mining implies access to the data being collected. But how much data is gathered depends heavily on the sensor platform and the storage and memory resources available. Often, when the platforms are designed, these areas are given short shrift. Memory utilizes precious power, and storage takes up valuable space and weight. This may greatly impact the types of algorithms that can be used and the approaches taken to perform the data analysis. Typically, this means that there is little or no data archived, so there is little chance for multiple passes over the data. This makes taking advantage of temporal analysis difficult. Algorithms that focus on dealing with streaming data may be the best approach – many of these are designed as one-pass analysis tools.

Another issue that may arise is concept drift or changes due to environmental conditions. Since there may be no archived data to provide historical context, care must be taken to deal with such things as diurnal, seasonal, or weather-based changes. Imagine an outdoor sensor monitoring a city park. The environment there will look quite different depending on the season – snow, deciduous trees, even differing clothing fashions will shift from season to season. If an historical archive of these changes is available, adapting to them is considerably easier. But what of an autonomous system without access to such archives? For systems that have periodic contact with a base station, this issue may be addressable through manual means – by uploading new classification algorithms as seasons change, for example. Without that access, the system needs to be able to adapt to these changes on its own. Such adaptability can be especially difficult to implement. For example, one approach is to have a group of algorithms to select from, and base the selection on the latest results – if the last run of the analysis indicated rain, then use algorithms that take that into account, until rain is no longer detected. However, in a rapidly changing environment this approach may lead to a system that is constantly chasing what happened in the immediate past rather than what is happening right now.

1.5 Georeferencing

One can image a fully functioning sensor network comprised of a group of heterogeneous sensor platforms connected together via some communications means. Each of these platforms would have one or more sensors connected to it and have a computing facility including a processor and memory. In many cases, the topology of the network, and the capabilities of each platform and its sensors would be known, allowing for the development of applications geared specifically for such a network.

However, in many cases, such a priori information may not be on hand. How then can the best use of these resources be made?

One approach is to use some type of formal descriptions as a means to pass information between sensors, the platforms they are connected to, and remote computing locations separate from the sensors and platforms themselves. Through the use of such formal means, sensing processes running on the sensors or sensor platforms could advertise their capabilities while other analysis and decision support processes running at any number of locations could discover these capabilities and match them with their own needs.

2 The Use of Standards for On-Board Data Mining

There are a number of approaches being researched and implemented for resource discovery and resource advertising within computer network systems. Most of these are aimed at either specific types of services or specific types of network architectures. However, with the advent of web services protocols, there is a move to make resource discovery and advertisement a more general purpose service within the wider network community. One such approach that has direct bearing on on-board data mining is undertaken by the developers of *SensorML*.

2.1 SensorML

SensorML is part of a larger effort by the Open GIS Consortium (OGC) to develop technologies and standards that will support better use of Geographic Information Systems (GIS) data (OGC). SensorML operates under the assumption that there are three fundamental types of information that may be observed. These types of information are the object to be observed, which must have physical properties that can be measured and quantified; data values taken from observations of that object by a sensor; and meta-data about the sensor, including the location and time of the observations. Meta-data about the sensor may also include characteristics of the sensor that can help a user understand the values of the measurements as well as their quality and veracity.

SensorML is concerned primarily with the description of sensor characteristics. The information is seen by the authors [SensorML] as being used in three primary ways:

- To process raw sensor data into usable measurements and for georegistration
- To provide a limited amount of data conversion on-board the sensor or sensor platform itself
- To provide either on-board or downstream processes with sensor characteristics that may impact further data analysis.

SensorML includes descriptions of:

- Observation characteristics such as physical properties measured, quality characteristics, and response characteristics.
- Geometry characteristics such as size, shape, and spatial weight function of samples, as well as geometric and temporal characteristics of sensor data collections
- Description and documentation such as history and reference information

In summary, the primary use for the language is to make access to sensor data more automated and straightforward by supporting the description of sensors and systems of sensors.

2.2 Describing Sensors in SensorML

The most basic definition assumed by SensorML is that "sensors are devices for the measurement of physical quantities." This is clearly a very broad definition, encompassing everything from the simplest thermometer to the most complex on-orbit satellites. In fact, this also includes the use of humans as the measurement device. A more subtle aspect is that the term "measurement" is also broadly used here, and is not limited to a numeric quantity.

From a modeling point of view, sensors are typically thought of as one of two primary types. The first is in-situ sensors, which typically measure something within its immediate area (e.g. a room-monitoring system). The second is remote sensors, which typically measure things from a much greater distance (e.g. a satellite that measures reflected radiation from the Earth's surface).

This distinction between the types is important to understand because SensorML uses only one type of coordinate system to describe both types of sensors. Any geometric properties described within the schema are defined in terms of the local coordinate system (local to that sensing component). It is only through the use of an association between the sensor and its platform (with its own coordinate system), and that platform's geospatial reference frame, that the sensor can be properly placed in relation to both its environment and other sensors. This makes it possible to describe a sensor once and then deploy it to many locations, including mobile ones. The implication for on-board data mining, especially for orbital platforms, is large.

Since the physical location of a sensor is not part of its descriptions, one might assume that it would make sense to describe a sensor type, and then have that description applied to a whole group or family of sensors. After all, this is the way XML languages are used to describe groups of data sets and other standard entities. However, each sensor is unique in a variety of ways, requiring a unique XML description. For example, each sensor has unique identifiers such as ID and serial number, as well as additional calibration information specific to that sensor. SensorML is set up to gather this additional information, and can also store a history description that records the changes to the sensor as time progresses.

2.3 Sensor Platforms

A "sensor system" includes one or more sensors and the platform on which they reside. The platform does not make any measurements of physical phenomena, and is thus not a sensor itself. However, the platform does measure its location and orientation, which may be dynamic, as in the case of using an aircraft or a satellite as a sensor platform. The platform's location and orientation information is very germane to the readings of the sensors aboard that platform. So, while the sensor platform and the sensor itself are considered to be two separate entities, they have a very strong association with one another.

This association is accomplished through the use of "coordinate frames." The sensor and the platform each have a coordinate frame. The sensor's coordinate frame locates the sensor on the platform, while the platform's coordinate frame locates the platform in spatial relation to the larger environment. The coordinate frames are described in terms of coordinate reference systems (CRS). The relationship between a sensor's CRS and its platform's CRS is then used to relate both to an external CRS, such as geographic latitude and longitude. These CRSs enable the georegistering of sensors and their measurements. Currently, CRSs are applied only to location, not to temporal information. For example, a relationship between "Earth time" and the delta time of a sensor's scan start cannot be captured in current CRSs.

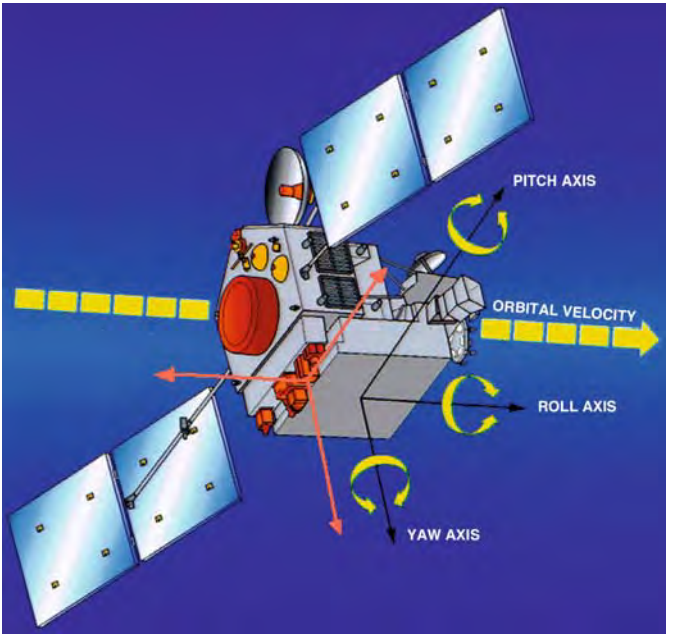


Fig. 3 Relationship of sensor frame (*p*ink) to the moving platform frame (*b*lack) (Courtesy of Open Geospatial Consortium, Inc.)

2.4 Measurements and Response

SensorML can be used to describe properties of a sensor's measurements, specifically, the physical properties being measured by the sensor and the quality of the measurements.

Response characteristics can also be modeled. For example, SensorML can model a sensor's responsiveness to changing environmental conditions. These responses include sensitivity, accuracy, and precision. This information could be very useful to an ensemble data mining approach for dynamically selecting the algorithms most appropriate for the given current conditions.

Overall, by providing a way to describe sensors and system of sensors, and their characteristics and locations, SensorML aids the on-board data mining effort by addressing the problem of georeferencing data from sensors that move.

3 The Use of FPGAs for On-Board Systems

One approach to addressing the problems of limited computing resources and limited power availability is through the use of Field Programmable Gate Arrays (FPGAs) as a means to drive the on-board computing systems. Simply put, FPGAs provide a means to implement software algorithms in hardware, thus increasing speed and hopefully reducing power consumption and heat generation. FPGAs are based on reconfigurable components such as flip-flops and multiplexers, with a bank of SRAM to hold the configuration information and control the functionality of the device. The state of the bits in the SRAM determines the behavior of the device, allowing it to be adapted or customized for different tasks as needed. While somewhat slower than customized hardware, their adaptability makes FPGAs more attractive than a customized chip or hardware solution. The primary downsides to using FPGAs are twofold: the hardware is more expensive, and they can be difficult to program.

Because they are reconfigurable, FPGAs are good tools for prototyping systems. In early FPGAs, the time to reconfigure the system was substantial, limiting their use for actual on-board systems. However, as the cost and time for reconfiguration have dropped (some systems approach reconfiguration times of milliseconds), and as they have become more sophisticated, they have come to be used in production systems. They are especially well suited to data mining applications because they have great potential for parallelism, they perform bit-level operations very efficiently, they can accommodate high bandwidth, and they can be changed dynamically to suit an individual application [68].

There are a number of applications where FPGAs have been used or tested for their applicability to data mining. In one study in Greece, the authors found that implementing the a priori algorithm for data mining using FPGAs was several orders-of-magnitude faster than the fastest available non-FPGA implementation [19]. In other studies, FPGAs were used for image processing and weather modeling, both

required substantial numbers of matrix operations. In one case, FPGAs were used in conjunction with a host computer to perform real-time image processing algorithms from an image stream. For each image, line buffering was performed using a 2D digital Finite Impulse Response filter convolution kernel; then noise reduction was performed using a Gaussian 3×3 kernel, followed by edge detection using Prewitt filters. The system performance was substantially faster than a software only solution, but still provided the researchers with the necessary flexibility to adapt the system [15].

In Turin, Italy, FPGAs have been used in a system that collects temperature, humidity, and precipitation data on a routine basis every 15 min and uses that data to make very local weather predictions for the next one, two, and three hours. Such forecasts are especially useful in applications where weather can cause dangerous conditions within a small area. For example, rain can have a substantial impact on the course of a race, and a thunderstorm or fog can shut down an airport. Predicting weather involves considerable data from different sources and is usually computationally intensive. In most cases, observation data is sent to a central computing location, often a large-scale cluster configuration. In this case, however, data mining techniques are employed locally to select the best predictors for a given set of conditions, since the cross-correlations among predictors may change depending on conditions. By implementing such a system using FPGAs, the performance requirements for real-time forecasting were met, and the system was platform independent and adaptable to other applications [14].

In another study using FPGAs for image processing, a single FPGA was used, without any host computer or other co-processors. This created a very inexpensive system with high performance and very low power consumption of only 200 mW, compared to existing systems created to do the same sorts of tasks, which used 600-4,000 mW. The FPGA system was at least comparable in speed to other systems, at a tiny fraction of the cost [16].

The ability to achieve high performance with low expense and power consumption is obviously very appealing for satellites and space applications. One system was designed using FPGAs to ascertain the physical properties of an observed area from satellite data in real time, with the intention of using the system for on-board image analysis and data compression. Data streams from different sensors had to be fused and processed to accomplish this. The system was created by connecting the sensors to a PCI bus, and connecting a host processor and a series of FPGAs to the same bus to process the data. With this simple architecture, the system was faster than Pentium-based PCs by two to three orders-of-magnitude. In addition, the authors suggested that further improvement could be achieved by replacing the bus with interconnections among the FPGAs, since the bus was the primary bottleneck in their studies, and by improving the parallelization of the algorithms used [13].

An FPGA-based system has been proposed for use aboard the Mars scout mission Mars Volcanic Emission and Life (MARVEL), expected to launch in 2011. On-board will be the Mars Atmospheric Trace Molecule Spectroscopy (MATMOS) instrument, observing the sunrise and sunset in a 3-min observation period everyday. These observations will produce far more data than can possibly be downlinked

to Earth, but 112 min per day will be available for data processing, which will include heavy use of FFT [17]. That may not sound like much processing time, but considering the environment, it is a substantial commitment of resources.

Several existing systems were considered and discarded because they lacked the processing power to perform this task in the time allotted. However, an FPGA system-on-a-chip solution was found to meet the requirements. The FPGA system is cheap, small, scalable, reconfigurable, and uses very little power. FPGAs are especially suited for use in systems for space missions because they are easier to change, update, and reuse than traditional hardware; they carry less risk of obsolescence; they lend themselves to parallel processing and modularity; and they offer more implementation options than traditional systems. Notably, with an FPGA system, the division of processing between hardware and software is flexible [17].

For this application, an FPGA system was the only available choice that could keep up with the requirements for on-board data processing. The authors estimate that the on-board data processing system they propose will reduce the volume of data for downlink by about 80 times, resulting in an expected savings of tens of millions of dollars over a 2-year mission [17].

Some current space observing missions are using multiple sensors of different types on the same satellite, resulting in large quantities of heterogeneous data to be downlinked and processed. As instruments are deployed farther from Earth, downlinking all of the data becomes less and less feasible, requiring years or decades for deep space missions. For those applications, on-board data fusion becomes critical, and it must be accomplished within the constraints of low power consumption [13]. One clear direction for computing aboard these missions is using FPGAs in a smart payload computer for space, using multiple FPGAs in parallel and large memory banks to hold the data waiting to be processed. Depending on the application, custom co-processors such as a dedicated FFT core may also be used. By implementing these ideas and parallelizing processing as much as possible, new systems can achieve performance that was impossible with traditional processors [17].

