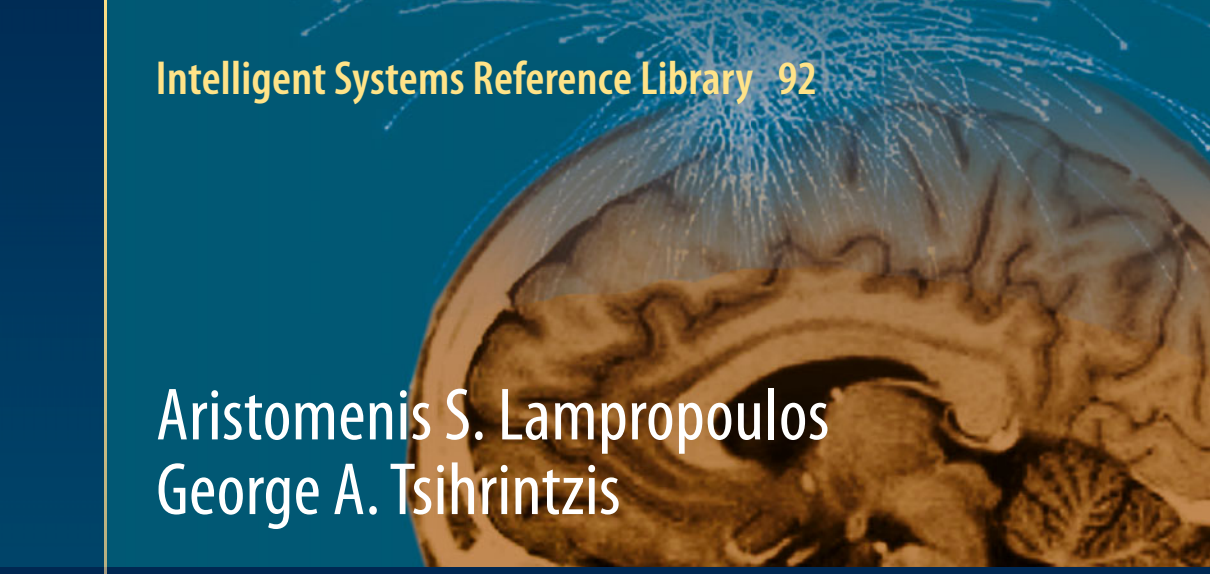
Intelligent Systems Reference Library 92



Aristomenis S. Lampropoulos
George A. Tsihrintzis

Machine Learning Paradigms

Applications in Recommender Systems

Springer

Intelligent Systems Reference Library

Volume 92

About this Series

The aim of this series is to publish a Reference Library, including novel advances and developments in all aspects of Intelligent Systems in an easily accessible and well structured form. The series includes reference works, handbooks, compendia, textbooks, well-structured monographs, dictionaries, and encyclopedias. It contains well integrated knowledge and current information in the field of Intelligent Systems. The series covers the theory, applications, and design methods of Intelligent Systems. Virtually all disciplines such as engineering, computer science, avionics, business, e-commerce, environment, healthcare, physics and life science are included.

More information about this series at http://www.springer.com/series/8578

Aristomenis S. Lampropoulos
George A. Tsihrintzis

Machine Learning Paradigms

Applications in Recommender Systems

Springer

Aristomenis S. Lampropoulos
Department of Informatics
University of Piraeus
Piraeus
Greece

George A. Tsihrintzis
Department of Informatics
University of Piraeus
Piraeus
Greece

ISSN 1868-4394 ISSN 1868-4408 (electronic)
Intelligent Systems Reference Library
ISBN 978-3-319-19134-8 ISBN 978-3-319-19135-5 (eBook)
DOI 10.1007/978-3-319-19135-5

Library of Congress Control Number: 2015940994

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

To my beloved family and friends

Aristomenis S. Lampropoulos

To my wife and colleague, Prof.-Dr. Maria Virvou, and our daughters, Evina, Konstantina and Andreani

George A. Tsihrintzis

Foreword

Recent advances in Information and Communication Technologies (ICT) have increased the computational power of computers, while at the same time, various mobile devices are embedded in them. The combination of the two leads to an enormous increase in the extent and complexity of data generation, storage, and sharing. “Big data” is the term commonly used to describe data so extensive and complex that they may overwhelm their user, overload him/her with information, and eventually, frustrate him/her. YouTube for example, has more than 1 billion unique visitors each month, uploading 72 hours of video every minute! It would be extremely difficult for a user of YouTube to retrieve the content he/she is really interested in unless some help is provided.

Similar difficulties arise with all types of multimedia data, such as audio, image, video, animation, graphics, and text. Thus, innovative methods to address the problem of extensive and complex data are expected to prove useful in many and diverse data management applications.

In order to reduce the risk of information overload of users, recommender system research and development aims at providing ways of individualizing the content returned to a user via attempts to understand the user’s needs and interests. Specific recommender systems have proven useful in assisting users in selecting books, music, movies, clothes, and content of various other forms.

At the core of recommender systems lie machine learning algorithms, which monitor the actions of a recommender system user and learn about his/her needs and interests. The fundamental idea is that a user provides directly or indirectly examples of content he/she likes (“positive examples”) and examples of content he/she dislikes (“negative examples”) and the machine learning module seeks and recommends content “similar” to what the user likes and avoids recommending content “similar” to what the user dislikes. This idea sounds intuitively correct and has, indeed, led to useful recommender systems. Unfortunately, users may be willing to provide examples of content they like, but are very hesitant when asked to provide examples of content they dislike. Recommender systems built on the assumption of availability of both positive and negative examples do not perform well when negative examples are rare.

It is exactly this problem that the authors have tackled in their book. They collect results from their own recently-published research and propose an innovative approach to designing recommender systems in which only positive examples are made available by the user. Their approach is based on one-class classification methodologies in recent machine learning research.

The blending of recommender systems and one-class classification seems to be providing a new very fertile field for research, innovation, and development. I believe the authors have done a good job addressing the book topic. I consider the book at hand particularly timely and expect that it will prove very useful to researchers, practitioners, and graduate students dealing with problems of extensive and complex data.

March 2015

Dumitru Dan Burdescu
Professor, Eng., Math., Ph.D.
Head of Software Engineering Department, Director of
"Multimedia Application Development" Research Centre
Faculty of Automation, Computers and Electronics
University of Craiova, Craiova, Romania

Preface

Recent advances in electronic media and computer networks have allowed the creation of large and distributed repositories of information. However, the immediate availability of extensive resources for use by broad classes of computer users gives rise to new challenges in everyday life. These challenges arise from the fact that users cannot exploit available resources effectively when the amount of information requires prohibitively long user time spent on acquaintance with and comprehension of the information content. Thus, the risk of information overload of users imposes new requirements on the software systems that handle the information. Such systems are called **Recommender Systems** (RS) and attempt to provide information in a way that will be most appropriate and valuable to its users and prevent them from being overwhelmed by huge amounts of information that, in the absence of RS, they should browse or examine.

In this monograph, first, we explore the use of objective content-based features to model the individualized (subjective) perception of similarity between multimedia data. We present a content-based RS which constructs music similarity perception models of its users by associating different similarity measures to different users. The results of the evaluation of the system verify the relation between subsets of objective features and individualized (music) similarity perception and exhibit significant improvement in individualized perceived similarity in subsequent recommended items. The investigation of these relations between objective feature subsets and user perception offer an indirect explanation and justification for the items one selects. The users are clustered according to specific subsets of features that reflect different aspects of the music signal. This assignment of a user to a specific subset of features allows us to formulate indirect relations between his/her perception and corresponding item similarity (e.g., music similarity) that involve his/her preferences. Consequently, the selection of a specific feature subset can provide a justification/reasoning of the various factors that influence the user's perception of similarity to his/her preferences.

Secondly, we address the recommendation process as a hybrid combination of one-class classification with collaborative filtering. Specifically, we follow a cascade scheme in which the recommendation process is decomposed into two levels.

In the first level, our approach attempts to identify for each user only the desirable items from the large amount of all possible items, taking into account only a small portion of his/her available preferences. Toward this goal, we apply a one-class classification scheme, in the training stage of which only positive examples (desirable items for which users have expressed an opinion-rating value) are required. This is very important, as it is sensibly hard in terms of time and effort for users to explicitly express what they consider as non-desirable to them. In the second level, either a content-based or a collaborative filtering approach is applied to assign a corresponding rating degree to these items. Our cascade scheme first builds a user profile by taking into consideration a small amount of his/her preferences and then selects possible desirable items according to these preferences which are refined and into a rating scale in the second level. In this way, the cascade hybrid RS avoids known problems of content-based or collaborative filtering RS.

The fundamental idea behind our cascade hybrid recommendation approach is to mimic the social recommendation process in which someone has already identified some items according to his/her preferences and seeks the opinions of others about these items, so as to make the best selection of items that fall within his/her individual preferences. Experimental results reveal that our hybrid recommendation approach outperforms both a pure content-based approach or a pure collaborative filtering technique. Experimental results from the comparison between the pure collaborative and the cascade content-based approaches demonstrate the efficiency of the first level. On the other hand, the comparison between the cascade content-based and the cascade hybrid approaches demonstrates the efficiency of the second level and justifies the use of the collaborative filtering method in the second level.

Piraeus, Greece
March 2015

Aristomenis S. Lampropoulos
George A. Tsihrintzis

Acknowledgments

We would like to thank Prof. Dr. Lakhmi C. Jain for agreeing to include this monograph in the Intelligent Systems Reference Library (ISRL) book series of Springer that he edits. We would also like to thank Prof. Dumitru Dan Burdescu of the University of Craiova, Romania, for writing a foreword to the monograph. Finally, we would like to thank the Springer staff for their excellent work in typesetting and publishing this monograph.

Contents

Chapter 1
Introduction

Abstract Recent advances in electronic media and computer networks have allowed the creation of large and distributed repositories of information. However, the immediate availability of extensive resources for use by broad classes of computer users gives rise to new challenges in everyday life. These challenges arise from the fact that users cannot exploit available resources effectively when the amount of information requires prohibitively long user time spent on acquaintance with and comprehension of the information content. Thus, the risk of information overload of users imposes new requirements on the software systems that handle the information. One of these requirements is the incorporation into the software systems of mechanisms that help their users when they face difficulties during human-computer interaction sessions or lack the knowledge to make decisions by themselves. Such mechanisms attempt to identify user information needs and to personalize human-computer interactions. (Personalized) Recommender Systems (RS) provide an example of software systems that attempt to address some of the problems caused by information overload. This chapter provides an introduction to Recommender Systems.