By enabling very efficient parallel processing, FPGAs represent a considerable step toward addressing the problems of power, computation, and, to a certain extent, memory and storage in an on-board data mining system. As more sophisticated data mining can be done in an on-board system, the issue of bandwidth will also be alleviated – these systems will be able to summarize, filter, and prioritize data so that the volume of data needing to be downlinked is reduced considerably.

4 Applications for On-Board Data Mining

Although the problems and obstacles associated with on-board data mining are very real, advances in technology have enabled on-board data mining to be performed in some applications. On-board data mining has already proven useful in the areas of autonomous and unmanned vehicles, as well as biometrics.

4.1 Autonomy

One area where on-board data mining has been successfully implemented and deployed is in autonomous spacecraft. For components such as satellites and planetary rovers, the traditional approach has been to gather and store the data on-board the component, and then wait for an opportunity to send it all to ground-based downlink stations on Earth for processing. However, there are problems and severe limitations with this approach, the primary one being that there is usually limited bandwidth available to send the data to Earth [6]. In some cases, a satellite has the capability to collect much more data than it can reasonably transmit. If all data processing and analysis is done at a ground-based facility, any data that exceeds the downlink capacity cannot be used [10]. Also, when interesting events or discoveries are uncovered in the data that comes from a satellite or rover, often further observation of the phenomena would be desirable. If data analysis is performed after the fact, there may be a considerable time lag between the time when a phenomenon is first observed by the instrument and when it is detected on the ground. By the time the data is downlinked and analyzed, retargeting the craft to further study the phenomenon becomes difficult or impossible [6].

However, if the data can be processed on-board the craft, an interesting phenomena can be detected much more quickly, and the craft can react dynamically by retargeting itself for further observations. Furthermore, if data processing occurs on-board the craft, the data can be prioritized and only the interesting or relevant data would be sent to Earth [6]. For instance, if an image is obscured by clouds or other weather that the instrument cannot penetrate, there is no need to waste bandwidth downlinking it. (Of course if scientists are interested in studying clouds, that is another matter.) On the other hand, if an instrument's data contains information about a new and unexpected volcanic eruption, that data should be a high priority for downlinking, and should be able to preempt other lesser data aside in the schedule.

There are several projects in which on-board data mining has been successfully implemented on autonomous space craft. Three of the most successful are EO-1, Mars Odyssey, and rovers on Mars.

4.2 EO-1

Earth Observing-1 (EO-1) is an Earth science oriented satellite, flying in the same orbit as Landsat 7. One of its missions is to test new technologies for remote earth observation [7]. In 2003, the Autonomous Sciencecraft Experiment (ASE) was launched aboard EO-1. The ASE commands EO-1 and retargets the spacecraft in response to events of interest to scientists, capturing further detail of those events as they occur [4]. ASE consists of three parts: on-board science processing algorithms, which analyze data on-board the craft, detect events, and specify an appropriate reaction to those events; on-board planning and scheduling software, which schedules

the craft's activities based on analysis of the data, input from the ground, and the craft's capabilities; and robust execution software, which monitors the plan and progress, making sure the craft operates within the bounds of safety even in the face of anomalies [4].

Specifically, science analysis algorithms look for recognizable landscape features or events such as floods, ice formations, clouds, and volcanoes [5]. Before launch, scientists chose locations to monitor events to look for, and an appropriate course of action for each event. They also used archived data to train automatic feature recognition software to recognize specific types of events and landscape features. Once on-board the spacecraft, the algorithms can monitor an area for changes over time or unusual occurrences by comparing multiple images taken over the same location [3].

This is where the strength of data mining in an on-board environment comes into play. As part of the on-board data analysis, scientists developed a Support Vector Machine (SVM) classifier to identify the pixels in an image as water, ice, land, snow, or cloud. Sometimes, if data is missing or bad in some bands, an extra "unknown" class is also used. Knowing the proportion of each of these elements in the image can give clues as to what is happening on the ground. For instance, if an image is not obstructed by clouds and the area has the proper proportions of snow, ice, and water, then that area is flagged as probably containing ice breakup [6]. This type of event can prove useful in studying the effects of global climate change.

When the algorithm detects an event of interest or an anomaly, it can summarize the data, send an alert, change the craft's observation schedule to study the phenomenon further, or simply select only the relevant data to downlink to earth [3].

Once the science analysis algorithms have identified a phenomenon for further study, the on-board planning and scheduling software rebuilds the spacecraft's plan to include further observations, if this is feasible and in keeping with the craft's mission priorities. The Continuous Activity Scheduling Planning Execution and Replanning (CASPER) system is responsible for this task. Based on observations, plans from the ground, and priorities set by the scientists, the CASPER system can retarget the craft for further observations [4]. The CASPER system works by taking all the goals specified for the spacecraft by scientists and by the automated system, considering the current state of the craft, and identifying all conflicts. Then it iteratively modifies the plan until all conflicts are eliminated, using the scientists' specified priorities for data observation [5].

The CASPER system conveys the plan to the robust execution system using spacecraft command language (SCL) [4]. Then the robust execution system executes the plan. It monitors the state of the craft and checks each command for safety before executing it. In the case of a hazard or anomaly, the robust execution system can override the plan generated by CASPER to preserve the safety of the craft [5].

In action, the ASE system operates in a cycle. First, ASE gets prioritized goals from the ground. Then, CASPER makes a plan, choosing the appropriate instruments for each target. The EO-1 spacecraft takes images of the target, and the algorithms analyze those images. If they include changes or phenomena of interest, a new goal of continuing to monitor the location is created. Otherwise, the data

in question is marked as not of interest and is not downlinked to earth. If a new goal has been generated, CASPER makes a new plan, and the cycle starts again [5].

The ASE system was created in the face of some difficult challenges. One of the major challenges was communication with earth. The EO-1 spacecraft is in communication with earth for only eight 10-min periods per day. This puts great limitations on the amount of input and control from the ground, and the amount of data that can be downlinked. Meanwhile, the instruments on the craft have limited observation capabilities, so the craft must control itself without being able to see the bigger picture. All this must be done with very limited computing resources. On-board EO-1, ASE was allocated 4 MIPS and only part of the craft's 128 MB of memory. Yet the ASE system has to manage a complex spacecraft with thousands of parts in the hostile environment of space at high financial stakes, ensuring the safety of the craft and maximizing the scientific return from it [4].

Since launch, the ASE system has had considerable impact on the EO-1 mission. The ability for the craft to re-task itself dynamically has enabled much greater flexibility in observations. Previously, plans for the craft were made 5–11 days in advance; with ASE, the timeframe is more like several hours. Also, ASE makes it easier to work around anomalies. Before, the planning and scheduling was done by hand and required considerable thought and knowledge of the spacecraft's capabilities. Automating that task has saved considerable time. Overall, implementing ASE has resulted in cost reductions of over $1 million per year [5].

4.3 Mars Odyssey

As with EO-1, the Mars Odyssey satellite, which observes and maps the surface of Mars, has benefited from on-board data mining efforts. In particular, this craft had the potential to cover more area if there were sufficient bandwidth to downlink the data [10]. Given the success of ASE, scientists wanted to create a similar system to observe Mars [12]. As on EO-1, on-board data mining could enable detection and fast reaction to events as they happen as well as prioritizing the data for downlink. Also, in the course of analysis, additional products such as summaries and alerts are generated with little or no extra effort. These products can be very useful to scientists, and they can be downlinked in place of a whole file when expedient to save bandwidth [11].

In this effort, the Thermal Emission Imaging System (THEMIS) camera on the Odyssey was used. This camera was chosen because of its high spatial resolution and thermal sensitivity [11]. Areas of particular interest to scientists observing Mars include detecting thermal anomalies, dust storms, and the movement of polar ice caps [12].

Detecting thermal anomalies on Mars is of particular interest to scientists because so little is known about Mars. Thermal anomalies are caused by lava flow, frost at low altitude, and very fresh impact craters. They also could be caused by subsurface hydrothermal activity, which would be a significant discovery. An on-board data mining algorithm was developed to detect these thermal anomalies using

a threshold based on the season, location, and time of day. If a pixel has a value outside the expected range, it is labeled as an anomaly. Post-processing is conducted to eliminate images with too many pixels flagged, which is indicative of a false alarm [11].

Another area of interest to scientists is estimating aerosol opacity. Specifically, algorithms have been developed using an SVM regression model to flag dust and water ice clouds, which are of interest in understanding the Martian atmosphere [11].

Monitoring the seasonal migration of the polar ice caps on Mars is also of interest to scientists [10]. These ice caps are made of CO_2, and grow and shrink each year with the seasons. This change represents a significant change in the distribution of mass on the planet, which leads to a change in the center of gravity, that is so dramatic it can be observed from earth [10].

In observing the ice caps, one goal is to monitor the location of the edge of the ice cap. To this end, images were taken of the north polar ice cap, on the side of Mars facing the sun, from north to south. Some images contained only ice cap, some contained only noncap land, and some contained the edge. First, the images were classified by whether they contained the edge or not. Since the CO_2 ice has a significantly different temperature than that of nonice cap land, a histogram of the temperatures was used. If a histogram of the temperatures recorded in the image contained two peaks, one represented the temperatures of the ice cap and the other the temperatures of the land, indicating that the edge of the ice cap was contained in the image. If the histogram contained only one peak, the image contained only ice or nonfrozen land [11].

Once the images were classified, the images containing the edge were processed to find the location of the edge. The ice caps do not actually have a distinct edge; rather, the ice gets thinner and thinner, then there is CO_2 frost, then just patches of frost. The scientists declared the point at which less than half the ground had frozen CO_2 to be the edge of the ice cap. The Bimodal Image Temperature (BIT) histogram analysis algorithm was created to find the edge of the ice cap in an image. The algorithm uses the temperature histogram of the image to identify the temperature of the ice cap edge. The minimum point of the dip between the ice cap peak and the land peak represents the temperature of the edge of the ice cap. The algorithm goes through the image and marks each pixel as less than that temperature (ice cap) or greater (nonice land). The northernmost line in the image where less than half of the pixels are marked as ice cap is then declared to be the edge of the ice cap. This location is noted [10].

This algorithm runs in linear time, and it has the advantage of not requiring the data to be calibrated before processing. This is significant, since the data onboard the satellite is not calibrated, and calibrating it would require considerably more processing resources [11]. Another advantage is that once the location of the edge of the ice cap is known, if there is not enough bandwidth to downlink the entirety of the images containing it, just the location data can be transmitted to earth [10].

One unexpected result of these efforts is the discovery of an additional band of water ice sometimes present at the edge of the polar ice cap. This band appears as

Fig. 4 Mars Odyssey's mission is to map chemical elements and minerals on the surface of Mars, look for water in the shallow subsurface, and analyze the radiation environment to determine its potential effects on human health (Image Courtesy of NASA)

a third peak in the temperature histogram of some images, too warm to be CO_2 ice but too cold to be nonice land. It grows wider as the CO_2 ice recedes in the spring. Prior to this work, its existence was only guessed at [11].

As with running ASE on EO-1, this effort faced major constraints. The processor aboard the Odyssey runs at 20 MHz, and only of 20% capacity, and 40 MB of memory was allocated for on-board data analysis. With those processing resources, the algorithm had to be fast enough to keep up with the data collection, which occurs at 9,600 pixels per second [11].

4.4 Mars Rover

Like the previous satellite projects, the Mars rovers face constraints of limited downlink capacity and major delays, making full control from Earth nearly impossible. The traditional approach of manually selecting targets for sampling based on the previous (Martian) day's images, or guessing and blindly directing the rover toward suspected targets, was fraught with delays and inaccuracies [8]. Meanwhile, as rover guidance and distance measurement systems improved, the rovers were able to travel farther, gathering more data, but with no increase in bandwidth to downlink that data [9].

Fig. 5 The Mars Rover mission has been wildly successful in exploring the Martian surface (Image Courtesy of NASA)

The On-board Autonomous Science Investigation System (OASIS) was developed to allow each rover to analyze data on-board, identifying items of interest in its images and retargeting itself as necessary for further study. OASIS manages geologic data on-board the rover, identifies terrain features and selects targets of interest, and plans and schedules rover movement using the same CASPER system used on EO-1 [9].

The Mars rover mainly seeks to identify different types of rocks in the landscape. To do this, feature detection is used. First, an image is split into ground vs. sky. If the sky is in the image, edge detection is performed to find the horizon, seed areas of low variance are selected, and the sky is identified by filling in from those seeds [9].

Once sky is identified, clouds are detected by finding areas of high variance in the sky. For rock detecting, the sky is masked out. The remaining image is normalized, smoothed, and the edges are detected. Rocks are identified by using an edge walker to find closed shapes in the image [9].

Once the rocks in the image have been identified, the algorithm picks specific points on individual rocks as targets [8]. Rock properties such as albedo, shape, size, and texture are estimated and used to prioritize the rocks for further study, since these characteristics indicate the type of rock. Unusual rocks are marked with a higher priority, but care is taken to make sure representative rocks are also included [9].

Once this processing has occurred, the algorithm can flag images as useful or not. For example, images with no clouds are of no use to scientists who are studying clouds and should not be sent to earth unless they contain something else of

interest. Once the targets have been identified, new goals are generated and sent to CASPER, which creates a new plan for the rover, studying as many targets as possible according to priority [9].

Future expansions to this research include more sophisticated rock property gauging, recognizing more features, being able to recognize the same target from different angles and distances, and being able to identify larger geographical boundaries such as hills, plains, and river channels. However, even without those improvements, this effort has benefited the rover program by enabling a rover to identify new targets on the fly, examine interesting things as it discovers them if resources are available, and prioritize data for downlink to earth, thus maximizing scientific benefit from the resources available [9]. Clearly, adding on-board data mining capabilities to deep space missions is the next step.

4.5 Deep Space

As space exploration moves farther from Earth, the large latencies, high error rates, and limited communication opportunities involved mean more potential benefit from performing data mining on-board the instrument or spacecraft [35]. Future deep space missions will include autonomous operations, including on-board fault-detection and repair as well as on-board data mining [36]. Intelligent scheduling, data prioritizing, and on-board data mining will be crucial to the maximization of resources in space [35].