1.1 Introduction to Recommender Systems

RS are defined in [16] as software systems in which "people provide recommendations as inputs, which the system then aggregates and directs to appropriate recipients." Today, the term includes a wider spectrum of systems describing any system that provides individualization of the recommendation results and leads to a procedure that helps users in a personalized way to interesting or useful objects in a large space of possible options. RS form an important research area because of the abundance of their potential practical applications.

Clearly, the functionality of RS is similar to the social process of recommendation and reduction of information that is useless or uninteresting to the user. Thus, one might consider RS as similar to search engines or information retrieval systems. However, RS are to be differentiated from search engines or information retrieval systems as a RS not only finds results, but additionally uses its embedded individualization and personalization mechanisms to select objects (items) that satisfy the

A.S. Lampropoulos and G.A. Tsihrintzis, *Machine Learning Paradigms*,
Intelligent Systems Reference Library 92, DOI 10.1007/978-3-319-19135-5_1

specific querying user needs. Thus, unlike search engines or information retrieval systems, a RS provides information in a way that will be most appropriate and valuable to its users and prevents them from being overwhelmed by huge amounts of information that, in the absence of RS, they should browse or examine. This is to be contrasted with the target of a search engine or an information retrieval system which is to "match" items to the user query. This means that a search engine or an information retrieval system tries to form and return *a ranked list* of all those items that match the query. Techniques of active learning such as *relevance-feedback* may give these systems the ability to refine their results according to the user preferences and, thus, provide a simple form of recommendation. More complex search engines such as *GOOGLE* utilize other kinds of criteria such as "*authoritativeness*", which aim at returning as many useful results as possible, but *not* in an individualized way.

A learning-based RS typically works as follows: (1) the recommender system collects all given recommendations at one place and (2) applies a learning algorithm, thereafter. Predictions are then made either with a model learnt from the dataset (model-based predictions) using, for example, a clustering algorithm [3, 18] or on the fly (memory-based predictions) using, for example, a nearest neighbor algorithm [3, 15]. A typical prediction can be a list of the top-N recommendations or a requested prediction for a single item [7].

Memory-based methods store training instances during training which are can be retrieved when making predictions. In contrast, model-based methods generalize into a model from the training instances during training and the model needs to be updated regularly. Then, the model is used to make predictions. Memory-based methods learn fast but make slow predictions, while model-based methods make fast predictions but learn slowly.

The roots of RS can be traced back to Malone et al. [11], who proposed three forms of filtering: cognitive filtering (now called content-based filtering), social filtering (now called collaborative filtering (CF)) and economic filtering. They also suggested that the best approach was probably to combine these approaches into the category of, so-called, *hybrid* RS.

1.2 Formulation of the Recommendation Problem

In general, the recommendation problem is defined as the problem of estimating ratings for the items that have not been seen by a user. This estimation is based on:

- ratings given by the user to other items,
- ratings given to an item by other users,
- and other user and item information (e.g. item characteristics, user demographics).

The recommendation problem can be formulated [1] as follows:

Let U be the *set of all users* $U = \{u_1, u_2, ..., u_m\}$ and let I be the *set of all possible items* $I = \{i_1, i_2, ..., i_n\}$ that can be recommended, such as music files, images, movies, etc. The space I of possible items can be very large.

Let f be a utility function that measures the usefulness of item i to user u,

$$f : U \times I \rightarrow R, \tag{1.1}$$

where R is a totally ordered set (e.g. the set of nonnegative integers or real numbers within a certain range). Then, for each user $u \in U$, we want to choose an item $i' \in I$ that maximizes the user utility function, i.e.

$$\forall u \in U, i'_u = \arg\max_{i \in I} f(u, i). \tag{1.2}$$

In RS, the utility of an item is usually represented by a rating, which indicates how a particular user liked a particular item, e.g., user u_1 gave the object i_1 the rating of $R(1, 1) = 3$, where $R(u, i) \in \{1, 2, 3, 4, 5\}$.

Each user u_k, where $k = 1, 2, ..., m$, has a list of items I_{u_k} about which the user has expressed his/her preferences. It is important to note that $I_{u_k} \subseteq I$, while it is also possible for I_{u_k} to be the null set. This latter means that users are not required to express their preferences for all existing items.

Each element of the user space U can be defined with a profile that includes various user characteristics, such as age, gender, income, marital status, etc. In the simplest case, the profile can contain only a single (unique) element, such as User ID.

Recommendation algorithms enhance various techniques by operating

- either on *rows* of the matrix R, which correspond to ratings of a single user about different items,
- or on *columns* of the matrix R, which correspond to different users' ratings for a single item.

However, in general, the utility function can be an arbitrary function, including a profit function. Depending on the application, a utility f can either be specified by the user, as is often done for the user-defined ratings, or computed by the application, as can be the case for a profit-based utility function. Each element of the user space U can be defined with a profile that includes various user characteristics, such as age, gender, income, marital status, etc. In the simplest case, the profile can contain only a single (unique) element, such as User ID.

Similarly, each element of the item space I is defined via a set of characteristics. The central problem of RS lies in that a utility function f is usually not defined on the entire $U \times I$ space, but only on some subset of it. This means that f needs to be ***generalized*** to the entire space $U \times I$. In RS, a utility is typically represented by ratings and is initially defined only on the items previously rated by the users.

Generalizations from known to unknown ratings are usually done by:

- specifying heuristics that define the utility function and empirically validating its performance, or
- estimating the utility function that optimizes a certain performance criterion, such as Mean Absolute Error (MAE).

Once the unknown ratings are estimated, actual recommendations of an item to a user are made by selecting the highest rating among all the estimated ratings for that user, according to Eq. 1.2. Alternatively, we can recommend the N best items to a user. Additionally, we can recommend a set of users to an item.

1.2.1 The Input to a Recommender System

The input to a RS depends on the type of the filtering algorithm employed. The input belongs to one of the following categories:

1. Ratings (also called votes), which express the opinion of users on items. Ratings are normally provided by the user and follow a specified numerical scale (example: 1-bad to 5-excellent). A common rating scheme is the binary rating scheme, which allows only ratings of either 0 or 1. Ratings can also be gathered implicitly from the users purchase history, web logs, hyper-link visits, browsing habits or other types of information access patterns.
2. Demographic data, which refer to information such as the age, the gender and the education of the users. This kind of data is usually difficult to obtain. It is normally collected explicitly from the user.
3. Content data, which are based on content analysis of items rated by the user. The features extracted via this analysis are used as input to the filtering algorithm in order to infer a user profile.

1.2.2 The Output of a Recommender System

The output of a RS can be either a *prediction* or a *recommendation*.

- A *prediction* is expressed as a numerical value, $R_{a,j} = R(u_a, i_j)$, which represents the anticipated opinion of active user u_a for item i_j. This predicted value should necessarily be within the same numerical scale (example: 1-bad to 5-excellent) as the input referring to the opinions provided initially by active user u_a. This form of RS output is also known as *Individual Scoring*.
- A *recommendation* is expressed as a list of N items, where $N \leq n$, which the active user is expected to like the most. The usual approach in that case requires this list to include only items that the active user has not already purchased, viewed or rated. This form of RS output is also known as *Top-N Recommendation* or *Ranked Scoring*.

1.3 Methods of Collecting Knowledge About User Preferences

To generate personalized recommendations that are tailored to the specific needs of the active user, RS collect ratings of items by users and build user-profiles in ways that depend on the methods that the RS utilize to collect personal information about user preferences. In general, these methods are categorized into three approaches:

- an *Implicit approach*, which is based on recording user behavior,
- an *Explicit approach*, which is based on user interrogation,
- a *Mixing approach*, which is a combination of the previous two.

1.3.1 The Implicit Approach

This approach does not require active user involvement in the knowledge acquisition task, but, instead, the user behavior is recorded and, specifically, the way that he/she reacts to each incoming piece of data. The goal is to learn from the user reaction about the relevance of the data item to the user. Typical examples for implicit ratings are purchase data or reading time of Usenet news [15]. In the CF system in [9], they monitored reading times as an indicator for relevance. This revealed a relationship between time spent on reviewing data items and their relevance. In [6], the system learns the user profile by passively observing the hyperlinks clicked on and those passed over and by measuring user mouse and scrolling activity in addition to user browsing activity. Also, in [14] they utilize agents that operate as adaptive Web site RS. Through analysis of Web logs and web page structure, the agents infer knowledge of the popularity of various documents as well as a combination of document similarity. By tracking user actions and his/her acceptance of the agent recommendations, the agent can make further estimations about future recommendations to the specific user. The main benefits of implicit feedback over explicit ratings are that they remove the cognitive cost of providing relevance judgements explicitly and can be gathered in large quantities and aggregated to infer item relevance [8].

However, the implicit approach bears some serious implications. For instance, some purchases are gifts and, thus, do not reflect the active user interests. Moreover, the inference that purchasing implies liking does not always hold. Owing to the difficulty of acquiring explicit ratings, some providers of product recommendation services adopt bilateral approaches. For instance, Amazon.com computes recommendations based on explicit ratings whenever possible. In case of unavailability, observed implicit ratings are used instead.

1.3.2 The Explicit Approach

Users are required to explicitly specify their preference for any particular item, usually by indicating their extent of appreciation on 5-point or 7-point **Thurstone scales**. These scales are mapped to numeric values, e.g. $R_{i,j} \in [1, 2, 3, 4, 5]$. Lower values commonly indicate least favorable preferences, while higher values express the user's liking.[1] Explicit ratings impose additional efforts on users. Consequently, users often tend to avoid the burden of explicitly stating their preferences and either leave the system or rely upon "free-riding" [2]. Ratings made on these scales allow these judgments to be processed statistically to provide averages, ranges, or distributions. A central feature of explicit ratings is that the user who evaluates items has to examine them and, then, to assign to them values from the rating scale. This imposes a cognitive cost on the evaluator to assess the performance of an object [12].