Autonomy is already being pursued in deep space exploration. In 1998, Deep Space 1 (DS1) was launched for the purpose of testing the viability of new, high-risk technologies. It was the first deep space probe to use an autonomous on-board navigation system, called AutoNav. It was given a baseline trajectory, plus a set of information about the locations of asteroids, planets, stars, and the intended targets for DS-1. As DS-1 separated from the launch vehicle, AutoNav worked with the Attitude Control System (ACS) to determine the craft's attitude based on the positions of the stars, and move the craft as needed. This included turning the spacecraft to be in optimal position to catch the sun for power generation [35].

To perform these navigation functions, the AutoNav system relies on images gathered from on-board cameras. The AutoNav system plans which upcoming images it requires, and works with the ACS system to position the craft so that those images can be captured. The images are processed, and the necessary information is extracted to compute the current location and attitude of the craft. The AutoNav system is then able to calculate the track from the current position to the next and make an assessment of the craft's progress. If necessary, AutoNav works with ACS to perform a course correction [38].

In testing, the AutoNav system was able to get the spacecraft within 2.5 km of its target locations and arrive within 5 s of its scheduled times – which is fairly impressive for an autonomous system with all of space to run around in. This is all the more impressive because there were considerable problems with the CCD

camera on which the system relied. Furthermore, this type of navigation combined with continuous low-thrust propulsion had never been attempted before, either automatically or manually [38]. After some ground-based parameter tuning, AutoNav's accuracy was further improved [40].

After the initial tests ended, the AutoNav system continued to control DS1, including maneuvering the craft for an encounter with comet Borrelly. Again, AutoNav performed successfully, maneuvering the craft within 2,200 km of the comet despite the failure of another major component of the ACS, the Stellar Reference Unit (SRU) [39]. As a result of its navigational success, DS1 was able to send back the best images and science data ever collected from a comet [41].

DS1 also used an autonomous remote agent to plan and update the spacecraft's schedule based on the mission goals and the state of the craft. This system performed well in tests, including working around simulated faults [40], proving the viability of such autonomous navigation capabilities.

By incorporating these autonomous systems into DS1, NASA was able to reduce costs considerably. The ground operations team averaged only 50 full-time equivalent personnel, resulting in considerable cost savings over the larger team that would be required for a more hands-on mission [40]. The total cost for the DS1 mission was under $150 million [41] – relatively inexpensive in the world of space craft.

As an additional note, autonomous spacecraft can also be supported by autonomous ground stations. The Deep Space Terminal (DS-T) was created as a terminal in the Deep Space Network (DSN) to monitor instruments in space without human intervention. In 1998, DS-T successfully demonstrated its intended function by autonomously receiving, processing, recording, and distributing data from a probe orbiting Mars to the rest of the network [37].

5 Unmanned Vehicles

Like autonomous vehicles in space, unmanned vehicles, whether on the ground (UGVs), in the air (UAVs), or under water (AUVs), represent a great opportunity to take advantage of on-board data mining. For these vehicles to make the leap from remotely controlled devices to autonomous objects, they need to be able to perform their own planning, scheduling, and control tasks. One approach for accomplishing this, suggested by Tim Grant [46], follows the Observe–Orient-Decide–Act model (sometimes referred to as the OODA-Loop). The OODA-Loop was originally developed for use by fighter pilots to quickly make decisions and act on those decisions [46] and has been used successfully in a number of AI-based systems. In addition, complete path planning algorithms have been developed to find optimal paths using two different approaches: probabilistic roadmaps (RPM) and rapidly exploring random trees (RRT). Since, in the case of some UAVs, the vehicles in question would be flying in the same airspace as commercial airliners, collision avoidance is crucial. The systems also need to be able to replan based on new information, such as obstacles or updated no-fly zones. These approaches performed well in a test flight

involving fully deployed UAVs [48]. Similarly, navigation systems for unmanned ground vehicles (UGVs) have been proposed, using GPS and compass readings for location and orientation [49].

For small devices, movement similar to that of birds or insects is often useful. This behavior is called swarming. Basically, each vehicle or sensor moves based on simple rules, but as they interact, an emergent intelligent behavior seems to occur, as seen with ants looking for food – each ant's behavior may seem random, but taken together the collective hive behaves in an intelligent manner. As such algorithms are optimized for parallel processing, they can be very useful for determining UAV behavior, especially when there are many small, unsophisticated UAVs working together [45].

Another approach is to use fuzzy logic for planning and control of UAVs. By employing fuzzy decision trees, a plan can be developed that accounts for factors including risk to the vehicle, fuel consumption, other costs, and mission goals and priorities, while determining the optimal trajectory for each UAV. Using fuzzy decision trees, the UAVs can also collaborate automatically without human intervention, again taking into account each vehicle's goals, safety, and priorities [43].

UAVs are becoming increasingly important to the military, especially for surveillance missions. In particular, urban surveillance presents challenges of seeing around corners and picking out the relevant information from a busy image. One application in which on-board data mining can be particularly advantageous is in target tracking. In this case, a UAV can go into a dangerous environment and pursue a target such as a suspected terrorist or a vehicle. To accomplish this, the UAV needs to be able to identify the target from the images it collects, track its movement though a series of temporally separate images, and navigate to follow the same path. This is not an easy task, but must be performed on-board. Bandwidth constraints limit the ability to transmit the images for processing elsewhere and to send feedback to the vehicle – the delays involved would be unacceptable. This is one application where the advances in on-board data mining result in clear, immediate benefits to the safety of humans – expendable and autonomous UAVs are sent into harm's way and convey results back to humans located in relative safety [50].

In one approach addressing this type of application, Support Vector Regression was used to establish the location of a stationary enemy radar station. This was done by tracking the radar pulse as received by three UAVs flying together in a triangular formation and using that information to calculate the location of the source of the signal. This work represents a first step toward a scalable solution that would track stationary and moving targets using multiple mobile platforms [51].

UAVs are also coming into wider use by civilian agencies, including the department of transportation. They provide a lower cost alternative to manned vehicles while allowing faster and more flexible deployment compared to fixed cameras mounted along highways. For remote areas, UAVs may be the only cost-effective option for traffic monitoring. Beyond simply transmitting video of traffic flow to ground sites for processing, UAVs with on-board data mining capabilities have the potential to recognize accidents or emergencies and alert emergency personnel with images and location information [44].

There are many opportunities for applying on-board data mining in underwater applications as well. In one project, autonomous underwater vehicles (AUVs) were used to study the sea floor, looking for bacterial mats living near offshore mud volcanoes or areas rich in organic matter. As multiple AUVs worked together, their images could be combined to produce video mosaics and georeferenced images showing the areas explored. Scientists on one such project developed the IBU software to automatically analyze the images, a process that had been completely manual before [52].

As this project produced large volumes of data, thousands of images per campaign, on-board processing was used to distinguish the relevant parts of an image from the background and send only the relevant parts on for further processing on the surface. After performing manual and automated analysis of a set of 2,840 video images and comparing the automated results with the manually identified images, scientists found that the automated analysis software exhibited better than 90% precision. These positive results can be extended to move more of the processing to the UAVs, allowing them to become more autonomous and make decisions based on their analysis of images as they are captured [52].

New unmanned underwater vehicles are being developed that can autonomously control themselves and collaborate with each other while floating freely with the ocean currents. By moving with the currents rather than being stationary or self-propelled, these vehicles can observe the ocean life in its natural state without disturbing it. This sort of device is also uniquely suited to tracking such things as oil spill spread, pollution dispersion, and plankton community evolution. In one test case, individual prototype devices have been deployed successfully. The eventual goal is to release many of these devices together and allow them to interact through an acoustic network. Since these devices are designed to travel without human control and the range of underwater modems is only a few miles, bandwidth will be at a premium [47]. Any on-board data mining that these devices can do will be very valuable in reducing the communication requirements.

6 Biometrics

In addition to enabling instruments in hostile or distant environments to perform autonomously, on-board data mining technology is also enabling advances in biometric systems. Biometric systems are one or more sensors used to monitor some facet of the human body. Such sensors may be placed in well connected areas – for example, the fingerprint systems now in use at many airports. But others either are placed in a covert way to surreptitiously monitor a location or are portable systems meant for use by individuals to monitor their health. Thus, many of the usual problems and approaches to on-board mining come into play with these systems.

As an example, computer-based authentication systems are commonplace – key cards, username/password systems, etc. Many such systems currently use knowledge-based or token-based authentication: authentication based on something

you know (such as a password or PIN) or something you have (such as a card or a key). However, these systems are prone to unauthorized access if someone obtains the password or steals the card or key. Biometrics offers the possibility for unique and secure identification of each and every person. There are two primary types of biometric data: static, involving a physical attribute such as fingerprints or an iris scan; and dynamic, involving behavior, such as keystroke dynamics, signature dynamics, or even a person's gait [21].

To date, most single-source biometric systems have relatively poor performance; however, if multiple sources of biometric data are combined, biometric systems can approach near-flawless performance. For example, a system may check the user's fingerprint, gait, and voice for identification. By using more than one attribute, the user may still be authenticated if the reading of one attribute is corrupted in some way. For example, if the user has a cold, voice recognition may be impacted. Similarly, dust on the lens or unfavorable lighting may interfere with facial recognition. Using multiple attributes allows a positive identification to be made despite such problems. However, it results in a high volume of heterogeneous data to process. One method that has been found effective in such a case is Bayesian Model Averaging with Decision Trees and a sweeping strategy. This method is particularly effective in situations where risk evaluation is crucial, such as at border checkpoints and transportation hubs [21].

Even when using just one biometric attribute, data volume can still be an issue, and identification is subject to problems such as lighting variation. For example, although the use of fingerprints for identification is one of the most mature and proven biometrics, fingerprint recognition and classification are still a difficult problem. To improve accuracy and reduce processing time, one approach is to normalize the images, extract features, classify the image, and then recognize the fingerprint based on its classification. Normalization includes adjusting all images to the same darkness/brightness, and adjusting the orientation. Images can also be enhanced by thinning and increasing contrast. Then a line detector can be used for feature extraction of uniquely identifying characteristics, in this case detection of ridges and furrows. A given pattern of ridges and furrows is classified as a whorl, arch, tent, etc., using a neural network approach. Finally, using a crisp k nearest neighbor algorithm, the image is matched against selected images in the database based on the classification. This saves considerable processing time in not having to search the entire database. In tests, this approach achieved a 97.4% recognition rate [22].

Similarly, face recognition is subject to difficulties resulting from cropping, differences in pose angle, and differences in lighting. Several approaches have been used, including Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and eigenfaces, an approach using feature vectors and the Euclidean distance between them. However, all of these methods are very sensitive to outliers in the data, specifically "noisy" images with lighting, cropping, or pose angle different from the images in the database, making their use in convert situations difficult. However, one study found that by implementing an automated system to filter out images whose feature vectors represent outliers, that is, noisy images, the recognition rate of a face recognition system could be improved by 10–20% [26].

These incremental types of improvements become crucial in on-board applications, where processing time, power, and network connectivity may all be limited. For example, in border control systems, the person must be identified accurately and processed expediently. Considerable discussion has occurred regarding including biometric data in passports. In 2002, the International Civil Aviation Organization (ICAO) put forth standards for future passports, including the use of smart cards for storing biometric data [31]. The benefits of more reliable identification for border control and security purposes are obvious. However, this plan is not without obstacles. To develop and maintain a database of biometric data, fingerprints and faces for example, for travelers worldwide is a large and costly endeavor. Identifying a traveler from such a large database in a timely fashion is another problem. Also, the cost of issuing passports with secure smart card technology must be considered, as well as the security of the data once in the smart card and in the database, and user resistance to giving biometric data. The security of the system is of utmost importance: if the biometrics data can be stolen or manipulated, the entire effort may do more harm than good. In 2005, an early Dutch biometric passport was cracked in 2 h using a PC, so this is a real issue [29]. However, some progress has been made. Visitors to the U.S. from countries participating in a visa waiver program have fingerprint scans and a digital photograph collected at the time a visa is issued. At that time, these biometrics are compared against a database of criminals and terrorist suspects. If the visitor is not among the banned, the visa is issued. The biometric data is then used to verify the visitor's identity on entry and exit into the US. [32] [33].

In the commercial world, there are a number of examples of biometrics systems that have been deployed successfully. IBM Thinkpads and Korean LG mobile phones have both incorporated fingerprint recognition for user authentication. Also, in a pilot study, the Port of Palm Beach, Florida, used photographs and fingerprints to identify and track visitors, and to keep out banned visitors. Furthermore, the Pinellas County Sheriff's Office in Florida implemented a face recognition system using digital cameras and docking stations in patrol cars. In less than 1 year, 37 identifications were made leading to arrests. None of these identifications would have been made without the on-board systems in the patrol cars [31].

A more unusual application for on-board data mining with biometrics is used for mood analysis. The Mood Phone analyzes acoustic information in speech and ascertains the speaker's mood. This information is conveyed in real time to the listener by a color coded light that also varies in intensity based on the intensity of the speaker's mood. This information can be very useful to people who are impaired in detecting others' moods themselves, including people with autism or Asperger's syndrome. Another emotion detecting application uses a glove with sensors to monitor skin temperature, conductivity, and heart rate. This system can estimate the user's emotional state with up to 75% accuracy. One possible application for this technology is for use with students, to detect when they become frustrated with a training scenario and need personal attention [23].

Biometrics such as command use frequency and habits based on time of day can be used to detect intrusion and thwart an attack to a computer network. One study used these factors to achieve a 90% recognition rate for users of a university system

for users who had executed at least 50 commands [24]. Haptics, including pressure and torque when using a stylus and feedback-generating pad, have also been shown to uniquely identify users [25]. Even simpler attributes such as typing style and idle time patterns can be used to identify a user. For example, in a system developed for online gaming to stop account theft and sharing one account among several people, the patterns of idle time between commands were found to uniquely identify users. In this study, activity logs for 287 users of the game Angel's Love were analyzed. Any users who logged fewer than 200 min during the study time were eliminated. For the remaining users, there was a high correlation between active time (intervals where the character moved continually, with no pauses of a second or more) and idle time (intervals of 1 s to 30 min when the character did not move). The idle time distribution had greater variation among users, varying in central tendency and overall shape, whereas the active time varied considerably less among users, so the idle time was used as the indicator. In this study, when the length of playing history was fixed to 200 min, user detection could be performed in 20 min with over 90% accuracy. The addition of other factors, such as movement patterns in the game and mouse/keyboard behavior, could speed up the detection process [28].

Keyboard behavior, specifically typing style, has been shown to be effective in user authentication and requires no special hardware [30]. Using the delays between pairs of keystrokes, users can be identified uniquely when entering a user ID and password [20, 30]. In one study, only eight rules were required to identify the legitimate users with over 95% accuracy. Specifically, the first few and last pairs of characters were enough to make a correct classification – the entire user ID and password do not need to be stored or analyzed. Also, the legitimate user tends to type his or her login ID faster than any other person [20].

On-board data mining has life-saving potential when applied to health monitoring and alert systems. Sets of sensors that are worn continually allow for a better understanding of the patient's baseline state as well as immediate notification in the event of an anomaly such as a heart attack or stroke. Traditional in-home health monitoring systems were awkward and cumbersome, relying heavily on wires for communication. They had short memory banks of 24 h or less and no processing capabilities on their own. Instead, systems are being developed that can store and process several weeks of data while withstanding the conditions of ordinary life, including temperature extremes, vigorous activity, and sleep. The sensors are smaller, less obtrusive, noninvasive, and nonirritating to the skin. By combining these sensors with intelligent processing and on-board data mining software, a significant improvement is realized. Health deterioration can be detected earlier and health care providers notified, while reducing the need for in-person medical appointments and monitoring. Furthermore, these advanced systems can provide a valuable bank of detailed information about the patient's condition to the medical practitioners, much more so than with traditional monitoring systems or occasional office visits [55].

Systems have been designed to accomplish this task. In one proposed system, a Cyberjacket, a modular wearable computing platform, is augmented with an ECG, and oxygen saturation monitor, and a temperature sensor. The Cyberjacket already includes a GPS device and accelerometers to detect location and motion.

By combining these components, a reasonably unobtrusive and mobile system can be created, enabling health conditions to be monitored in the patient's everyday life and referenced with context information including time of day and location [54].

A similar system has been proposed using a Smart Shirt with sensors for heart rate, body temperature, and blood pressure, as well as a GPS device to record location. Data from the sensors would be fed into a PDA using Bluetooth. Then the PDA would display signal data and perform quick analysis on the data. In the event of an anomaly or emergency, the PDA would alert medical personnel, using location information provided by the GPS to direct them to the patient. Under normal circumstances, the PDA would also feed the data to a larger server, which would perform further data mining using association rules and classification [53].

A third system, the Wearable Wireless Body/Personal Area Network (WWBAN) has been implemented and tested. This system involves a three-tier network: the wearable sensors send data to a PDA using a short-range wireless network, the PDA passes the data on to a home or central server using wireless LAN protocols, and then the home server can periodically update the patient's record at the hospital or doctor's office using a secure internet connection. The PDA includes personal server (PS) software, providing an interface to display the data to the user while monitoring the user's health and providing feedback, including alerts in the event of an anomalous condition. Using data from the motion sensors, the PS can discriminate the general activity of the user (sitting, walking, running, lying down) and incorporate that information with the health metrics provided by the sensors and the patient's history to give a more complete picture of the patient's condition and status [57,60].

These systems and others like them have great potential for many health applications in addition to ongoing monitoring of vital signs. Another system, called LiveNet, uses classifiers to distinguish among activities such as walking, sitting, and biking, as well as more subtle activities such as head-nodding or shaking. The LiveNet system also incorporates voice processing to detect stress and emotional arousal. In addition, the system can detect shivering with up to 95% accuracy using real-time classifier systems based on Gaussian Mixture Models. One application for this is in monitoring soldiers working in harsh climates and determining their hypothermia risk. LiveNet is also in pilot testing for monitoring Parkinson's disease symptoms, epilepsy seizures, and even depression. The continual monitoring and analysis provided by the system has great potential in helping doctors tailor treatments based on individual patients' true symptoms and experiences, rather than heuristics and average dosages [58].

Data from wearable sensors can also provide insight into stroke patients' recovery and rehabilitation. By using wearable sensors to monitor stroke patients' activities, therapy can be better tailored to their needs. During a test of the system, patients performed tasks including lifting a pencil, turning a key, flipping a card, and other tasks often used to assess a stroke patient's level of impairment. Using algorithms from the data mining toolkit Waikato Environment for Knowledge Analysis (WEKA) and linear regression models, features from each task were analyzed, and models were built to predict the patient's scores on clinical tests, getting within 10% of the average clinical score [59].

7 Sensor Networks

As the field of on-board data mining advances, integrating the data from multiple sensors and getting them to communicate and work together is the next step. Deployment of multiple networked sensors is becoming a more and more common method to monitor large geographic areas. This is due in part to the declining costs of such sensors, but also to the improvements in both wired and wireless network connections. Such networking is the cornerstone of sensor networks. In order for the system to derive the maximum advantage from the gathered information, the components in the network must be able to communicate their observations to each other.

NASA has been at the forefront of such networks for some time. However, it is the military and homeland security interests that are increasingly pushing the technologies in this area. For example, the Defense Intelligence Agency's 5-year plan for leveraging basic research describes a sensor network consisting of intelligent and autonomous space-borne, airborne, and ground-based sensors [61]. These sensors will act independently of one another, yet each will be capable of both publishing and receiving sensor information, observations, and alerts among other sensors in the network. Furthermore, these sensors will be capable of acting on alerts and information received from other sensors in the network, perhaps altering acquisition properties of their instruments, changing the location of their platform, or updating processing strategies for their own observations to provide responsive information or additional alerts.

Such autonomous and intelligent sensor networking capabilities provide significant benefits for collections of heterogeneous sensors within any environment, including those used for treaty verification, covert monitoring, and changing battlespaces, but they are crucial for multi-sensor observations and surveillance. For these applications, real-time communication with external components and users may be inhibited, and the environment may be hostile.

In all environments, mission automation and communication capabilities among disparate sensors will enable quicker response to interesting, rare, or unexpected events, especially important in time-critical situations such as chemical detection, missile firings, or battlefield situations. Another advantage of an intelligent network of heterogeneous sensors is that all of the sensors can benefit from the unique capabilities of each sensor in the network. There are a number of efforts underway to manage real-time sensor data. Some areas of focus include data compression and reduction, sensor fusion, real-time data mining, and other operational level algorithms (e.g. [62–66]). Many of these approaches deal quite effectively with streaming data in resource-constrained environments such as sensor networks.

NASA and the National Science Foundation have a number of sensor network efforts currently in operation and a number that are in the research and development phases. Some of these are aimed at space-based applications, but many are ground-based systems geared toward environmental, weather, biological, and health monitoring. For example, several sensor networks have been deployed to monitor volcanic activity around the world, including some on the US mainland

as well as in Hawaii and Alaska. Most of these systems are comprised of sensors deployed around the perimeter of known active volcanoes, and are diligently measuring for any change in activity (e.g. [67]). Some of these networks are also linked with satellite data such as NASA's Moderate Resolution Imaging Spectroradiometer (MODIS), which takes thermal measurements on a global basis. For the people living and working near such geologic features, any change that may signal an imminent eruption is obviously of great interest. Most of these networks include some level of data processing within the network itself, including limited amounts of data mining.

Another area of intense research is in the monitoring of coastal areas. Applications include border protection efforts, using networks of cameras to monitor ports, and even tsunami warning systems that are being deployed after the devastating Indian Ocean tsunami in 2004. The use of sensor networks for environmental monitoring is becoming more widespread as well. All of these applications represent areas where on-board data mining provides added value by enabling information gathering that was previously infeasible or by supporting the optimized use of resources.

8 Conclusion

On-board data mining represents a powerful technology with the potential to improve the efficacy and efficiency of a range of applications. Great strides have already been made in the areas of biometrics, autonomous and unmanned vehicles, and other areas, resulting in new exploration, better data availability to scientists, cost savings, and improvements in security and medicine. As technology improves, harnessing the power of on-board data mining will become increasingly feasible. It is likely that you will be seeing its use in applications in your own work and even your own home and vehicles. This will be a field to watch for exciting new developments.

References

1. Organization web site including all standards documents, http://www.opengeospatial.org/ogc
2. SensorML, http://vast.uah.edu/SensorML/
3. S. Chien, T. Debban, C. Yen, R. Sherwood, R. Castano, B. Cichy, A. Davies, M. Burl, A. Fukunaga, Revolutionary deep space science missions enabled by onboard autonomy. International Symposium on AI, Robotics, and Automation in Space, 2003
4. D. Tran, S. Chien, R. Sherwood, R. Castano, B. Cichy, A. Davies, G. Rabideau, The autonomous sciencecraft experiment onboard the EO-1 spacecraft. AAMAS, 2004
5. G. Rabideau, D. Tran, S. Chien, B. Cichy, R. Sherwood, Mission operations of earth observing-1 with onboard autonomy. Proceedings 2nd IEEE International Conference on Space Mission Challenges for Information Technology, 2006

6. R. Castano, N. Tang, T. Doggett, S. Chien, D. Mazzoni, R. Greeley, B. Cichy, A. Davies, Onboard classifiers for science event detection on a remote sensing spacecraft. Proc. 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006, pp. 845–851
7. NASA: Earth Observing-1, http://eo1.gsfc.nasa.gov/new/general/index.html. Accessed 1 Feb 2008
8. R. Castano, T. Estlin, D. Gaines, A. Castano, B. Bornstein, C. Chouinard, R.C. Anderson, M. Judd, Automated target selection for opportunistic rover science. 37th Lunar and Planetary Science Conference, 2006
9. R. Castano, T. Estlin, R. Anderson, D. Gaines, A. Castano, OASIS: onboard autonomous science investigation system for opportunistic rover science. J. field robot. **24**(5), 379–397 (2007)
10. K. Wagstaff, R. Castano, S. Chien, A. Ivanov, E. Pounders, T. Titus, An onboard data analysis method to track the seasonal polar caps on Mars. International Symposium on AI, Robots, and Automation in Space, 2005
11. R. Castano, K. Wagstaff, S. Chien, T. Stough, B. Tang, On-board analysis of uncalibrated data for a spacecraft at Mars. 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007, pp. 922–930
12. K. Wagstaff, J. Bandfield, R. Castano, S. Chien, M. Smith, Dust storms and water ice clouds: feature detection for use onboard THEMIS. 37th Lunar and Planetary Science Conference, 2006
13. Y. Cai, Y. Hu, Onboard inverse physics from sensor web. 2nd IEEE International Conference on Space Mission Challenges for Information Technology, 2006
14. T. Meindl, W. Moniaci, D. Gallesio, E. Pasero, Embedded Hardware architecture for statistical rain forecast. Res. Microelectron. Electron. **1**, 133–136 (2005)
15. J.A. Kalomiros, J. Lygouras, Design and evaluation of a hardware/software FPGA-based system for fast image processing. Microprocess. Microsyst. (2008). doi:10.1016/j.micpro.2007.09.001
16. M. Lorenz, L. Mengibar, E. SanMillan, L. Entrena, Low power data processing system with self-reconfigurable architecture. J. Syst. Arch. **53**(9), 568–576 (2007)
17. P. Pingree, J.-F. Blavier, G. Toon, D. Bekker, An FPGA/SoC approach to on-board data processing enabling new mars science with smart payloads. IEEE Aerospace Conference, 2007, pp. 1–12
18. Z. Baker, V. Prasanna, An architecture for efficient hardware data mining using reconfigurable computing systems. 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2006, pp. 67–75
19. Z. Baker, V. Prasanna, Efficient hardware data mining with the apriori algorithm on FPGAs. 13th, Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2005, pp. 3–12
20. K. Revett, S.T. de Magalhaes, H. Santos, Data mining a keystroke dynamics based biometrics database using rough sets. Portuguese conference on Artificial Intelligence, 2005, pp. 188–191
21. C. Maple, V. Schetinin, Using a Bayesian averaging model for estimating the reliability of decisions in multimodal biometrics. First International Conference on Availability, Reliability, and Security, 2006
22. K. Umamaheswari, S. Sumathi, S.N. Sivanandam, K.K.N. Anburajan, Efficient finger print image classification and recognition using neural network data mining. International Conference on Signal Processing, Communications, and Networking, 2007, pp. 426–432
23. J. Krikke, B. Alfonsi. "In the news." IEEE Intelligent Systems, vol. 21, issue 3, Jan.-Feb. 2006, pp. 102–104
24. H.Z. Bing, V.P. Shirochin , S. Jun, An intelligent lightweight intrusion detection system (IDS). IEEE Region 10 TENCON, 2005, pp. 1–7
25. R. Iglesias, M. Orozco, J. Valdes, A. El Saddik, Behavioral features for different haptic-based biometric tasks. IEEE International Workshop on Haptic, Audio, and Visual Environments and Games, 2007, pp. 102–106