1.3.3 The Mixing Approach

Newsweeder [10], a Usenet filtering system, is an example of a system that uses a combination of the explicit and the implicit approach, as it requires minimum user involvement. In this system, the users are required to rate documents for their relevance. The ratings are used as training examples for a machine learning algorithm that is executed nightly to generate user interest profiles for the next day. Newsweeder is successful in reducing user involvement. However, the batch profiling used in Newsweeder is a shortcoming as profile adaptation is delayed significantly.

1.4 Motivation of the Book

The motivation of this book is based on the following facts that constitute important open research problems in RS. It is well known that users hardly provide explicit feedbacks in RS. More specifically, users tend to provide ratings only for items that they are interested in and belong to their preferences and avoid, to provide feedback in the form of negative examples, i.e. items that they dislike or they are not interested in. As stated in [5, 17], "It has been known for long time in human computer interaction that users are extremely reluctant to perform actions that are not directed towards their immediate goal if they do not receive immediate benefits". However, common RS based on machine learning approaches use classifiers that, in order to learn user interests, require both positive (desired items that users prefer) and

[1] The Thurstone scale was used in psychology for measuring an attitude. It was developed by Louis Leon Thurstone in 1928, as a means of measuring attitudes towards religion. It is made up of statements about a particular issue. A numerical value is associated with each statement, indicating how favorable or unfavorable the statement is judged to be.

negative examples (items that users dislike or are not interested in). Additionally, the effort for collecting negative examples is arduous as these examples should uniformly represent the entire set of items, excluding the class of positive items. Manually collecting negative samples could be biased and require additional effort by users. Moreover, especially in web applications, users consider it very difficult to provide personal data and rather avoid to be related with internet sites due to lack of faith in the privacy of modern web sites [5, 17]. Therefore, RS based on demographic data or stereotypes that resulted from such data are very limited since there is a high probability that the user-supplied information suffers from noise induced by the fact that users usually give fake information in many of these applications.

Thus, machine learning methods need to be used in RS, that utilize only positive examples provided by users without additional information either in the form of negative examples or in the form of personal information for them. PEBL [19] is an example of a RS to which only positive examples are supplied by its users. Specifically, PEBL is a web page classification approach that works within the framework of learning based only on positive examples and uses the mapping-convergence algorithm combined with SVM.

On the other hand, user profiles can be either explicitly obtained from user ratings or implicitly learnt from the recorded user interaction data (i.e. user play-lists). In the literature, collaborative filtering based on explicit ratings has been widely studied while binary collaborative filtering based on user interaction data has been only partially investigated. Moreover, most of the binary collaborative filtering algorithms treat the items that users have not yet played/watched as the "un-interested in" items (negative class), which, however, is a practically invalid assumption.

Collaborative filtering methods assume availability of a range of high and low ratings or multiple classes in the data matrix of Users-Items. One-class collaborative filtering proposed in [13] provides weighting and sampling schemes to handle one-class settings with unconstrained factorizations based on the squared loss. Essentially, the idea is to treat all non-positive user-item pairs as negative examples, but appropriately control their contribution in the objective function via either uniform, user-specific or item-specific weights.

Thereby, we must take into consideration that the recommendation process could not only be expanded in a classification scheme about users' preferences as in [19], but should also take into account the opinion of other users in order to eliminate the problem of "local optima" of the content-based approaches [5, 17]. On the other hand, pure collaborative approaches have the main drawback that they tend to recommend items that could possibly be biased by a group of users and to ignore information that could be directly related to item content and a specific user's preferences. Thus, an approach is required that pays particular attention to the above matters.

Most of the existing recommendation methods have as a goal to provide accurate recommendations. However, an important factor for a RS is its ability to adapt according to user perception and to provide a kind of justification to a recommendation which allow its recommendations to be accepted and trusted by users. Recommendations based only on ratings, without taking into account the content of the recommended items fail to provide qualitative justifications. As stated in [4], "when

the users can understand the strengths and limitations of a RS, the acceptance of its recommendations is increased." Thus, new methods are needed that make enhanced use of similarity measures to provide both individualization and an indirect way for justifications for the items that are recommended to the users.

1.5 Contribution of the Book

The contribution of this book is two-fold. The first contribution develops, presents and evaluates a content-based RS based on multiple similarity measures that attempt to capture user *perception* of similarity and to provide individualization and justifications of recommended items according to the similarity measure that was assigned to each user. Specifically, a content-based RS, called MUSIPER,[2] is presented which constructs music similarity perception models of its users by associating different similarity measures with different users. Specifically, a user-supplied relevance feedback procedure and related neural network-based incremental learning allow the system to determine which subset of a full set of objective features approximates more accurately the subjective music similarity perception of a specific user. Our implementation and evaluation of MUSIPER verifies the relation between subsets of objective features and individualized music similarity perception and exhibits significant improvement in individualized perceived similarity in subsequent recommended items. Additionally, the investigation of the relation between objective feature subsets and user perception offers an explanation and justification for the items one selects.

The selection of the objective feature subsets in MUSIPER was based on semantic categorization of the features in a way that formed groups of features that reflect semantically different aspects of the music signal. This semantic categorization helped us to formulate indirect relations between a user's specific perception and corresponding item similarity (in this case, music similarity) that involves his/her preferences. Thus, the selected features in a specific feature subset provides a justification-reasoning for the factors that influence the specific user's perception of similarity between objects and, consequently, for his/her preferences. As it was observed, *no* single feature subset outperformed the other subsets for all uses. Moreover, it was experimentally observed that the users of MUSIPER were clustered by the eleven feature subsets in MUSIPER into eleven corresponding clusters. It was also observed that, in this clustering scheme, empty user clusters appeared, which implies that the corresponding feature subsets failed to model the music similarity perception of any user at all. On the other hand, there were other feature subsets the corresponding clusters of which contained approximately 27 and 18 % of the users of MUSIPER. These two findings are indicative of the effect of qualitative differences of the corresponding feature subsets. They provide strong evidence justifying our initial hypothesis that relates feature subsets with the similarity perception of an individual.

[2]MUSIPER is an acronym that stands for MUsic SImilarity PERception.

Additionally, they indicate that users tend to concentrate around particular factors (features) that eventually influence their perception of item similarity and corresponding item preferences.

The second contribution of this book concerns the development and evaluation of a hybrid cascade RS that utilizes only positive examples from a user Specifically, a content-based RS is combined with collaborative filtering techniques in order primarily to predict ratings and secondly to exploit the content-based component to improve the quality of recommendations. Our approach focuses on:

1. using only positive examples provided by each user and
2. avoiding the "local optima" of the content-based RS component that tends to recommend only items that a specific user has already seen without allowing him/her to view the full spectrum of items. Thereby, a need arises for enhancement of collaborative filtering techniques that combine interests of users that are comparable to the specific user.

Thus, we decompose the recommendation problem into a two-level cascaded recommendation scheme. In the first level, we formulate a one-class classification problem based on content-based features of items in order to model the individualized (subjective) user preferences into the recommendation process. In the second level, we apply either a content-based approach or a collaborative filtering technique to assign a corresponding rating degree to these items. Our realization and evaluation of the proposed cascade hybrid recommender approach demonstrates its efficiency clearly. Our recommendation approach benefits from both content-based and collaborative filtering methodologies. The content-based level eliminates the drawbacks of the pure collaborative filtering that do not take into account the subjective preferences of an individual user, as they are biased towards the items that are most preferred by the remaining users. On the other hand, the collaborative filtering level eliminates the drawbacks of the pure content-based recommender which ignores any beneficial information related to users with similar preferences. The combination of the two approaches into a cascade form mimics the social process where someone has selected some items according to his/her preferences and, to make a better selection, seeks opinions about these from others.

1.6 Outline of the Book

The book is organized as follows:

In Chap. 2, related works are presented on approaches to address fundamental problems of RS. In Chap. 3, the general problem and key definitions, paradigms, and results are presented of the scientific discipline of learning, with particular emphasis on machine learning. More specifically, we focus on statistical learning and the two main paradigms that have developed in statistical inference: the parametric paradigm and the general non-parametric paradigm. We concentrate our analysis on classification problems solved with the use of Support Vector Machines (SVM) as

applicable to our recommendation approaches. Particularly, we summarize the One-Class Classification approach and the application of One-Class SVM Classification to the recommendation problem.

Next, Chap. 4 presents features that are utilized to analyze the content of multimedia data. Specifically, we present the MPEG-7 framework which forms a widely adopted standard for processing multimedia files. Additionally, we present the MARSYAS framework for extraction of features from audio files.

In Chap. 5, the content-based RS, called MUSIPER, is presented and analyzed. MUSIPER uses multiple similarity measures in order to capture the perception of similarity of different users and to provide individualization and justifications for items recommended according to the similarity measure assigned to each user.

In the following two Chaps. 6 and 7, we present our cascade recommendation methods based on a two-level combination of one-class SVM classifiers with collaborative filtering techniques.

Finally, we summarize the book, draw conclusions and point to future related research work in Chap. 8.