26. S. Berrani, C. Garcia, On the impact of outliers on high-dimensional data analysis methods for face recognition. Proc. 2nd International Workshop on Computer Vision Meets Databases, 2005, vol. 160, pp. 43–49
27. R.B. Rao, J. Bi, G. Fung, M. Salganicoff, N. Obuchowski, D. Naidich, LungCAD: a clinically approved, machine learning system for lung cancer detection. Proc. 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007, pp. 1033–1037
28. K. Chen, L. Hong, User identification based on game-play activity patterns. Proceedings of 6th ACM SIGCOMM Workshop on Network and System Support for Games, 2007, pp. 7–12
29. T. Kwon, H. Moon, Biometric authentication for border control applications. Trans. Knowl. Data Eng. doi:10.1109/TKDE.2007.190716
30. A. Guven, I. Sogukpinar, Understanding users' keystroke patterns for computer access security. Comput. Secur. **22**(8), 695–706 (2003)
31. Biometrics enter mobile world, Biomet. Technol. Today **13**(8), 10–11 (2005)
32. U.S. Department of Homeland Security, US-VISIT: How it works. Accessed 22 Feb 2008
33. U.S. Department of Homeland Security. US-VISIT biometric exit process to improve national security. Accessed 22 Feb 2008
34. T.A. Carper, M.D. Gardiner, Biometric Embedded Device. U.S. Patent 20080040615, filed 2007
35. R. Slywczak, Developing autonomous missions through intelligent on-board architectures. Proceedings of IEEE Conference on Networking, Sensing and Control, 2005, pp. 359–364
36. L. Alkalai, Perspectives on dependable computing for solar system exploration. 2002 Pacific Rim International Symposium on Dependable Computing, 2002
37. L. Paal, N. Golshan, F. Fisher, E. Law, W. Veruttipong, M. Stockett' Deep space terminal demonstration. The Telecommunications and Mission Operations Progress Report, TMO PR 42-138, April-June 1999
38. J.E. Reidel, S. Bhaskaran, S. Desai, D. Han, B. Kennedy, T. McElrath, G.W. Null, M. Ryne, S.P. Synnott, T.C. Want, R.A. Werner, Using autonomous navigation for interplanetary missions: The validation of deep space 1 AutoNav. International Conference on Low-Cost Planetary Missions, 2000
39. S. Collins, Deep space 1 flight experience: adventures on an ion drive. 25th Annual AAS Guidance and Control Conference, 2002
40. M.D. Rayman, P. Varghese, D.H. Lehman, L.L. Livesay, Results from the deep space 1 technology validation mission. Acta Astronaut. **47**(2–9) , 475–487 (2000)
41. NASA, Deep Space 1, http://nmp.jpl.nasa.gov/ds1/. Accessed 3 Mar 2007
42. NASA, Deep Space 1: Quick Facts. Accessed 3 March 2007
43. J.F. Smith III, T.H. Nguyen, Fuzzy decision trees for planning and autonomous control of a coordinated team of UAVs. Proceedings of SPIE 6567, 656708, 2007
44. S. Srinivasan, H. Latchman, J. Shea, T. Wong, J. McNair, Airborne traffic surveillance systems: video surveillance of highway traffic. In Proceedings of the ACM 2nd International Workshop on Video Surveillance & Amp; Sensor Networks, New York, NY, USA, October 15, 2004. VSSN '04 (ACM, New York, 2004), pp. 131–135
45. J.J. Corner, G.B. Lamont, Parallel simulation of UAV swarm scenarios. In Proceedings of the 36th Conference on Winter Simulation, Washington, DC, December 05 – 08, 2004. Winter Simulation Conference. Winter Simulation Conference, pp. 355–363, 2004
46. T. Grant, 2005. Unifying planning and control using an OODA-based architecture. In Proceedings of the 2005 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries (White River, South Africa, September 20–22, 2005). ACM International Conference Proceeding Series, vol. 150. South African Institute for Computer Scientists and Information Technologists, pp. 159–170

47. J. Jaffe, C. Schurgers, Sensor networks of freely drifting autonomous underwater explorers. In Proceedings of the 1st ACM International Workshop on Underwater Networks, Los Angeles, CA, USA, Sept 25–25, 2006. WUWNet '06. ACM, (New York, NY, 2006), pp. 93–96
48. M. Wzorek, P. Doherty, Reconfigurable path planning for an autonomous unmanned aerial vehicle. Hybrid Information Technology, 2006. ICHIT'06. Vol 2. International Conference on. Volume 2, Nov. 2006, pp. 242–249
49. B.-J. Yoon, M.-W. Park, J.-H. Kim, UGV(unmanned ground vehicle) navigation method using GPS and compass. SICE-ICASE, 2006. International Joint Conference, Oct. 2006, pp. 3621–3625
50. B. Ludington, J. Reimann, G. Vachtsevanos, I. Barlas, Target tracking with unmanned aerial vehicles: From single to swarm vehicle autonomy and intelligence. Control and automation, 2006. MED '06. 14th Mediterranean Conference, June 2006, pp.1–6
51. B. Sundaram, M. Palaniswami, S. Reddy, M. Sinickas, Radar localization with multiple unmanned aerial vehicles using support vector regression. Intelligent sensing and information processing, 2005. ICISIP 2005. Third International Conference , 14-17 Dec 2005, pp. 232–237
52. K. Jerosch, A. Ldtke, M. Schlter, G.T. Ioannidis, Automatic content-based analysis of georeferenced image data: Detection of Beggiatoa mats in seafloor video mosaics from the Hkon Mosby Mud Volcano. Comput. Geosci. **33**(2), 202–218 (2007)
53. P.-T. Cheng, L.-M. Tsai, L.-W. Lu, D.-L. Yang, The design of PDA-based biomedical data processing and analysis for intelligent wearable health monitoring systems. Computer and Information Technology, 2004. CIT '04. The Fourth International Conference, 14–16 Sept 2004, pp. 879–884
54. J. Crowe, B. Hayes-Gill, M. Sumner, C. Barratt, B. Palethorpe, C. Greenhalgh, O. Storz, A. Friday, J. Humble, C. Setchell, C. Randell, H.L. Muller, Modular sensor architecture for unobtrusive routine clinical diagnosis. Distributed Computing Systems Workshops, 2004. Proceedings of 24th International Conference , 2004, pp. 451–454
55. P. Kulkarni, Y. ztrk, Requirements and design spaces of mobile medical care. SIGMOBILE Mob. Comput. Commun. Rev. **11**(3), 12–30 (2007)
56. A. Jaimes, Sit straight (and tell me what I did today): A human posture alarm and activity summarization system. In Proceedings of the 2nd ACM Workshop on Continuous Archival and Retrieval of Personal Experiences (Hilton, Singapore, November 11–11, 2005). CARPE '05. ACM, New York, NY, pp. 23–34
57. A. Milenkovic, C. Otto, E. Jovanov, Wireless sensor networks for personal health monitoring: Issues and an implementation. Computer Communications, Special issue: Wireless Sensor Networks: Performance, Reliability, Security, and Beyond (Elsevier, Amsterdam, 2006)
58. M. Sung, C. Marci, A. Pentland, Wearable feedback systems for rehabilitation. J. NeuroEng. Rehabil. **2**,17 (2005)
59. T. Hester, R. Hughes, D.M. Sherrill, B. Knorr, M. Akay, J. Stein, P. Bonato, Using wearable sensors to measure motor abilities following stroke. Wearable and implantable body sensor networks, 2006. BSN 2006. International Workshop, 3–5 April 2006
60. E. Jovanov, A. Milenkovic, C. Otto, P.C. de Groen, A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation. J. NeuroEng. Rehabil. **2**, 6(2005). Available at: http://www.jneuroengrehab.com/content/2/1/6/
61. L. Meador (Panel Chairman), Leveraging basic research into MASINT capabilities (A five-year plan), Response to Conference Report to H.R. 4546, the National Defense Authorization Act for Fiscal 2003, March 2003
62. Research issues in data stream association rule mining N. Jiang, L. Gruenwald, ACM SIGMOD RECORD **35**, pp. 14–19 (2006)
63. S. Ivengar, S. Sastry, N. Balakrishnan, Foundations of data fusion for automation. IEEE Instrum. Meas. Mag. **6**(I4), 35–41 (2003)
64. L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, R. Motwani, Streaming data algorithms for high quality clustering. Proceedings of the 18th International Conference on Data Engineering, pp. 685–694, 2002
65. L. Weixian, L. Yilong, J. Fu, Data fusion of multiradar system by using genetic algorithm, IEEE Trans. Aerosp. Electron. Syst. **38**(2), 601–612 (2002)

66. M. Halatchev, L. Gruenwald, Estimating missing data in related sensor data streams. International Conference on Management of Data, Jan 2005, pp. 83–94
67. G.W. Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, M. Welsh, Deploying a wireless sensor network on an active volcano, In IEEE Internet Computing, Special issue on data-driven applications in sensor networks, March/April 2006
68. Z.K. Baker, V.K. Prasanna, Efficient Hardware Data Mining with the Apriori Algorithm on FPGAs, fccm, pp. 3–12, 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'05), 2005

Data Streams: An Overview and Scientific Applications

Charu C. Aggarwal

1 Introduction

In recent years, advances in hardware technology have facilitated the ability to collect data continuously. Simple transactions of everyday life such as using a credit card, a phone, or browsing the web lead to automated data storage. Similarly, advances in information technology have lead to large flows of data across IP networks. In many cases, these large volumes of data can be mined for interesting and relevant information in a wide variety of applications. When the volume of the underlying data is very large, it leads to a number of computational and mining challenges:

- With increasing volume of the data, it is no longer possible to process the data efficiently by using multiple passes. Rather, one can process a data item at most once. This leads to constraints on the implementation of the underlying algorithms. Therefore, stream mining algorithms typically need to be designed so that the algorithms work with one pass of the data.
- In most cases, there is an inherent temporal component to the stream mining process. This is because the data may evolve over time. This behavior of data streams is referred to as *temporal locality*. Therefore, a straightforward adaptation of one-pass mining algorithms may not be an effective solution to the task. Stream mining algorithms need to be carefully designed with a clear focus on the evolution of the underlying data.

Another important characteristic of data streams is that they are often mined in a distributed fashion. Furthermore, the individual processors may have limited processing and memory. Examples of such cases include sensor networks, in which it may be desirable to perform in-network processing of data stream with limited

C.C. Aggarwal (✉)
IBM T. J. Watson Research Center, NY, USA
e-mail: charu@us.ibm.com

M.M. Gaber (ed.), *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*, DOI 10.1007/978-3-642-02788-8_14,

processing and memory [1, 2]. This chapter will provide an overview of the key challenges in stream mining algorithms which arise from the unique setup in which these problems are encountered.

This chapter is organized as follows. In the next section, we will discuss the generic challenges that stream mining poses to a variety of data management and data mining problems. The next section also deals with several issues which arise in the context of data stream management. In Sect. 3, we discuss several mining algorithms on the data stream model. Section 4 discusses various scientific applications of data streams. Section 5 discusses the research directions and conclusions.

2 Stream Management Issues

Since data streams are processes which create large volumes of incoming data, they lead to several challenges in both processing the data as well as applying traditional database operations. For example, when the incoming rate of the data streams is higher that that can be processed by the system, techniques are required in order to selectively pick data points from the stream, without losing accuracy. This technique is known as *loadshedding*.

Loadshedding in Data Streams. Since data streams are generated by processes which are extraneous to the stream processing application, it is not possible to control the incoming stream rate. As a result, it is necessary for the system to have the ability to quickly adjust to varying incoming stream processing rates. One particular type of adaptivity is the ability to gracefully degrade performance via "load shedding" (dropping unprocessed tuples to reduce system load) when the demands placed on the system cannot be met in full given available resources. The loadshedding can be tailored to specific kinds of applications such as query processing or data mining. A discussion of several loadshedding techniques are provided in book.

Join Processing in Data Streams. Stream join is a fundamental operation for relating information from different streams. This is especially useful in many applications such as sensor networks in which the streams arriving from different sources may need to be related with one another. In the stream setting, input tuples arrive continuously, and result tuples need to be produced continuously as well. We cannot assume that the input data is already stored or indexed, or that the input rate can be controlled by the query plan. Standard join algorithms that use blocking operations, e.g., sorting, no longer work. Conventional methods for cost estimation and query optimization are also inappropriate, because they assume finite input. Moreover, the long-running nature of stream queries calls for more adaptive processing strategies that can react to changes and fluctuations in data and stream characteristics. The "stateful" nature of stream joins adds another dimension to the challenge. In general, in order to compute the complete result of a stream join, we need to retain all past arrivals as part of the processing state, because a new tuple may join with an arbitrarily old tuple arrived in the past. This problem is exacerbated by unbounded input streams, limited processing resources, and high performance requirements,

as it is impossible in the long run to keep all past history in fast memory. A survey on different join processing techniques in data streams may be found in book.

Indexing Data Streams. The problem of indexing data streams attempts to create a an indexed representation, so that it is possible to efficiently answer different kinds of queries such as aggregation queries or trend based queries. This is especially important in the data stream case because of the huge volume of the underlying data. We note that most traditional indexes require multiple passes to create. We do not have this luxury in the data stream model because of one-pass constraints. A discussion of several models for data stream computation may be found in [3].

Stream Cube Analysis of Multi-dimensional Streams. Much of stream data resides at a multi-dimensional space and at rather low level of abstraction, whereas most analysts are interested in relatively high-level dynamic changes in some combination of dimensions. To discover high-level dynamic and evolving characteristics, one may need to perform multi-level, multi-dimensional on-line analytical processing (OLAP) of stream data. Such necessity calls for the investigation of new architectures that may facilitate on-line analytical processing of multi-dimensional stream data [4, 5]. A **stream_cube** architecture was proposed in [4, 5] that effectively performs on-line partial aggregation of multi-dimensional stream data, captures the essential dynamic and evolving characteristics of data streams, and facilitates fast OLAP on stream data. Stream cube architecture facilitates online analytical processing of stream data. While this is needed for online stream mining and query processing, the process of model creation requires effective management of the incoming data.

3 Stream Mining Algorithms

In this section, we will discuss the key stream mining problems and will discuss the challenges associated with each problem. We will also provide a wide overview of the different directions of research for these problems.

3.1 Data Stream Clustering

Clustering is a widely studied problem in the data mining literature. However, it is more difficult to adapt arbitrary clustering algorithms to data streams because of one-pass constraints on the data set. An interesting adaptation of the k-means algorithm has been discussed in [6] which uses a partitioning based approach on the entire data set. This approach uses an adaptation of a k-means technique in order to create clusters over the entire data stream. However, in practical applications, it is often desirable to be able to examine clusters over user-specified time-horizons. For example, an analyst may desire to examine the behavior of the clusters in the

data stream over the past 1 week, the past 1 month, or the past year. In such cases, it is desirable to store *intermediate cluster statistics*, so that it is possible to leverage these in order to examine the behavior of the underlying data.

One such technique is micro-clustering [7], in which we use cluster feature vectors [8] in order to perform stream clustering. The cluster feature vectors keep track of the first-order and second-order moments of the underlying data in order to perform the clustering. These features satisfy the following critical properties which are relevant to the stream clustering process:

- *Additivity Property*. The statistics such as the first- or second-order moments can be maintained as a simple addition of statistics over data points. This is critical in being able to maintain the statistics efficiently over a fast data stream. Furthermore, additivity also implies subtractivity; thus, it is possible to obtain the statistics over a particular time horizon, by subtracting out the statistics at the beginning of the horizon from the statistics at the end of the horizon.
- *Computational Convenience*. The first and second order statistics can be used to compute a vast array of cluster parameters such as the cluster centroid and radius. This is useful in order to be able to compute important cluster characteristics in real time.

It has been shown in [7], that the micro-cluster technique is much more effective and versatile than the k-means based stream technique discussed in [6]. This broad technique has also been extended to a variety of other kinds of data. Some examples of such data are as follows:

- *High Dimensional Data*. The stream clustering method can also be extended to the concept of projected clustering [9]. A technique for high dimensional projected clustering of data streams is discussed in [10]. In this case, the same micro-cluster statistics are used for maintaining the characteristics of the clusters, except that we also maintain additional information which keeps track of the projected dimensions in each cluster. The projected dimensions can be used in conjunction with the cluster statistics to compute the projected distances which are required for intermediate computations. Another innovation proposed in [10] is the use of decay-based approach for clustering. The idea in the decay-based approach is relevant for the case of evolving data stream model, and is applicable not just to the high dimensional case, but any of the above variants of the micro-cluster model. In this approach, the weight of a data point is defined as $2^{-\lambda \cdot t}$, where t is the current time-instant. Thus, each data point has a half-life of $1/\lambda$, which is the time in which the weight of the data point reduces by a factor of 2. We note that the decay-based approach poses a challenge because the micro-cluster statistics are affected at each clock tick, even if no points arrive from the data stream. In order to deal with this problem, a *lazy approach* is applied to decay-based updates, in which we update the decay-behavior for a micro-cluster only if a data point is added to it. The idea is that as long as we keep track of the last time t_s at which the micro-cluster was updated, we only need to multiply the micro-cluster statistics by $2^{-\lambda(t_c - t_s)}$, where t_c is the current time instant. After multiply the decay statistics by this factor, it is possible to add the micro-cluster

statistics of the current data point. This approach can be used since the statistics of each micro-cluster decay by the same factor in each track, and it is therefore possible to implicitly keep track of the decayed values, as long as a data point is not added to the micro-cluster. In the latter case, the statistics need to be updated explicitly, while other counts can still be maintained implicitly.

- *Uncertain Data*. In many cases, such as in sensor networks, the underlying data may be noisy and uncertain. In such cases, it may be desirable to incorporate the uncertainty into the clustering process. In order to do so, the micro-cluster statistics are appended with the information about the underlying uncertainty in the data. This information can be used in order to make more robust clustering computations. The advantages of using the uncertainty into the clustering process are illustrated in [11].
- *Text and Categorical Data*. A closely related problem is that of text and categorical data. The main difference with the quantitative domain is the nature of the statistics which are stored for clustering purposes. In this case, we maintain the counts of the frequencies of the discrete attributes in each cluster. Furthermore, we also maintain the inter-attribute correlation counts which may be required in a variety of applications. In [12], an efficient algorithm has been proposed for clustering text and categorical data streams. This algorithm also allows for a decay-based approach as in [10].

3.2 Data Stream Classification

The problem of classification is perhaps one of the most widely studied in the context of data stream mining. The problem of classification is made more difficult by the evolution of the underlying data stream. Therefore, effective algorithms need to be designed in order to take temporal locality into account. The concept of stream evolution is sometimes referred to as *concept drift* in the stream classification literature. Some of these algorithms are designed to be purely one-pass adaptations of conventional classification algorithms [13], whereas others (such as the methods in [14, 15]) are more effective in accounting for the evolution of the underlying data stream. The broad methods which are studied for classification in the data stream scenario are as follows:

VFDT Method. The VFDT (Very Fast Decision Trees) method has been adapted to create decision trees which are similar to those constructed by a conventional learner with the use of sampling based approximations. The VFDT method splits a tree using the current best attribute, taking into consideration the fact that the number of examples used are sufficient to preserve the Hoeffding bound in a way that the output is similar to that of a conventional learner. The key question during the construction of the decision tree is the choice of attributes to be used for splits. Approximate ties are broken using a user-specified threshold of acceptable error-measure for the output. It can be shown that for any small value of δ, a particular choice of the split variable is the correct choice with probability at least

$1 - \delta$, if a sufficient number of stream records have been processed. This number has been shown in [13] to increase at a relatively modest rate of $\ln(1/\delta)$. This bound can then be extended to the entire decision tree, so as to quantify the probability that the same decision tree as a conventional learner is created. The VFDT method has also been extended to the case of evolving data streams. This framework is referred to as CVFDT [15], and it runs VFDT over fixed sliding windows in order to always have the most updated classifier. Jin and Agrawal [16] have extended the VFDT algorithm in order to process numerical attributes and reduce the sample size which is calculated using the Hoeffding bound. Since this approach reduces the sample size, it improves efficiency and space requirements for a given level of accuracy.

On Demand Classification. While most stream classification methods are focussed on a training stream, the on demand method is focussed on the case when both the training and the testing stream evolves over time. In the on demand classification method [14], we create class-specific micro-clusters from the underlying data. For an incoming record in the test stream, the class label of the closest micro-cluster is used in order to determine the class label of the test instance. In order to handle the problem of stream evolution, the micro-clusters from the specific time-horizon are used for the classification process. A key issue in this method is the choice of horizon which should be used in order to obtain the best classification accuracy. In order to determine the best horizon, a portion of the training stream is separated out and the accuracy is tested over this portion with different horizons. The optimal horizon is then used in order to classify the test instance.

Ensemble-based Classification. This technique [17] uses an ensemble of classification methods such as C4.5, RIPPER and naive Bayes in order to increase the accuracy of the predicted output. The broad idea is that a data stream may evolve over time, and a different classifier may work best for a given time period. Therefore, the use of an ensemble method provides robustness in the concept-drifting case.

3.3 Frequent Pattern Mining

The problem of frequent pattern mining was first introduced in [18], and was extensively analyzed for the conventional case of disk resident data sets. In the case of data streams, one may wish to find the frequent itemsets either over a sliding window or the entire data stream [19, 20]. In the case of data streams, the problem of frequent pattern mining can be studied under several models:

Entire Data Stream Model. In this model, the frequent patterns need to be mined over the entire data stream. Thus, the main difference from a conventional pattern mining algorithm is that the frequent patterns need to be mined in one pass over the entire data stream. Most frequent pattern mining algorithms require multiple passes in order to estimate the frequency of patterns of different sizes in the data. A natural method for frequent pattern counting is to use sketch-based algorithms in order to determine frequent patterns. Sketches are often used in order to determine

heavy-hitters in data streams, and therefore, an extension of the methodology to the problem of finding frequent patterns is natural. Along this line, Manku and Motwani [21] proposed the first one lass algorithm called *Lossy Counting*, in order to find all frequent itemsets over a data stream. The algorithm allows false positives, but not false negatives. Thus, for a given support level s, the algorithm is guaranteed not to contain all frequent itemsets whose support is greater than $s-\epsilon$. Another interesting approach in [22] determines all the frequent patterns whose support is greater than s with probability at least $1-\delta$, which the value of δ is as small as desired, as long as one is willing to add space and time complexity proportional to $\ln(1/\delta)$. Thus, this model does not allow false negatives, but may miss some of the frequent patterns. The main advantage of such a technique is that it is possible to provide a more concise set of frequent patterns at the expense of losing some of the patterns with some probability which is quite low for practical purposes.

Sliding Window Model. In many cases, the data stream may evolve over time, as a result of which it is desirable to determine all the frequent patterns over a particular sliding window. A method for determining the frequent patterns over a sliding window is discussed in [23]. The main assumption of this approach is that the number of frequent patterns are not very large, and therefore, it is possible to hold the transactions in each sliding window in main memory. The main focus of this approach is to determine closed frequent itemsets over the data stream. A new mining algorithm called MOMENT is proposed, and the main idea is based on the fact that the boundary between closed frequent itemsets and frequent itemsets moves very slowly. A closed enumeration tree is developed in order to keep track of the boundary between closed frequent itemsets and the rest of the itemsets. Another method which is able to mine frequent itemsets over arbitrary time granularities is referred to as FPSTREAM [24]. This method is essentially an adaptation of the FP-Tree method to data streams.

Damped Window Model. We note that pure sliding windows are not the only way by which the evolution of data streams can be taken into account during the mining process. A second way is to introduce a *decay factor* into the computation. Specifically, the weight of each transaction is multiplied by a factor of $f < 1$, when a new transaction arrives. The overall effect of such an approach is to create an exponential decay function on the arrivals in the data stream. Such a model is quite effective for evolving data stream, since recent transactions are counted more significantly during the mining process. An algorithm proposed in [25] maintains a lattice for recording the potentially frequent itemsets and their counts. While the counts of each lattice may change upon the arrival of each transaction, a key observation is that it is sufficient to update the counts in a lazy way. Specifically, the decay factor is applied only to those itemsets whose counts are affected by the current transaction. However, the decay factor will have to be applied in a modified way by taking into account the last time that the itemset was touched by an update. In other words, if t_c be the current transaction index, and the last time the count for the itemset was updated was at transaction index $t_s < t_c$, then we need to multiply the current counts of that itemset by $f^{t_s-t_c}$ before incrementing the count of this modified value. This approach works because the counts of each itemset reduce by the same decay factor

in each iteration, as long as a transaction count is not added to it. We note that such a lazy approach is also applicable to other mining problem, where statistics are represented as the sum of decaying values. For example, in [10], a similar lazy approach is used in order to maintain decay-based micro-cluster statistics for a high dimensional projected stream clustering algorithm.

3.4 Change Detection in Data Streams

As discussed earlier, the patterns in a data stream may evolve over time. In many cases, it is desirable to track and analyze the nature of these changes over time. In [26–28], a number of methods have been discussed for change detection of data streams. In addition, data stream evolution can also affect the behavior of the underlying data mining algorithms since the results can become stale over time. The broad algorithms for change diagnosis in data streams are as follows:
Velocity Density Estimation. In velocity density estimation [26], we compute the rate of change of data density of different points in the data stream over time. Depending upon the direction of density rate of change, one may identify regions of *dissolution*, *coagulation*, and *shift*. Spatial profiles can also be constructed in order to determine the directions of shift in the underlying data. In addition, it is possible to use the velocity density concept in order to identify those combinations of dimensions which have a high level of evolution. Another technique for change quantification is discussed in [27], which uses methods for probability difference quantification in order to identify the changes in the underlying data. In [28], a method is discussed in order to determine statistical changes in the underlying data. Clustering [7] can be used in order to determine significant evolution in the underlying data. In [7], micro-clustering is used in order to determine significant clusters which have evolved in the underlying data.

A separate line of work is the determination of significant changes in the results of data mining algorithms because of evolution. For example in [7], it has been shown how to determine significant evolving clusters in the underlying data. In [14], a similar technique has been used to keep a refreshed classification model in the presence of evolving data. In this respect, micro-clustering provides an effective technique, since it provides a way to store intermediate statistics of the underlying data in the form of clusters. In [14], a micro-cluster based nearest neighbor classifier is used in order to classify evolving data streams. The key idea is to construct class-specific micro-clusters over a variety of time horizons, and then utilize the time horizon with the greatest accuracy in order to perform the classification process. The issue of stream evolution has been extended to many other problems such as synopsis construction and reservoir sampling [29]. We will provide a detailed discussion of some of the methods such as reservoir sampling slightly later.

3.5 Synopsis Construction in Data Streams

The large volume of data streams poses unique space and time constraints on the computation process. Many query processing, database operations, and mining algorithms require efficient execution which can be difficult to achieve with a fast data stream. Furthermore, since it is impossible to fit the entire data stream within the available space, the space efficiency of the approach is a major concern. In many cases, it may be acceptable to generate *approximate solutions* for many problems by summarizing the data in a time and space-efficient way. In recent years, a number of *synopsis structures* have been developed, which can be used in conjunction with a variety of mining and query processing techniques [30]. Some key synopsis methods include those of sampling, wavelets, sketches and histograms. The key challenges which arise in the context of synopsis construction of data streams are as follows:

Broad Applicability. The synopsis structure is typically used as an intermediate representation, which is then leveraged for a variety of data mining and data management problems. Therefore, the synopsis structure should be constructed in such a way that it has applicability across a wide range of problems.

One-pass constraint. As in all data stream algorithms, the one-pass constraint is critical to synopsis construction algorithms. We would like to design all synopsis construction algorithms in one pass, and this is not the case for most traditional methods. In fact, even simply methods such as sampling need to be re-designed in order to handle the one-pass constraint.

Time and Space Efficiency. Since data streams have a very large volume, it is essential to create the synopsis in a time- and space-efficient way. In this sense, some of the probabilistic techniques such as sketches are extremely effective for counting-based applications, since they require constant-space for provable probabilistic accuracy. In other words, the time- and space-efficiency depends only upon the accuracy of the approach rather than the length of the data stream.

Data Stream Evolution. Since the stream evolves over time, a synopsis structure which is constructed from the overall behavior of the data stream is not quite as effective as one which uses recent history. Consequently, it is often more effective to create synopsis structures which either work with sliding windows, or use some decay-based approach in order to weight the data stream points.

One key characteristic of many of the above methods is that while they work effectively in the one-dimensional case, they often lose their effectiveness in the multi-dimensional case either because of data sparsity or because of inter-attribute correlations. Next, we will discuss the broad classes of techniques which are used for synopsis construction in data streams. Each of these techniques have their own advantages in different scenarios, and we will take care to provide an overview of the different array of methods which are used for synopsis construction in data streams. The broad techniques which are used for synopsis construction in data streams are as follows:

Reservoir Sampling. Sampling methods are widely used for traditional database applications, and are extremely popular because of their broad applicability across a

wide array of tasks in data streams. A further advantage of sampling methods is that unlike many other synopsis construction methods, they maintain their inter-attribute correlations across samples of the data. It is also often possible to use probabilistic inequalities in order to bound the effectiveness of a variety of applications with sampling methods.