References

1. Adomavicius, G., Tuzhilin, E.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**, 734–749 (2005)
2. Avery, C., Zeckhauser, R.: Recommender systems for evaluating computer messages. Commun. ACM **40**(3), 88–89 (1997). doi:10.1145/245108.245127
3. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 43–52. Morgan Kaufmann (1998)
4. Herlocker, J.L., Konstan, J.A., Riedl, J.: Explaining collaborative filtering recommendations. In: Proceedings of the 2000 ACM Conference on ComputeR Supported Cooperative Work CSCW'00, pp. 241–250. ACM, New York (2000). doi:10.1145/358916.358995
5. Ingo, S., Alfred, K., Ivan, K.: Learning user interests through positive examples using content analysis and collaborative filtering (2001). http://citeseer.ist.psu.edu/schwab01learning.html
6. Jude, J.G., Shavlik, J.: Learning users' interests by unobtrusively observing their normal behavior. In: Proceedings of International Conference on Intelligent User Interfaces, pp. 129–132. ACM Press (2000)
7. Karypis, G.: Evaluation of item-based top-n recommendation algorithms. In: Proceedings of the Tenth International Conference on Information and Knowledge Management CIKM'01, pp. 247–254. ACM, New York (2001). doi:10.1145/502585.502627
8. Kelly, D., Teevan, J.: Implicit feedback for inferring user preference: a bibliography. SIGIR Forum **37**(2), 18–28 (2003). doi:10.1145/959258.959260
9. Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R., Riedl, J.: GroupLens: applying collaborative filtering to usenet news. Commun. ACM **40**(3), 77–87 (1997)
10. Lang, K.: Newsweeder: learning to filter netnews. In: Proceedings of 12th International Machine Learning Conference (ML95), pp. 331–339 (1995)
11. Malone, T.W., Grant, K.R., Turbak, F.A., Brobst, S.A., Cohen, M.D.: Intelligent information-sharing systems. Commun. ACM **30**(5), 390–402 (1987). doi:10.1145/22899.22903
12. Nichols, D.M.: Implicit rating and filtering. In: Proceedings of the Fifth DELOS Workshop on Filtering and Collaborative Filtering, pp. 31–36 (1997)

13. Pan, R., Zhou, Y., Cao, B., Liu, N.N., Lukose, R., Scholz, M., Yang, Q.: One-class collaborative filtering. In: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining ICDM'08, pp. 502–511. IEEE Computer Society, Washington (2008). doi:10.1109/ICDM.2008.16
14. Pazzani, M.J.: A framework for collaborative, content-based and demographic filtering. Artif. Intell. Rev. **13**(5–6), 393–408 (1999). doi:10.1023/A:1006544522159
15. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: GroupLens: an open architecture for collaborative filtering of netnews. In: Proceedings of Computer Supported Collaborative Work Conference, pp. 175–186. ACM Press (1994)
16. Resnick, P., Varian, H.R.: Recommender systems. Commun. ACM **40**(3), 56–57 (1997)
17. Schwab, I., Pohl, W., Koychev, I.: Learning to recommend from positive evidence. In: Proceedings of the 5th International Conference on Intelligent User Interfaces IUI '00, pp. 241–247. ACM, New York (2000). doi:10.1145/325737.325858
18. Ungar, L., Foster, D., Andre, E., Wars, S., Wars, F.S., Wars, D.S., Whispers, J.H.: Clustering methods for collaborative filtering. In: Proceedings of AAAI Workshop on Recommendation Systems. AAAI Press (1998)
19. Yu, H., Han, J., Chang, K.C.C.: PEbL: web page classification without negative examples. IEEE Trans. Knowl. Data Eng. **16**(1), 70–81 (2004). doi:10.1109/TKDE.2004.1264823

Chapter 2
Review of Previous Work Related to Recommender Systems

Abstract The large amount of information resources that are available to users imposes new requirements on the software systems that handle the information. This chapter reviews the state of the art of the main approaches to designing RSs that address the problems caused by information overload. In general, the methods implemented in a RS fall within one of the following categories: (a) Content-based Methods, (b) Collaborative Methods and (c) Hybrid Methods.

2.1 Content-Based Methods

Modern information systems embed the ability to monitor and analyze users' actions to determine the best way to interact with them. Ideally, each users actions are logged separately and analyzed to generate an individual user profile. All the information about a user, extracted either by monitoring user actions or by examining the objects the user has evaluated [9], is stored and utilized to customize services offered. This user modeling approach is known as *content-based learning*. The main assumption behind it is that a user's behavior remains unchanged through time; therefore, the content of past user actions may be used to predict the desired content of their future actions [4, 27]. Therefore, in content-based recommendation methods, the rating $R(u, i)$ of the item i for the user u is typically estimated based on ratings assigned by user u to the items $I_n \in I$ that are "similar" to item i in terms of their content, as defined by their associated features.

To be able to search through a collection of items and make observations about the similarity between objects that are not directly comparable, we must transform raw data at a certain level of information granularity. Information granules refer to a collection of data that contain only essential information. Such granulation allows more efficient processing for extracting features and computing numerical representations that characterize an item. As a result, the large amount of detailed information of one item is reduced to a limited set of features. Each feature is a vector of low dimensionality, which captures some aspects of the item and can be used to determine item similarity. Therefore, an item i could be described by a feature vector

$$F(i) = [\, feature_1(i), feature_2(i), feature_3(i), \ldots feature_n(i)]. \tag{2.1}$$

A.S. Lampropoulos and G.A. Tsihrintzis, *Machine Learning Paradigms*,
Intelligent Systems Reference Library 92, DOI 10.1007/978-3-319-19135-5_2

For example, in a music recommendation application, in order to recommend music files to user u, the content-based RS attempts to build a profile of the user's preferences based on features presented in music files that the user u has rated with high rating degrees. Consequently, only music files that have a high degree of similarity with these highly rated files would be recommended to the user. This method is known as "item-to-item correlation" [41]. The type of user profile derived by a content-based RS depends on the learning method which is utilized by the system. This approach to the recommendation process has its roots in information retrieval and information filtering [3, 36]. Retrieval-based approaches utilize interactive learning techniques such as *relevance feedback* methods, in order to organize and retrieve data in an effective personalized way. In relevance feedback methods, the user is part of the item-management process, which means that the user evaluates the results provided by the system. Then, the system adapts, its performance according to the user's preferences. In this way, the method of relevance feedback has the efficiency not only to take into account the user subjectivity in perceiving the content of items, but also to eliminate the gap between high-level semantics and low-level features which are usually used for the content description of items [12, 13, 35].

Besides the heuristics that are based mostly on information retrieval methods [3, 12, 13, 35, 36] such as the Rocchio algorithm or correlation-based schemes, other techniques for content-based recommendation utilize Pattern Recognition/Machine Learning approaches, such as Bayesian classifiers [28], clustering methods, decision trees, and artificial neural networks.

These techniques differ from information retrieval-based approaches as they calculate utility predictions based not on a heuristic formula, such as a cosine similarity measure, but rather are based on a model learnt from the underlying data using statistical and machine learning techniques. For example, based on a set of Web pages that were rated by the user as "relevant" or "irrelevant," the naive Bayesian classifier is used in [28] to classify unrated Web pages.

Some examples of content-based methods come from the area of music data. In [10, 19, 24, 25, 47], they recommend pieces that are similar to users' favorites in terms of music content such as mood and rhythm. This allows a rich artist variety and various pieces, including unrated ones, to be recommended. To achieve this, it is necessary to associate user preferences with music content by using a practical database where most users tend to rate few pieces as favorites.

A relevance feedback approach for music recommendation was presented in [19] and based on the TreeQ vector quantization process initially proposed by Foote [14]. More specifically, relevance feedback was incorporated into the **user model** by modifying the quantization weights of desired vectors. Also, a relevance feedback music retrieval system, based on SVM Active Learning, was presented in [25], which retrieves the desired music piece according to mood and style similarity.

In [2], the authors explore the relation between the user's rating input, musical pieces with high degree of rating that were defined as the listener's favorite music, and music features. Specifically, labeled music pieces from specific artists were analyzed in order to build a correlation between user ratings and artists through music features. Their system forms the user profile as preference for music pieces of a specific artist.

They confirmed that favorite music pieces were concentrated along certain music features.

The system in [52] proposes the development of a user-driven similarity function by combining timbre-, tempo-, genre-, mood-, and year-related features into the overall similarity function. More specifically, similarity is based on a weighted combination of these features and the end-user can specify his/her personal definition of similarity by weighting them.

The work in [15] tries to extend the use of signal approximation and characterization from genre classification to recognition of user taste. The idea is to learn music preferences by applying instance-based classifiers to user profiles. In other words, this system does *not* build an individual profile for every user, but instead tries to recognize his/her favorite genre by applying instance-based classifiers to user rating preferences by his/her music playlist.

2.2 Collaborative Methods

CF methods are based on the assumption that similar users prefer similar items or that a user expresses similar preferences for similar items. Instead of performing content indexing or content analysis, CF systems rely entirely on interest ratings from the members of a participating community [18]. CF methods are categorized into two general classes, namely *model-based* and *memory-based* [1, 7].

Model-based algorithms use the underlying data to learn a probabilistic model, such as a cluster model or a Bayesian network model [7, 53], using statistical and machine learning techniques. Subsequently, they use the model to make predictions. The clustering model [5, 51] works by clustering similar users in the same class and estimating the probability that a particular user is in a particular class. From there, the clustering model computes the conditional probability of ratings.

Memory-based methods, store raw preference information in computer memory and access it as needed to find similar users or items and to make predictions. In [29], CF was formulated as a classification problem. Specifically, based on a set of user ratings about items, they try to induce a model for each user that would allow the classification of unseen items into two or more classes, each of which corresponds to different points in the accepted rating scale.

Memory-based CF methods can be further divided into two groups, namely user-based and item-based [37] methods. On the one hand, user-based methods look for users (also called "neighbors") similar to the active user and calculate a predicted rating as a weighted average of the neighbor's ratings on the desired item. On the other hand, item-based methods look for similar items for an active user.

2.2.1 User-Based Collaborative Filtering Systems

User-based CF systems are systems that utilize **memory-based algorithms**, meaning that they operate over the entire user-item matrix R, to make predictions. The majority

of such systems mainly deal with **user-user similarity calculations**, meaning that they utilize user neighborhoods, constructed as collections of similar users. In other words, they deal with the rows of the user-item matrix, R, in order to generate their results. For example, in a personalized music RS called RINGO [43], similarities between the tastes of different users are utilized to recommend music items. This user-based CF approach works as follows: A new user is matched against the database to discover neighbors, which are other customers who, in the past, have had a similar taste as the new user, i.e. who have bought similar items as the new user. Items (unknown to the new user) that these neighbors like are then recommended to the new user. The main steps of this process are:

1. Representation of Input data,
2. Neighborhood Formation, and
3. Recommendation Generation.

2.2.1.1 Representation of Input Data

To represent input data, one needs to define a set of ratings of users into a user-item matrix, R, where each $R(u, i)$ represents the rating value assigned by the user u to the item i. As users are not obligated to provide their opinion for all items, the resulting user-item matrix may be a **sparse matrix**. This sparsity of the user-item matrix is the main reason causing filtering algorithms not to produce satisfactory results. Therefore, a number of techniques were proposed to reduce the sparsity of the initial user-item matrix to improve the efficiency of the RS. *Default Voting* is the simplest technique used to reduce sparsity. A default rating value is inserted to items for which there does not exist a rating value. This rating value is selected to be neutral or somewhat indicative of negative preferences for unseen items [7].