However, a key problem in extending sampling methods to the data stream scenario, is that one does not know the total number of data points to be sampled in advance. Rather, one must maintain the sample in a dynamic way over the entire course of the computation. A method called reservoir sampling was first proposed in [31], which maintains such a sample dynamically. This technique was originally proposed in the context of one-pass access of data from magnetic-storage devices. However, the techniques also naturally extend to the data stream scenario.

Let us consider the case, where we wish to obtain an unbiased sample of size n from the data stream. In order to initialize the approach, we simply add the first n points from the stream to the reservoir. Subsequently, when the $(t+1)$th point is received, it is added to the reservoir with probability $n/(t+1)$. When the data point is added to the reservoir, it replaces a random point from the reservoir. It can be shown that this simple approach maintains the uniform sampling distribution from the data stream. We note that the uniform sampling approach may not be very effective in cases where the data stream evolves significantly. In such cases, one may either choose to generate the stream sample over a sliding window, or use a decay-based approach in order to bias the sample. An approach for sliding window computation over data streams is discussed in [32].

A second approach [29] uses biased decay functions in order to construct synopsis from data streams. It has been shown in [29] that the problem is extremely difficult for arbitrary decay functions. In such cases, there is no known solution to the problem. However, it is possible to design very simple algorithms for some important classes of decay functions. One of these classes of decay functions is the *exponential decay function*. The exponential decay function is extremely important because of its *memory less property*, which guarantees that the future treatment of a data point is independent of the past data points which have arrived. An interesting result is that by making simple implementation modifications to the algorithm of [31] in terms of modifying the probabilities of insertion and deletion, it is possible to construct a robust algorithm for this problem. It has been shown in [29] that the approach is quite effective in practice, especially when there is significant evolution of the underlying data stream.

While sampling has several advantages in terms of simplicity and preservation of multi-dimensional correlations, it loses its effectiveness in the presence of data sparsity. For example, a query which contains very few data points is unlikely to be accurate with the use of a sampling approach. However, this is a general problem with most techniques which are effective at counting frequent elements, but are not quite as effective at counting rare or distinct elements in the data stream.

Sketches. Sketches use some properties of random sampling in order to perform counting tasks in data streams. Sketches are most useful when the *domain size* of a data stream is very large. In such cases, the number of possible distinct elements

become very large, and it is no longer possible to track them in space-constrained scenarios. There are two broad classes of sketches: *projection based* and *hash based*. We will discuss each of them in turn.

Projection based sketches are constructed on the broad idea of random projection [33]. The most well known projection-based sketch is the AMS sketch [34, 35], which we will discuss below. It has been shown in [33], that by by randomly sampling subspaces from multi-dimensional data, it is possible to compute ϵ-accurate projections of the data with high probability. This broad idea can easily be extended to the massive domain case, by viewing each distinct item as a dimension, and the counts on these items as the corresponding values. The main problem is that the vector for performing the projection cannot be maintained explicitly since the length of such a vector would be of the same size as the number of distinct elements. In fact, since the sketch-based method is most relevant in the distinct element scenario, such an approach defeats the purpose of keeping a synopsis structure in the first place.

Let us assume that the random projection is performed using k sketch vectors, and r_i^j represents the jth vector for the ith item in the domain being tracked. In order to achieve the goal of efficient synopsis construction, we store the random vectors implicitly in the form of a seed, and this can be used to dynamically generate the vector. The main idea discussed in [36] is that it is possible to generate random vectors with a seed of size $O(\log(N))$, provided that one is willing to work with the restriction that $r_i^j \in \{-1, +1\}$ should be 4-wise independent. The sketch is computed by adding r_i^j to the jth component of the sketch for the ith item. In the event that the incoming item has frequency f, we add the value $f \cdot r_i^j$. Let us assume that there are a total of k sketch components which are denoted by $(s_1 \ldots s_k)$. Some key properties of the pseudo-random number generator approach and the sketch representation are as follows:

- A given component r_i^j can be generated in poly-logarithmic time from the seed. The time for generating the seed is poly-logarithmic in the domain size of the underlying data.
- A variety of approximate aggregate functions on the original data can be computed using the sketches.

Some example of functions which can be computed from the sketch components are as follows:

- *Dot Product of two streams*. If $(s_1 \ldots s_k)$ be the sketches from one stream, and $(t_1 \ldots t_k)$ be the sketches from the other stream, then $s_j cdott_j$ is a random variable whose expected value of the dot product.
- *Second Moment*. If $(s_1 \ldots s_k)$ be the sketch components for a data stream, it can be shown that the expected value of s_j^2 is the second moment. Furthermore, by using Chernoff bounds, it can be shown that by selecting the median of $O(\log(1/\delta)$ averages of $O(1/\epsilon^2)$ copies of $s_j]cdott_j$, it is possible to guarantee the accuracy of the approximation to within $1 \stackrel{+}{-} \epsilon$ with probability at least $1 - \delta$.

- *Frequent Items*. The frequency of the ith item in the data stream is computed by by multiplying the sketch component s_j by r_i^j. However, this estimation is accurate only for the case of frequent items, since the error is estimation is proportional to the overall frequency of the items in the data stream.

More details of computations which one can perform with the AMS sketch are discussed in [34, 35].

The second kind of sketch which is used for counting is the *count-min* sketch [37]. The count-min sketch is based upon the concept of hashing, and uses $k = \ln(1/\delta)$ pairwise-independent hash functions, which hash onto integers in the range $(0 \ldots e/\epsilon)$. For each incoming item, the k hash functions are applied and the frequency count is incremented by 1. In the event that the incoming item has frequency f, the corresponding frequency count is incremented by f. Note that by hashing an item into the k cells, we are ensuring that we maintain an overestimate on the corresponding frequency. It can be shown that the minimum of these cells provides the ϵ-accurate estimate to the frequency with probability at least $1 - \delta$. It has been shown in [37] that the method can also be naturally extended to other problems such as finding the dot product or the second-order moments. The count-min sketch is typically more effective for problems such as frequency-estimation of individual items than the projection-based AMS sketch. However, the AMS sketch is more effective for problems such as second-moment estimation.

Wavelet Decomposition. Another widely known synopsis representation in data stream computation is that of the wavelet representation. One of the most widely used representations is the *Haar Wavelet*. We will discuss this technique in detail in this section. This technique is particularly simple to implement, and is widely used in the literature for hierarchical decomposition and summarization. The basic idea in the wavelet technique is to create a decomposition of the data characteristics into a set of wavelet functions and basis functions. The property of the wavelet method is that the higher order coefficients of the decomposition illustrate the broad trends in the data, whereas the more localized trends are captured by the lower order coefficients.

We assume for ease in description that the length q of the series is a power of 2. This is without loss of generality, because it is always possible to decompose a series into segments, each of which has a length that is a power of two. The Haar Wavelet decomposition defines 2^{k-1} coefficients of order k. Each of these 2^{k-1} coefficients corresponds to a contiguous portion of the time series of length $q/2^{k-1}$. The ith of these 2^{k-1} coefficients corresponds to the segment in the series starting from position $(i-1) \cdot q/2^{k-1} + 1$ to position $i * q/2^{k-1}$. Let us denote this coefficient by ψ_k^i and the corresponding time series segment by S_k^i. At the same time, let us define the average value of the first half of the S_k^i by a_k^i and the second half by b_k^i. Then, the value of ψ_k^i is given by $(a_k^i - b_k^i)/2$. More formally, if Φ_k^i denote the average value of the S_k^i, then the value of ψ_k^i can be defined recursively as follows:

$$\psi_k^i = (\Phi_{k+1}^{2 \cdot i-1} - \Phi_{k+1}^{2 \cdot i})/2 \tag{1}$$

The set of Haar coefficients is defined by the Ψ_k^i coefficients of order 1 to $\log_2(q)$. In addition, the global average Φ_1^1 is required for the purpose of perfect reconstruction. We note that the coefficients of different order provide an understanding of the major trends in the data at a particular level of granularity. For example, the coefficient ψ_k^i is half the quantity by which the first half of the segment S_k^i is larger than the second half of the same segment. Since larger values of k correspond to geometrically reducing segment sizes, one can obtain an understanding of the basic trends at different levels of granularity. We note that this definition of the Haar wavelet makes it very easy to compute by a sequence of averaging and differencing operations. In Table 1, we have illustrated how the wavelet coefficients are computed for the case of the sequence $(8, 6, 2, 3, 4, 6, 6, 5)$. This decomposition is illustrated in graphical form in Fig. 1. We also note that each value can be represented as a sum of $\log_2(8) = 3$ linear decomposition components. In general, the entire decomposition

Table 1 An example of wavelet coefficient computation

Granularity (Order k)	Averages Φ values	DWT Coefficients ψ values
$k = 4$	(8, 6, 2, 3, 4, 6, 6, 5)	–
$k = 3$	(7, 2.5, 5, 5.5)	(1, −0.5,−1, 0.5)
$k = 2$	(4.75, 5.25)	(2.25, −0.25)
$k = 1$	(5)	(-0.25)

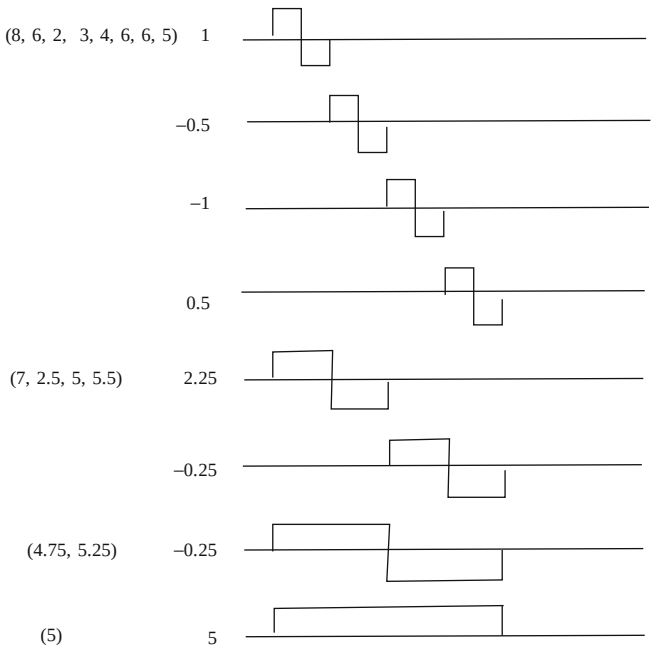


Fig. 1 Illustration of the wavelet decomposition

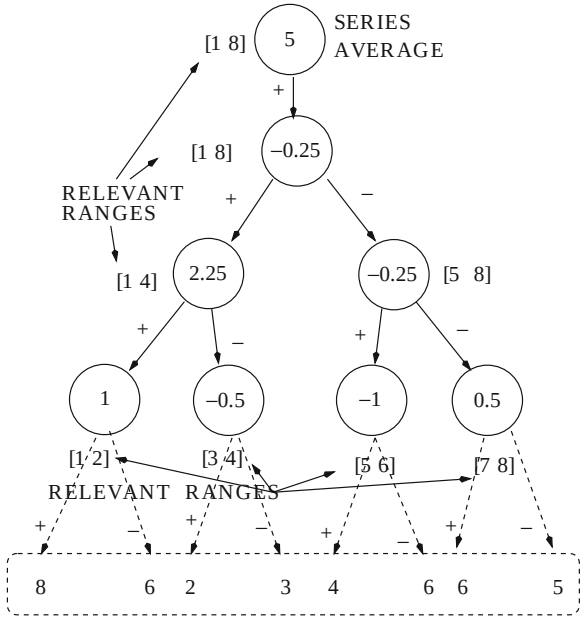


Fig. 2 The error tree from the wavelet decomposition

may be represented as a tree of depth 3, which represents the hierarchical decomposition of the entire series. This is also referred to as the *error tree*. In Fig. 2, we have illustrated the error tree for the wavelet decomposition illustrated in Table 1. The nodes in the tree contain the values of the wavelet coefficients, except for a special *super-root* node which contains the series average. This super-root node is not necessary if we are only considering the relative values in the series, or the series values have been normalized so that the average is already zero. We further note that the number of wavelet coefficients in this series is 8, which is also the length of the original series. The original series has been replicated just below the error-tree in Fig. 2, and it can be reconstructed by adding or subtracting the values in the nodes along the path leading to that value. We note that each coefficient in a node should be added, if we use the left branch below it to reach to the series values. Otherwise, it should be subtracted. This natural decomposition means that an entire contiguous range along the series can be reconstructed by using only the portion of the error-tree which is relevant to it. Furthermore, we only need to retain those coefficients whose values are significantly large, and therefore affect the values of the underlying series. In general, we would like to minimize the reconstruction error by retaining only a fixed number of coefficients, as defined by the space constraints. While wavelet decomposition is easy to perform for multi-dimensional data sets, it is much more challenging for the case of data streams. This is because data streams impose a one-pass constraint on the wavelet construction process. A variety of one-pass algorithms for wavelet construction are discussed in [30].

Histograms. The technique of histogram construction is closely related to that of wavelets. In histograms the data is binned into a number of intervals along an attribute. For any given query, the counts from the bins can be utilized for query resolution. A simple representation of the histogram method would simply partition the data into equi-depth or equi-width intervals. The main inaccuracy with the use of histograms is that the distribution of data points within a bucket is not retained, and is therefore assumed to be uniform. This causes inaccuracy because of extrapolation at the query boundaries. A natural choice is to use an equal number of counts in each bucket. This minimizes the error variation across different buckets. However, in the case of data streams, the boundaries to be used for equi-depth histogram construction are not known a priori. We further note that the design of equi-depth buckets is exactly the problem of quantile estimation, since the equi-depth partitions define the quantiles in the data. Another choice of histogram construction is that of minimizing the variance of frequency variances of different values in the bucket. This ensures that the uniform distribution assumption is approximately held, when extrapolating the frequencies of the buckets at the two ends of a query. Such histograms are referred to as V-optimal histograms. Algorithms for V-optimal histogram construction are proposed in [38, 39]. A more detailed discussion of several algorithms for histogram construction may be found in [3].