An extension of the method of Default Voting is to use either the *User Average Scheme* or the *Item Average Scheme* or the *Composite Scheme* [39]. More specifically:

- In the *User Average Scheme*, for each user, u, the average user rating over all the items is computed, $\overline{R}(u)$. This is expressed as the average of the corresponding row in the user-item matrix. The user average is then used to replace any missing $R(u, i)$ value. This approach is based on the idea that a user's rating for a new item could be simply predicted if we take into account the same user's past ratings.
- In the *Item Average Scheme*, for each item, the item average over all users is computed, $\overline{R}(i)$. This is expressed as the average of the corresponding column in the user-item matrix. The item average is then used as a fill-in for missing values $R(u, i)$ in the matrix.
- In the *Composite Scheme*, the collected information for items and users both contribute to the final result. The main idea behind this method is to use the average of user u on item i as a base prediction and then add a correction term to it based on how the specific item was rated by other users.

The scheme works as follows: When a missing entry regarding the rating of user u on item i is located, initially, the user average $\overline{R}(u)$ is calculated as the average of the corresponding user-item matrix row. Then, we search for existing ratings in the column which correspond to item i. Assuming that a set of l users, $U = \{u_1, u_2, \ldots, u_l\}$, has provided a rating for item i, we can compute a correction term for each user $u \in L$ equal to $\delta_k = R(u_k, i) - \overline{R}(u_k)$. After the corrections for all users in U are computed, the composite rating can be calculated as:

$$R(u,i) = \begin{cases} \overline{R}(u) + \frac{\sum_{k=1}^{l} \delta_k}{l}, & \text{if user } u \text{ has not rated item } i \\ R, & \text{if user } u \text{ has rated item } i \text{ with } R. \end{cases} \tag{2.2}$$

An alternative way of utilizing the composite scheme is through a simple transposition: first compute the item average, $\overline{R}(i_k)$, (i.e., average of the column which corresponds to item i) and then compute the correction terms, δ_k, by scanning through all l items $I = \{i_1, i_2, \ldots, i_l\}$ rated by user k. The fill-in value of $R(u, i)$ would then be:

$$R(u,i) = \overline{R}(i) + \frac{\sum_{k=1}^{l} \delta_k}{l}, \tag{2.3}$$

where l is the count of items rated by user u and the correction terms are computed for all items in I as $\delta_k = R(u, i_k) - \overline{R}(i_k)$

After generating a reduced-dimensionality matrix, we could use a vector similarity metric to compute the proximity between users and hence to form *neighborhoods of users* [38], as discussed in the following.

2.2.1.2 Neighborhood Formation

In this step of the recommendation process, the *similarity* between users is calculated in the user-item matrix, R, i.e., users similar to the active user, u_a, form a proximity-based neighborhood with him. More specifically, neighborhood formation is implemented in two steps: Initially, the similarity between all the users in the user-item matrix, R, is calculated with the help of some proximity metrics. The second step is the actual neighborhood generation for the active user, where the similarities of users are processed in order to select those users that will constitute the neighborhood of the active user. To find the similarity between users u_a and u_b, we can utilize the *Pearson correlation metric*. The Pearson correlation was initially introduced in the context of the GroupLens project [33, 43], as follows: Let us assume that a set of m users u_k, where $k = 1, 2, \ldots, m$, $U_m = \{u_1, u_2, \ldots, u_m\}$, have provided a rating $R(u_k, i_l)$ for item i_l, where $l = 1, 2, \ldots, n$, $I_n = \{i_1, i_2, \ldots, i_n\}$ is the set of items. The Pearson correlation coefficient is given by:

$$sim(u_a, u_b) = \frac{\sum_{l=1}^{n} (R(u_a, i_l) - \overline{R}(u_a))(R(u_b, i_l) - \overline{R}(u_b))}{\sqrt{\sum_{l=1}^{n} (R(u_a, i_l) - \overline{R}(u_a))^2 \sum_{l=1}^{n} (R(u_b, i_l) - \overline{R}(u_b))^2}}. \quad (2.4)$$

Another metric similarity uses the *cosine-based approach* [7], according to which the two users u_a and u_b, are considered as two vectors in *n-dimensional* item-space, where $n = |I_n|$. The similarity between two vectors can be measured by computing the cosine angle between them:

$$sim(u_a, u_b) = \cos(\overrightarrow{u_a}, \overrightarrow{u_b}) = \frac{\sum_{l=1}^{n} R(u_a, i_l) R(u_b, i_l)}{\sqrt{\sum_{l=1}^{n} R(u_a, i_l)^2} \sqrt{\sum_{l=1}^{n} R(u_b, i_l)^2}}. \quad (2.5)$$

In RS, the use of the Pearson correlation similarity metric to estimate the proximity among users performs better than the cosine similarity [7].

At this point in the recommendation process, a single user is selected who is called the *active user*. The active user is the user for whom the RS will produce predictions and proceed with generating his/her neighborhood of users. A *similarity matrix* S is generated, containing the similarity values between all users. For example, the ith row in the similarity matrix represents the similarity between user u_i and all the other users. Therefore, from this similarity matrix S various schemes can be used in order to select the users that are most similar to the active user. One such scheme is the *center-based scheme*, in which from the row of the active user u_a are selected those users who have the highest similarity value with the active user.

Another scheme for neighborhood formation is the *aggregate neighborhood formation scheme*. In this scheme, a neighborhood of users is created by finding users who are closest to the *centroid* of the current neighborhood and not by finding the users who are closest to the active user himself/herself. This scheme allows all users to take part in the formation of the neighborhood, as they are gradually selected and added to it.

2.2.1.3 Generation of Recommendations

The generation of recommendations is represented by predicting a rating, i.e., by computing a numerical value which constitutes a predicted opinion of the active user u_a for an item i_j unseen by him/her. This predicted value should be within the same accepted numerical scale as the other ratings in the initial user-item matrix R. In the generation of predictions, only those users participate that lie within the neighborhood of the active user. In other words, only a subset of k users participate

from the m users in the set U_m that have provided ratings for the specific item i_j, $U_k \subseteq U_m$. Therefore, a *prediction score* P_{u_a,i_j} is computed as follows [33]:

$$P_{u_a,i_j} = \overline{R}(u_a) + \frac{\sum_{t=1}^{k} (R(u_t, i_j) - \overline{R}(u_t)) * sim(u_a, u_t)}{\sum_{t=1}^{k} |sim(u_a, u_t)|}, \quad \text{where } U_k \subseteq U_l \quad (2.6)$$

Here, $\overline{R}(u_a)$ and $\overline{R}(u_t)$ are the average rating of the active user u_a and u_t, respectively, while $R(u_t, i_j)$ is the rating given by user u_t to item i_j. Similarity $sim(u_a, u_t)$ is the similarity among users u_a and u_t, computed using the Pearson correlation in Eq. 2.4. Finally, the RS will output several items with the best predicted ratings as the recommendation list.

An alternative output of a RS is the *top-N recommendations* output. In this case, recommendations form a list of N items that the active user is expected to like the most. For the generation of this list, users are ranked first according to their similarity to the active user. The k most similar (i.e. most highly ranked) users are selected as the k-nearest neighbors of the active user u_a. The frequency count of an item is calculated by scanning the rating of the item by the k-nearest neighbors. Then, the items are sorted based on frequency count. The N most frequent items that have not been rated by the active user are selected as the top-N recommendations [23].

2.2.2 Item-Based Collaborative Filtering Systems

A different approach [20, 37] is based on item relations and not on user relations, as in classic CF. Since the relationships between users are relatively dynamic, as they continuously buy new products, it is computationally hard to calculate the user-to-user matrix online. This causes the user-based CF approach to be relatively expensive in terms of computational load. In the item-based CF algorithm, we look into the set of items, denoted by I_{u_a}, that the active user, u_a, has rated and compute how similar they are to the target item i_t. Then, we select the k most similar items $I_k = \{i_1, i_2, \ldots, i_k\}$, based on their corresponding similarities $\{sim(i_t, i_1), sim(i_t, i_2), \ldots, sim(i_t, i_k)\}$. The predictions can then be computed by taking a weighted average of the active user's ratings on these similar items. The main steps in this approach are the same as in user-based CF. The difference in the present approach is that instead of calculating similarities between two users who have provided ratings for a common item, we calculate similarities between two items i_t, i_j which have been rated by a common user u_a. Therefore, the Pearson correlation coefficient and cosine similarity are, respectively, given as:

$$sim(i_t, i_j) = \frac{\sum_{l=1}^{n} (R(u_l, i_t) - \overline{R}(i_t))(R(u_l, i_j) - \overline{R}(i_j))}{\sqrt{\sum_{l=1}^{n} (R(u_l, i_t) - \overline{R}(i_t))^2 \sum_{l=1}^{n} (R(u_l, i_j) - \overline{R}(i_j))^2}} \tag{2.7}$$

$$sim(i_t, i_j) = \cos(\overrightarrow{i_t}, \overrightarrow{i_j}) = \frac{\sum_{l=1}^{n} R(u_l, i_t) R(u_l, i_j)}{\sqrt{\sum_{l=1}^{n} R(u_l, i_t)^2} \sqrt{\sum_{l=1}^{n} R(u_l, i_j)^2}}. \tag{2.8}$$

Next, the similarities between all items in the initial user-item matrix, R, are calculated. The final step in the CF procedure is to isolate k items from n, ($I_k \subseteq I_n$) in order to share the greatest similarity with item i_t for which we are seeking a prediction, form its neighborhood of items, and proceed with prediction generation. A prediction on item i_t for active user u_a is computed as the sum of ratings given by the active user on items belonging to the neighborhood I_k. These ratings are weighted by the corresponding similarity, $sim(i_t, i_j)$ between item i_t and item i_j, with $j = 1, 2, \ldots, k$, taken from neighborhood I_k:

$$P_{u_a, i_j} = \frac{\sum_{j=1}^{k} sim(i_t, i_j) * R(u_a, i_j)}{\sum_{j=1}^{k} \left| sim(i_t, i_j) \right|} \text{ where } I_k \subseteq I_n. \tag{2.9}$$

In [16], the authors proposed that the long-term interest profile of a user (*task profile*) be established either by explicitly providing some items associated with the current task or by implicitly observing the user behavior (*intent*). By utilizing the item-to-item correlation matrix, items that resemble the items in the task profile are selected for recommendation. Since they match the task profile, these items fit the current task of the user. Before recommending them to the user, these items will be re-ranked to fit the user interests based on the interest prediction.

2.2.3 Personality Diagnosis

Personality diagnosis may be thought of as a hybrid between memory and model-based approaches of CF. The main characteristic is that predictions have meaningful probabilistic semantics. Moreover, this approach assumes that preferences constitute a characterization of their underlying personality type for each user. Therefore, taking into consideration the active user's known ratings of items, it is possible to estimate the probability that he/she has the same personality type with another user. The personality type of a given user is taken to be the vector of "true" ratings for items

the user has seen. A true rating differs from the actually reported rating given by a user by an amount of (Gaussian) noise. Given the personality type of a user, the personality diagnosis approach estimates the probability that the given user is of the same personality type as other users in the system, and, consequently, estimates the probability that the user will like some new item [30].

The personality type for each user u_k is formulated as follows, where $k = 1, 2, \ldots, m$, $U_m = \{u_1, u_2, \ldots, u_m\}$, and the user u_k has a number of preferred items in $I_n = \{i_1, i_2, \ldots, i_n\}$:

$$\overset{true}{R}(u_k) = \left\{ \overset{true}{R}(u_k, i_1), \overset{true}{R}(u_k, i_2), \ldots, \overset{true}{R}(u_k, i_n) \right\}. \tag{2.10}$$

Here, $\overset{true}{R}(u_k, i_l)$, with $i_l \in I_n$ and $l = 1, 2, \ldots, n$, stands for true rating by user u_k of the item i_l. It is important to note the difference between *true* and *reported* (given) ratings of the user. The true ratings encode the underlying internal preferences for a user that are *not* directly accessible by the designer of the RS. However, the reported ratings are those which were provided by users and utilized by the RS.

It is assumed that the reported ratings given by users include Gaussian noise. This assumption has the meaning that one user could report different ratings for the same items under different situations, depending on the context. Thus, we can assume that the rating reported by the user for an item i_l is drawn from an independent normal distribution with mean $\overset{true}{R}(u_k, i_l)$. Particularly:

$$Pr\left(R(u_k, i_l) = x \middle| \overset{true}{R}(u_k, i_l) = y \right) \propto e^{-\frac{(x-y)^2}{2\sigma^2}}, \tag{2.11}$$

where σ is a free parameter, x is the rating that the user has reported to the RS, and y is the true rating value that the user u_k would have reported if there no noise were present.

Furthermore, we assume that the distribution of personality types in the rating array R of users-items is representative of the personalities found in the target population of users. Therefore, taking into account this assumption, we can formulate the prior probability $Pr\left(\overset{true}{R}(u_a) = \upsilon\right)$ that the active user u_a rates items accordingly to a vector υ as given by the frequency that the other users rate according to υ. Thereby, instead of explicitly counting occurrences, we simply define $\overset{true}{R}(u_a)$ to be a random variable that can take one of m values, $(R(u_1), R(u_2), \ldots, R(u_m))$, each with probability $\frac{1}{m}$:

$$Pr\left(\overset{true}{R}(u_a) = R(u_k)\right) = \frac{1}{m}. \tag{2.12}$$

Combining Eqs. 2.11 and 2.12 and given the active user's ratings, we can compute the probability that the active user is of the same personality type as any other user,

by applying the Bayes rule:

$$\begin{aligned} & Pr\left(\overset{true}{R}(u_a) = R(u_k) | R(u_a, i_1) = x_1, \ldots, R(u_a, i_n) = x_n\right) \\ & \propto Pr\left(R(u_a, i_1) = x_1 | \overset{true}{R}(u_a, i_1) = R(u_a, i_1)\right) \\ & \ldots Pr\left(R(u_a, i_n) = x_n | \overset{true}{R}(u_a, i_n) = R(u_a, i_n)\right) \\ & \cdot Pr\left(\overset{true}{R}(u_a) = R(u_k)\right). \end{aligned} \tag{2.13}$$

Hence, computing this quantity for each user u_k, we can compute the probability distribution for the active user's rating of an unseen item i_j. This probability distribution corresponds to the prediction P_{u_a,i_j} produced by the RS and equals the expected rating value of active user u_a for the item i_j:

$$\begin{aligned} P_{u_a,i_j} &= Pr\left(R(u_a, i_j) = x_j | R(u_a, i_1) = x_1, \ldots, R(u_a, i_n) = x_n\right) \\ &= \sum_{k=1}^{m} \Pr\left(R(u_a, i_j) = x_j | \overset{true}{R}(u_a) = R(u_k)\right) \\ &\cdot \Pr\left(\overset{true}{R}(u_a) = R(u_k) | R(u_a, i_1) = x_1, \ldots, R(u_a, i_n) = x_n\right). \end{aligned} \tag{2.14}$$

The model is depicted as a naive Bayesian network with the structure of a classical diagnostic model as follows:

- Firstly, we observe ratings and, using Eq. 2.13, compute the probability that each personality type is the cause. Ratings can be considered as "symptoms" while personality types as "diseases" leading to those symptoms in the diagnostic model.
- Secondly, we can compute the probability of rating values for an unseen item using Eq. 2.14. The most probable rating is returned as the prediction of the RS.

An alternative interpretation of personality diagnosis is to consider it as a clustering method with exactly one user per cluster. This is so because each user corresponds to a single personality type and the effort is to assign the active user to one of these clusters [7, 51].

An additional interpretation of personality diagnosis is that the active user is assumed to be "generated" by choosing one of the other users uniformly at random and adding Gaussian noise to his/her ratings. Given the active user's known ratings, we can infer the probability that he/she be actually one of other users and then compute probabilities for ratings of other items.

2.3 Hybrid Methods

Hybrid methods combine two or more recommendation techniques to achieve better performance and to take out drawbacks of each technique separately. Usually, CF

methods are combined with content-based methods. According to [1], hybrid RS could be classified into the following categories:

- Combining Separate Recommenders
- Adding Content-Based Characteristics to Collaborative Models
- Adding Collaborative Characteristics to Content-Based Models
- A Single Unifying Recommendation Model.

Combining Separate Recommenders

The Hybrid RS of this category include two separate systems, a collaborative one and a content-based one. There are four different ways of combining these two separate systems, namely the following:

- *Weighted Hybridization Method*. The outputs (ratings) acquired by individual RS are combined together to produce a single final recommendation using either a linear combination [11] or a voting scheme [29]. The *P-Tango* system [11] initially gives equal weights to both recommenders, but gradually adjusts the weights as predictions about user ratings are confirmed or not. The system keeps the two filtering approaches separate and this allows the benefit from individual advantages.
- *Switched Hybridization Method*. The system switches between recommendation techniques selecting the method that gives better recommendations for the current situation depending on some recommendation "quality" metric. A characteristic example of such a recommender is *The Daily Learner* [6], which selects the recommender sub-system that provides the higher level of confidence. Another example of this method is presented in [50] where either the content-based or the collaborative filtering technique is selected according to which of the two provided better consistency with past ratings of the user.
- *Mixed Hybridization Method*. In this method, the results from different recommender sub-systems are presented simultaneously. An example of such a recommender is given in [45] where they utilize a content-based technique based on textual descriptions of TV shows and collaborative information about users' preferences. Recommendations from both techniques are provided together in the final suggested program.
- *Cascade Hybridization Method*. In this method, one recommendation technique is utilized to produce a coarse ranking of candidates, while the second technique focuses only on those items for which additional refinement is needed. This method is more efficient than the weighted hybridization method which applies all of its techniques on all items. The computational burden of this hybrid approach is relatively small because recommendation candidates in the second level are partially eliminated in the first level. Moreover this method is more tolerant to noise in the operation of low-priority recommendations, since ratings of the high level recommender can only be refined, but never over-turned [9]. In other words, cascade hybridization methods can be analyzed into two sequential stages. The first stage (content-based method or knowledge-based/collaborative) selects intermediate recommendations. Then, the second stage (collaborative/content-based method

or knowledge-based) selects appropriate items from the recommendations of the first stage. Burke [8] developed a restaurant RS called *EntreeC*. The system first selects several restaurants that match a user's preferred cuisine (e.g., Italian, Chinese, etc.) with a knowledge-based method. In the knowledge-based method, the authors construct a feature vector according to defined attributes that characterize the restaurants. This method is similar to content-based methods; however, it must be noted that these metadata are content-independent and for this reason the term *knowledge-based* is utilized. These restaurants are then ranked with a collaborative method.

2.3.1 Adding Content-Based Characteristics to Collaborative Models

In [29], the authors proposed *collaboration via content*. This is a method that uses a prediction scheme similar to the standard CF, in which similarity among users is not computed on provided ratings, but rather on the content-based profile of each user. The underlying intuition is that like-minded users are likely to have similar content-based models and that this similarity relation can be detected without requiring overlapping ratings. The main limitation of this approach is that the similarity of users is computed using Pearson's correlation coefficient between content-based weight vectors.

On the other hand, in [26] the authors proposed the *content-boosted collaborative filtering* approach, which exploits a content-based predictor to enhance existing user data and then provides personalized suggestions through CF. The content-based predictor is applied to each row of the initial user-item matrix, corresponding to each user, and gradually generates a pseudo user-item matrix that is a full dense matrix. The similarity between the active user, u_a, and another user, u_i, is computed with CF using the new pseudo user-item matrix.

2.3.2 Adding Collaborative Characteristics to Content-Based Models

The main technique of this category is to apply dimensionality reduction on a group of content-based profiles. In [46], the authors used *latent semantic indexing* to create a collaborative view of a collection of user profiles represented as term vectors. This technique results in performance improvement in comparison with the pure content-based approach.

2.3.3 A Single Unifying Recommendation Model

A general unifying model that incorporates content-based and collaborative characteristics was proposed in [5], where the authors present the use of content-based and collaborative characteristics (e.g., the age or gender of users or the genre of movies) in a single rule-based classifier. Single unifying models were also presented in [31], where the authors utilized a unified probabilistic method for combining collaborative and content-based recommendations.