3.6 Dimensionality Reduction and Forecasting in Data Streams

Because of the inherent temporal nature of data streams, the problems of dimensionality reduction and forecasting and particularly important. When there are a large number of simultaneous data stream, we can use the correlations between different data streams in order to make effective predictions [40, 41] on the future behavior of the data stream. In particular, the well known MUSCLES method [41] is useful in applying regression analysis to data streams. The regression analysis is helpful in predicting the future behavior of the data stream. A related technique is the SPIRIT algorithm, which explores the relationship between dimensionality reduction and forecasting in data streams. The primary idea is that a compact number of hidden variables can be used to comprehensively describe the data stream. This compact representation can also be used for effective forecasting of the data streams. A discussion of different dimensionality reduction and forecasting methods (including SPIRIT) is provided in [3].

3.7 Distributed Mining of Data Streams

In many instances, streams are generated at multiple distributed computing nodes. An example of such a case would be sensor networks in which the streams are generated at different sensor nodes. Analyzing and monitoring data in such environments

requires data mining technology that requires optimization of a variety of criteria such as communication costs across different nodes, as well as computational, memory or storage requirements at each node. There are several management and mining challenges in such cases. When the streams are collected with the use of sensors, one must take into account the limited storage, computational power, and battery life of sensor nodes. Furthermore, since the network may contain a very large number of sensor nodes, the effective aggregation of the streams becomes a considerable challenge. Furthermore, distributed streams also pose several challenges to mining problems, since one must integrate the results of the mining algorithms across different nodes. A detailed discussion of several distributed mining algorithms are provided in [3].

4 Scientific Applications of Data Streams

Data streams have numerous applications in a variety of scientific scenarios. In this section, we will discuss different applications of data streams and how they tie in to the techniques discussed earlier.

4.1 Network Monitoring

Many large telecommunication companies have massive streams of data containing information about phone calls between different nodes. In many cases, it is desirable to analyze the underlying data in order to determine the broad patterns in the data. This can be extremely difficult especially if the number of source-destination combinations are very large. For example, if the company has over 10^6 possible nodes for both the source and the destination, the number of possible combinations is 10^{12}. Maintaining explicit information about such a large number of pairs is practically infeasible both from a space-and computational point of view.

Many natural solutions have been devised for this problem which rely on the use of a sketch-based approach in order to compress and summarize the underlying data. Sketches are extremely efficient because they use an additive approach in order to summarize the underlying data stream. Sketches can be used in order to determine important patterns such frequent call patterns, moments or even joins across multiple data sources.

4.2 Intrusion Detection

In many network applications, the intrusions appear as sudden bursts of patterns in even greater streams of attacks over the world-wide web. This makes the problem

extremely difficult, because one cannot scan the data twice, and we are looking for patterns which are embedded in a much greater volume of data.

Stream clustering turns out to be quite useful for such problems, since we can isolate small clusters in the data from much larger volumes of data. The formation of new clusters often signifies an anomalous event which needs to be investigated. If desired, the problem can be combined with supervised mining of the underlying data. This can be done by creating supervised clusters in which each cluster may belong only to a specific-class. When known intrusions are received in the stream, they can be used in order to create class-specific clusters. These class-specific clusters can be used to determine the nature of new clusters which arise from unknown intrusion behavior.

4.3 Sensor Network Analysis

Sensors have played an increasingly important role in recent years in collecting a variety of scientific data from the environment. The challenges in processing sensor-data are as follows:

- In many cases, sensor data may be uncertain in nature. The challenge is to clean the data in online fashion and then apply various application-specific algorithms to the problem. An example for the case of clustering uncertain data streams is discussed in [11].
- The number of streams which are processed together are typically very large. This is because of the large number of sensors at which the data may be collected. This leads to challenges in effective storage and processing of such data.
- Often the data from different sensors may only be available in aggregated form in order to save on storage space. This leads to challenges in extraction of the underlying information.

Synopsis construction techniques are a natural approach for sensor problems because of the nature of the underlying aggregation. Sketch techniques can be used in order to compress the underlying data and use it for a variety of data mining purposes.

4.4 Cosmological Applications

In recent years, cosmological applications have created large volumes of data. The installation of large space stations, space telescopes, and observatories result in large streams of data on different stars and clusters of galaxies. This data can be used in order to mine useful information about the behavior of different cosmological objects. Similarly, rovers and sensors on a planet or asteroid may send large amounts of image, video or audio data. In many cases, it may not be possible to manually

monitor such data continuously. In such cases, it may be desirable to use stream mining techniques in order to detect the important underlying properties.

The amount of data received in a single day in such applications can often exceed several tera-bytes. These data sources are especially challenging since the underlying applications may be spatial in nature. In such cases, an attempt to compress the data using standard synopsis techniques may lose the structure of the underlying data. Furthermore, the data may often contain imprecision in measurements. Such imprecisions may result in the need for techniques which leverage the uncertainty information in the data in order to improve the accuracy of the underlying results.

4.5 Mobile Applications

Recently new technologies have emerged which can use the information gleaned from on-board sensors in a vehicle in order to monitor the diagnostic health of the vehicle as well as driver characterization. Two such applications are the VEDAS system [42], and the OnStar system designed by General Motors. Such systems require quick analysis of the underlying data in order to make diagnostic characterizations in real time. Effective event-detection algorithms are required in order to perform this task effectively.

The stock market often creates large volumes of data streams which need to be analyzed in real time in order to make quick decisions about actions to be taken. An example of such an approach is the MobiMine approach [43] which monitors the stock market with the use of a PDA.

4.6 Environmental and Weather Data

Many satellites and other scientific instruments collect environmental data such as cloud cover, wind speeds, humidity data and ocean currents. Such data can be used to make predictions about long- and short-term weather and climate changes. Such data can be especially massive if the number of parameters measured are very large. The challenge is to be able to combine these parameters in order to make timely and accurate predictions about weather driven events. This is another application of event detection techniques from massive streams of sensor data.

5 Conclusions and Research Directions

Data streams are a computational challenge to data mining problems because of the additional algorithmic constraints created by the large volume of data. In addition, the problem of temporal locality leads to a number of unique mining challenges

in the data stream case. This chapter provides an overview to the generic issues in processing data streams, and the specific issues which arise with different mining algorithms.

While considerable research has already been performed in the data stream area, there are numerous research directions which remain to be explored. Most research in the stream area is focussed on the one pass constraint, and generally does not deal effectively with the issue of temporal locality. In the stream case, temporal locality remains an extremely important issue since the data may evolve considerably over time. Other important topics which need to be explored are as follows:

- Streams are often collected by devices such as sensors in which the data is often noisy and error-driven. Therefore, a key challenge is to effectively clean the data. This may involve either imputing or modeling the underlying uncertain data. This can be challenge, since any modeling needs to be done in real time, as large volumes of the data stream arrive.
- A related area of research is in using the modeled data for data mining tasks. Since the underlying data is uncertain, the uncertainty should be used in order to improve the quality of the underlying results. Some recent research addresses the issue of clustering uncertain data streams [11].
- Many recent applications such as privacy-preserving data mining have not been studied effectively in the context of data streams. It is often a challenge to perform privacy-transformations of continuously arriving data, since newly arriving data may compromise the integrity of earlier data. The data stream domain provides a number of unique challenges in the context of the privacy problem.

Acknowledgments Research of the first author was sponsored in part by the US Army Research laboratory and the UK ministry of Defense under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies of the US Government, the US Army Research Laboratory, the UK Ministry of Defense, or the UK Government. The US and UK governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice hereon.

References

1. G. Cormode, M. Garofalakis, Sketching Streams Through the Net: Distributed Approximate Query Tracking, in *VLDB Conference*, 2005
2. G. Kollios, J. Byers, J. Considine, M. Hadjieleftheriou, F. Li, (2005) Robust Aggregation in Sensor Networks. *IEEE Data Engineering Bulletin*
3. C. Aggarwal, (2007) *Data Streams: Models and Algorithms* (Springer, Berlin, 2007)
4. Y. Chen, G. Dong, J. Han, B.W. Wah, J. Wang, Multi-dimensional regression analysis of time-series data streams, in *VLDB Conference*, 2002
5. G. Dong, J. Han, J. Lam, J. Pei, K. Wang, Mining multi-dimensional constrained gradients in data cubes, in *VLDB Conference*, 2001
6. S. Guha, N. Mishra, R. Motwani, L. O'Callaghan, Clustering Data Streams, in *IEEE FOCS Conference*, 2000

7. C. Aggarwal, J. Han, J. Wang, P. Yu, A Framework for Clustering Evolving Data Streams, *VLDB Conference*, 2003
8. T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: An Efficient Data Clustering Method for Very Large Databases, in *ACM SIGMOD Conference*, 1996
9. C. Aggarwal, C. Procopiuc, J. Wolf, P. Yu, J.-S. Park, Fast Algorithms for Projected Clustering, in *ACM SIGMOD Conference*, 1999
10. C. Aggarwal, J. Han, J. Wang, P. Yu, A Framework for High Dimensional Projected Clustering of Data Streams, in *VLDB Conference*, 2004
11. C. Aggarwal, P. Yu, A Framework for Clustering Uncertain Data Streams, in *ICDE Conference*, 2008
12. C. Aggarwal, P. Yu, A Framework for Clustering Massive Text and Categorical Data Streams, in *SIAM Data Mining Conference*, 2006
13. P. Domingos, G. Hulten, Mining High-Speed Data Streams, in *Proceedings of the ACM KDD Conference*, 2000
14. C. Aggarwal, J. Han, J. Wang, P. Yu, On-Demand Classification of Data Streams, in *ACM KDD Conference*, 2004
15. G. Hulten, L. Spencer, P. Domingos, Mining Time Changing Data Streams, in *ACM KDD Conference*, 2001
16. R. Jin, G. Agrawal, Efficient Decision Tree Construction on Streaming Data, in *ACM KDD Conference*, 2003
17. H. Wang, W. Fan, P. Yu, J. Han, Mining Concept-Drifting Data Streams using Ensemble Classifiers, in *ACM KDD Conference*, 2003
18. R. Agrawal, T. Imielinski, A. Swami, Mining Association Rules between Sets of items in Large Databases, in *ACM SIGMOD Conference*, 1993
19. C. Giannella, J. Han, J. Pei, X. Yan, P. Yu, (2002) Mining Frequent Patterns in Data Streams at Multiple Time Granularities. *Proceedings of the NSF Workshop on Next Generation Data Mining*
20. R. Jin, G. Agrawal, An algorithm for in-core frequent itemset mining on streaming data, in *ICDM Conference*, 2005
21. G. Manku, R. Motwani, Approximate Frequency Counts over Data Streams, *VLDB Conference*, 2002
22. J. Xu Yu, Z. Chong, H. Lu, A. Zhou, False positive or false negative: Mining frequent itemsets from high speed transaction data streams, in *VLDB Conference*, 2004
23. Y. Chi, H. Wang, P. Yu, R. Muntz Moment: Maintaining closed frequent itemsets over a stream sliding window, *ICDM Conference*, 2004
24. C. Gianella, J. Han, J. Pei, X. Yan, P. Yu, (2002) Mining Frequent Patterns in Data Streams at Multiple Time Granularities. *NSF Workshop on Next Generation data Mining*
25. J.H. Chang, W.S. Lee, Finding recent frequent itemsets adaptively over online data streams, in *ACM KDD Conference*, 2003
26. C. Aggarwal, A Framework for Diagnosing Changes in Evolving Data Streams, in *ACM SIGMOD Conference*, 2003
27. T. Dasu, S. Krishnan, S. Venkatasubramaniam, K. Yi, (2005). An Information-Theoretic Approach to Detecting Changes in Multi-dimensional data Streams. *Duke University Technical Report CS-2005-06*
28. D. Kifer, S.-B. David, J. Gehrke, Detecting Change in Data Streams, in *VLDB Conference*, 2004
29. C. Aggarwal, On Biased Reservoir Sampling in the presence of Stream Evolution, in *VLDB Conference*, 2006
30. M. Garofalakis, J. Gehrke, R. Rastogi, Querying and mining data streams: you only get one look (a tutorial), in *SIGMOD Conference*, 2002
31. J.S. Vitter, ACM Transactions on Mathematical Software **11**(1), 37 1985
32. B. Babcock, M. Datar, R. Motwani, (2002) Sampling from a Moving Window over Streaming Data. *SIAM Symposium on Discrete Algorithms (SODA).*
33. W. Johnson, J. Lindenstrauss, Contemporary Mathematics **26**, 189 1984

34. N. Alon, P. Gibbons, Y. Matias, M. Szegedy, Tracking Joins and Self-Joins in Limited Storage, in *ACM PODS Conference*, 1999
35. N. Alon, Y. Matias, M. Szegedy, in *The Space Complexity of Approximating Frequency Moments*,(ACM, New York 1996), pp. 20–29.
36. P. Indyk, (2000) Stable Distributions, Pseudorandom Generators, Embeddings and Data Stream Computation. *IEEE FOCS*
37. G. Cormode, S. Muthukrishnan, (2004) An Improved Data Stream Summary: The Count-Min Sketch and its Applications. *LATIN*, pp. 29–38
38. Y. Ioannidis, V. Poosala, Balancing Histogram Optimality and Practicality for Query Set Size Estimation, in *ACM SIGMOD Conference*, 1995
39. H. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. Sevcik, T. Suel, (1998) Optimal Histograms with Quality Guarantees, in *VLDB Conference*, 1998
40. Y. Sakurai, S. Papadimitriou, C. Faloutsos, BRAID: Stream mining through group lag correlations, *ACM SIGMOD Conference*, 2005
41. B.-K. Yi, N.D. Sidiropoulos, T. Johnson, H.V. Jagadish, C. Faloutsos, A. Biliris, Online data mining for co-evolving time sequences, in *ICDE Conference*, 2000
42. H. Kargupta et al., VEDAS: A Mobile and Distributed Data Stream Mining System for Vehicle Monitoring, in *SDM Conference*, 2004
43. H. Kargupta et al., MobiMine: Monitoring the stock market using a PDA, *ACM SIGKDD Explorations*, January 2002

“This page left intentionally blank.”

Index