2.3.4 Other Types of Recommender Systems

Demographics-based RS. The basis for recommendations in demographics-based RS is the use of prior knowledge on demographic information about the users and their opinions for the recommended items. Demographics-based RS classify their users according to personal demographic data (e.g. age and gender) and classify items into user classes. Approaches falling into this group can be found in Grundy [34], a system for book recommendation, and in [21] for marketing recommendations. Similarly to CF, demographic techniques also employ user-to-user correlations, but differ in the fact that they do not require a history of user ratings. An additional example of a demographics-based RS is described in [29], in which information about users is taken from their home-pages to avoid the need to maintain a history of user ratings. Demographic characteristics for users (e.g. their age and gender) is also utilized in [5].

Knowledge-based RS. Knowledge-based RS use prior knowledge on how the recommended items fulfill the user needs. Thus, the goal of a knowledge-based RS is to reason about the relationship between a need and a possible recommendation. The user profile should encompass some knowledge structure that supports this inference. An example of such a RS is presented in [8], where the system *Entree* uses some domain knowledge about restaurants, cuisines, and foods to recommend a restaurant to its users. The main advantage using a knowledge-based system is that there is no bootstrapping problem. Because the recommendations are based on prior knowledge, there is no learning time before making good recommendations. However, the main drawback of knowledge-based systems is a need for knowledge acquisition for the specific domain which makes difficult the adaptation in another domain and not easily adapted to the individual user as it is enhanced by predefined recommendations.

2.4 Fundamental Problems of Recommender Systems

Cold Start Problem. The cold-start problem [42] is related to the learning rate curve of a RS. The problem could be analyzed into two different sub-problems:

- **New-User Problem**, i.e., the problem of making recommendations to a new user [32], where almost nothing is known about his/her preferences.
- **New-Item Problem**, i.e., the problem where ratings are required for items that have not been rated by users. Therefore, until the new item is rated by a satisfactory number of users, the RS would not be able to recommend this item. This problem appears mostly in collaborative approaches and could be eliminated with the use of content-based or hybrid approaches where content information is used to infer similarities among items.

  This problem is also related, with the *coverage* of a RS, which is a measure for the domain of items over which the system could produce recommendations. For example, low coverage of the domain means that only a limited space of items is used in the results of the RS and these results usually could be biased by preferences of other users. This is also known as the problem of *over-specialization*. When the system can only recommend items that score highly against a user's profile, the user is limited to being recommended items that are similar to those already rated. This problem, which has also been studied in other domains, is often addressed by introducing some randomness. For example, the use of genetic algorithms has been proposed as a possible solution in the context of information filtering [44].

Novelty Detection—Quality of Recommendations. From those items that a RS recommends to users, there are items that are already known to the users and items that are new (novel) and unknown to them. Therefore, there is a competitiveness between the desire for novelty and the desire for high quality recommendations. One hand, the quality of the recommendations [38] is related to "trust" that users express for the recommendations. This means that a RS should minimize false positive errors and, more specifically, the RS should not recommend items that are not desirable. On the other hand, novelty is related with the "timestamp—age" of items: the older items should be treated as less relevant than the newer ones and this causes increase to the novelty rate. Thus, a high novelty rate will produce poor quality recommendations because the users will not be able to identify most of the items in the list of recommendations.

Sparsity of Ratings. The sparsity problem [1, 22] is related to the unavailability of a large number of rated items for each active user. The number of items that are rated by users is usually a very small subset of those items that are totally available. For example, in *Amazon*, if the active users may have purchased 1 % of the items and the total amount of items is approximately 2 millions of books, this means that there are only 20,000 of books which are rated. Consequently, such sparsity in ratings degrades the accurate selection of the neighbors in the step of neighborhood formation and leads to poor recommendation results.

A number of possible solutions have been proposed to overcome the sparsity problem such as content-based similarities, item-based CF methods, use of demographic data and a number of hybrid approaches [9]. A different approach to deal with this problem is proposed in [40], where the authors utilized *dimension reduction techniques*, such as singular value decomposition, in order to transform the sparse

user-item matrix R into a dense matrix. The SVD is a method for matrix factorization that produces the best lower-rank approximations to the original matrix [29].

Scalability. RS, especially with large electronic sites, have to deal with a constantly growing number of users and items [7, 51]. Therefore, an increasing amount of computational resources is required as the amount of data grows. A recommendation method, that could be efficient when the number of data is limited, could be very time-consuming and scale poorly. Such a method would be unable to generate a satisfactory number of recommendations from a large amount of data. Thus, it is important that the recommendation approach be capable of scaling up in a successful manner [37].

Lack of Transparency Problem. RS are usually black boxes, which means that RS are not able to explain to their users why they recommend those specific items. In content-based approaches [47, 48], this problem could be minimized. However, in collaborative approaches, predictions may be harder to explain than predictions made by content-based models [17].

Gray Sheep User Problem. The majority of users falls into the class of so called "white-sheep", i.e. those who have high correlation with many other users. For these users, it should be easy to find recommendations. In a small or even medium community of users, there are users whose opinions do not consistently agree or disagree with any group of people [11]. There are users whose preferences are atypical (uncommon) and vary significantly from the norm. After neighborhood formation, these users will not have many other users as neighbors. As a result, there will be poor recommendations for them. From a statistical point of view, as the number of users of a system increases, so does the probability of finding other people with similar preferences, which means that better recommendations could be provided [49].

References

1. Adomavicius, G., Tuzhilin, E.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**, 734–749 (2005)
2. Arakawa, K., Odagawa, S., Matsushita, F., Kodama, Y., Shioda, T.: Analysis of listeners' favorite music by music features. In: Proceedings of the International Conference on Consumer Electronics (ICCE), pp. 427–428, IEEE (2006)
3. Baeza-Yates, R., Ribeiro-Neto, B. (eds.): Modern Information Retrieval. Addison-Wesley, New York (1999)
4. Balabanović, M., Shoham, Y.: Fab: content-based, collaborative recommendation. Commun. ACM **40**(3), 66–72 (1997). doi:10.1145/245108.245124
5. Basu, C., Hirsh, H., Cohen, W.: Recommendation as classification: using social and content-based information in recommendation. In: Proceedings of the Fifteenth National/Tenth Conference on Artificial intelligence/Innovative Applications of Artificial Intelligence AAAI'98/IAAI'98, pp. 714–720. American Association for Artificial Intelligence, Menlo Park (1998)
6. Billsus, D., Pazzani, M.J.: User modeling for adaptive news access. User Model. User-Adapt. Interact. **10**(2–3), 147–180 (2000). doi:10.1023/A:1026501525781

7. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 43–52. Morgan Kaufmann (1998)
8. Burke, R.: Knowledge-based recommender systems (2000)
9. Burke, R.: Hybrid recommender systems: survey and experiments. User Model. User-Adapt. Interact. **12**(4), 331–370 (2002). doi:10.1023/A:1021240730564
10. Celma, O., Ramrez, M., Herrera, P.: Foafing the music: a music recommendation system based on RSS feeds and user preferences. In: Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR). London (2005)
11. Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., Sartin, M.: Combining content-based and collaborative filters in an online newspaper. In: Proceedings of ACM SIGIR Workshop on Recommender Systems (1999)
12. Cox, I.J., Miller, M.L., Omohundro, S.M., Yianilos, P.N.: PicHunter: Bayesian relevance feedback for image retrieval. Int. Conf. Pattern Recognit., **3**, 361 (1996). doi:10.1109/ICPR.1996.546971
13. Doulamis, N.D., Doulamis, A.D., Varvarigou, T.A.: Adaptive algorithms for interactive multimedia. IEEE MultiMed. **10**(4), 38–47 (2003). doi:10.1109/MMUL.2003.1237549
14. Foote, J.: An overview of audio information retrieval. Multimed. Syst. **7**(1), 2–10 (1999)
15. Grimaldi, M., Cunningham, P.: Experimenting with music taste prediction by user profiling. In: Proceedings of Music Information Retrieval 2004 (MIR'04). New York (2004)
16. Herlocker, J.L., Konstan, J.A.: Content-independent task-focused recommendation. IEEE Internet Comput. **5**(6), 40–47 (2001). doi:10.1109/4236.968830
17. Herlocker, J.L., Konstan, J.A., Riedl, J.: Explaining collaborative filtering recommendations. In: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work CSCW'00, pp. 241–250. ACM, New York (2000). doi:10.1145/358916.358995
18. Herlocker, J.L., Konstan, J.A., Riedl, J.: An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. Inf. Retr. **5**(4), 287–310 (2002). doi:10.1023/A:1020443909834
19. Hoashi, K., Matsumo, K., Inoue, N.: Personalization of user profiles for content-based music retrieval based on relevance feedback. In: Proceedings of ACM International Conference on Multimedia 2003, pp. 110–119, ACM Press (2003)
20. Karypis, G.: Evaluation of item-based top-n recommendation algorithms. In: Proceedings of the Tenth International Conference on Information and Knowledge Management CIKM'01, pp. 247–254. ACM, New York (2001). doi:10.1145/502585.502627
21. Krulwich, B.: Lifestyle finder: intelligent user profiling using large-scale demographic data. AI Mag. **18**(2), 37–45 (1997)
22. Linden, G., Smith, B., York, J.: http://Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. **7**(1), 76–80 (2003). doi:10.1109/MIC.2003.1167344
23. Liu, D.R., Shih, Y.Y.: Integrating AHP and data mining for product recommendation based on customer lifetime value. Inf. Manag. **42**(3), 387–400 (2005). doi:10.1016/j.im.2004.01.008
24. Logan, B.: Music recommendation from song sets. In: Proceedings of 5th International Conference on Music Information Retrieval, pp. 425–428 (2004)
25. Mandel, M., Poliner, G., Ellis, D.: Support vector machine active learning for music retrieval ACM multimedia systems journal. ACM Multimed. Syst. J. **12**(1), 3–13 (2006)
26. Melville, P., Mooney, R.J., Nagarajan, R.: Content-boosted collaborative filtering for improved recommendations. In: Proceedings of Eighteenth National Conference on Artificial intelligence, pp. 187–192. American Association for Artificial Intelligence, Menlo Park (2002)
27. Mooney, R.J., Roy, L.: Content-based book recommending using learning for text categorization. In: Proceedings of the Fifth ACM Conference on Digital Libraries, pp. 195–204. ACM DL'00, New York (2000). doi:10.1145/336597.336662
28. Pazzani, M., Billsus, D.: Learning and revising user profiles: the identification of interesting web sites. Mach. Learn. **27**(3), 313–331 (1997). doi:10.1023/A:1007369909943
29. Pazzani, M.J.: A framework for collaborative, content-based and demographic filtering. Artif. Intell. Rev. **13**(5–6), 393–408 (1999). doi:10.1023/A:1006544522159

30. Pennock, D.M., Horvitz, E., Lawrence, S., Giles, C.L.: Collaborative filtering by personality diagnosis: a hybrid memory and model-based approach. In: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence UAI'00, pp. 473–480. Morgan Kaufmann Publishers Inc., San Francisco (2000)
31. Popescul, A., Ungar, L.H., Pennock, D.M., Lawrence, S.: Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence UAI'01, pp. 437–444. Morgan Kaufmann Publishers Inc., San Francisco (2001)
32. Rashid, A.M., Albert, I., Cosley, D., Lam, S.K., McNee, S.M., Konstan, J.A., Riedl, J.: Getting to know you: learning new user preferences in recommender systems. In: Proceedings of the 7th International Conference on Intelligent User Interfaces IUI'02, pp. 127–134. ACM, New York (2002). doi:10.1145/502716.502737
33. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: GroupLens: an open architecture for collaborative filtering of netnews. In: Proceedings of Computer Supported Collaborative Work Conference, pp. 175–186. ACM Press (1994)
34. Rich, E.: User Modeling via Stereotypes, pp. 329–342. Morgan Kaufmann Publishers Inc., San Francisco (1998)
35. Rui, Y., Huang, T.S., Ortega, M., Mehrotra, S.: Adaptive algorithms for interactive multimedia. IEEE Trans. Circuits Syst. Video Technol. **8**(5), 644–655 (1998)
36. Salton, G. (ed.): Automatic Text Processing. Addison-Wesley, Reading (1989)
37. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of 10th International Conference on World Wide Web, pp. 285–295. ACM, New York (2001). doi:10.1145/371920.372071
38. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Analysis of recommendation algorithms for e-commerce. In: Proceedings of 2nd ACM Conference on Electronic Commerce, pp. 158–167. ACM, New York (2000). doi:10.1145/352871.352887
39. Sarwar, B.M.: Scalability, and distribution in recommender systems. Ph.D. dissertation, University of Minnesota (2001)
40. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.T.: Application of dimensionality reduction in recommender system—a case study. In: ACM WebKDD Workshop (2000)
41. Schafer, J.B., Konstan, J.A., Riedl, J.: E-commerce recommendation applications. Data Min. Knowl. Discov. **5**(1–2), 115–153 (2001). doi:10.1023/A:1009804230409
42. Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: Methods and metrics for cold-start recommendations. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval SIGIR'02, pp. 253–260. ACM, New York (2002). doi:10.1145/564376.564421
43. Shardanand, U., Maes, P.: Social information filtering: algorithms for automating "word of mouth". In: Proceedings of SIGCHI Conference on Human Factors in Computing Systems, pp. 210–217. ACM Press/Addison-Wesley Publishing Co., New York (1995). doi:10.1145/223904.223931
44. Sheth, B., Maes, P.: Evolving agents for personalized information filtering. In: Proceedings of 19th Conference on Artificial Intelligence for Applications, pp. 345–352 (1993)
45. Smyth, B., Cotter, P.: A personalized TV listings service for the digital TV age. Knowl.-Based Syst. **13**, 53–59 (2000)
46. Soboroff, I., Nicholas, C., Nicholas, C.K.: Combining content and collaboration in text filtering. In: Proceedings of the IJCAI99 Workshop on Machine Learning for Information Filtering, pp. 86–91 (1999)
47. Sotiropoulos, D.N., Lampropoulos, A.S., Tsihrintzis, G.A.: MUSIPER: a system for modeling music similarity perception based on objective feature subset selection. User Model. User-Adapt. Interact. **18**(4), 315–348 (2008)
48. Symeonidis, P., Nanopoulos, A., Manolopoulos, Y.: Providing justifications in recommender systems. IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **38**(6), 1262–1272 (2008)
49. Terveen, L., Hill, W.: Human-computer collaboration in recommender systems. In: Carroll, J. (ed.) HCI in the New Millennium. Addison Wesley, New York (2001)

50. Tran, T., Cohen, R.: Hybrid recommender systems for electronic commerce. In: Proceedings of Knowledge-Based Electronic Markets. Papers from the AAAI Workshop, AAAI Technical Report WS-00-04, pp. 78–83. Menlo Park (2000)
51. Ungar, L., Foster, D., Andre, E., Wars, S., Wars, F.S., Wars, D.S., Whispers, J.H.: Clustering methods for collaborative filtering. In: Proceedings of AAAI Workshop on Recommendation Systems. AAAI Press (1998)
52. Vignoli, F., Pauws, S.: A music retrieval system based on user driven similarity and its evaluation. In: Proceedings of 6th International Conference on Music Information Retrieval, pp. 272–279. London (2005)
53. Zhang, Y., Koren, J.: Efficient Bayesian hierarchical user modeling for recommendation system. In: Proceedings of 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 47–54. ACM, New York (2007). doi:10.1145/1277741.1277752

Chapter 3
The Learning Problem

Abstract In general, Learning can be defined as the modification of a behavior tendency according to experiences which have been acquired. Thus, Learning embeds distinctive attributes of intelligent behavior. Machine Learning is the study of how to develop algorithms, computer applications, and systems that have the ability to learn and, thus, improve through experience their performance at some tasks. This chapter presents the formalization of the Machine Learning Problem.

3.1 Introduction

Learning is an essential human function which allows change in order to become better, according to a given criterion, when a similar situation occurs. Learning does not adopt the situation of rote learning. *Rote learning* describes the *raw* direct implantation of knowledge into a learner without the need of inference or other transformation of knowledge to be required by him/her. Rote learning focuses only on memorization of given facts without the need to draw inferences from the incoming information. However, the learning process deals with the difficulty to extract inferences and to generalize a behavior when a novel situation arises. Therefore, learning could be considered as a type of intelligence which covers a large spectrum of processes and is difficult to define precisely.

Most definitions of learning refer to it as the process to gain knowledge or skills, by study, instruction or experience. The learning process, as it is referred to in [5], could be decomposed into a number of tasks such as "the acquisition of new declarative knowledge, the development of motor and cognitive skills through instruction or practice, the organization of new knowledge into general effective representations, and the discovery of new facts through observations and experimentation".

Taking into consideration the above aspects of the learning process, we could define as **Machine Learning** the computer science discipline which is related with the computational modeling of learning into computers. Therefore, if a system is able to learn and adapt to such changes, this helps the system designers to provide

A.S. Lampropoulos and G.A. Tsihrintzis, *Machine Learning Paradigms*,
Intelligent Systems Reference Library 92, DOI 10.1007/978-3-319-19135-5_3

only the sufficient and necessary situations without the need to foresee all possible situations. Consequently, we may not be able to implement completely and exactly the desired process required of a system, but we could construct a good and useful *approximation* to it.

3.2 Types of Learning

The machine learning model includes two main entities, the entity of teacher and the entity of learner [5, 10]. The teacher plays the role of the entity that contains the required knowledge in order to perform a given task, while the learner has to learn the knowledge to perform the task. The amount of inference performed by the learner on the information provided by the teacher defines the learning strategies. Usually there is a trade-off in the amount of effort required by the learner and the teacher. As the learner is able to perform a larger amount of inference, the effort of the teacher is decreasing and vice versa. Hence, we can define three types of learning:

Learning from Instruction This type of learning consists of the learner acquiring knowledge from the teacher and transforming it into an internal representation for performing inferences. In this kind of learning, the role of the teacher is crucial as he/she is responsible for organizing the knowledge in a way that incrementally increases the learner's actual knowledge.

Learning by Analogy This type of learning consists of acquiring new facts or skills by transforming and increasing existing knowledge that bears strong similarity to the desired new concept or skill into a form effective and useful in the new situation. This type of learning requires more inference on the part of the learner than does learning by heart (rote learning) or learning from instruction. A fact or skill analogous in relevant parameters should be retrieved from memory and then the retrieved knowledge should be applied to new situations.

Learning from Examples This the most important type of learning in terms of relevance to computer science and constitutes the core of what is meant by the term *machine learning*. Specifically, this is the learning approach which could be considered as a problem of finding desired dependencies using a "limited" number of observations. This type of learning is driven from statistics, where the problem focuses on how to use samples drawn from an unknown probability distribution in order to decide from which distribution a new sample has been drawn. A related problem is how to estimate the value of an unknown function at a new point given the values of this function at a set of sample points. In other words, given a set of a examples of a concept, the learner induces a general concept description that describe the examples. In this type of learning, the amount of inference performed by the learner is not limited as in learning from instruction or in learning by analogy, but is, comparatively, much greater.

Generally, there are two subcategories of this type of learning namely *inductive* and *deductive* learning. Inductive learning aims at obtaining or discovering

general relations-dependencies-rules from particular given examples which are called training data. On the other hand, deductive learning attempts to use a set of known relations-dependencies-rules that fit the observed training data. Most machine learning approaches belong to the inductive learning category and it is these methods that we will focus on in this chapter.
More specifically, learning from examples could be discriminated in three categories according to the way that the training data are utilized to find either a dependence or an inference or a description for the general set of data in a specific problem. These categories are:

Supervised Learning includes prediction and classification tasks and constitutes the main category of learning that we will focus on in our analysis. In supervised learning, the objects that are related to a specific concept are pairs of input-output patterns. This means that data that belong to the same concept are already associated with target values, as, for example, classes which define the identities of concepts [2, 9, 15, 33, 36].

Unsupervised Learning is about understanding or finding a concise description of data by passively mapping or clustering data according to some order principles. This means that the data constitute only a set of objects where a label is not available to define the specific associated concept as it is in supervised learning. Thus, the goal of unsupervised learning is to create groups-clusters of similar objects according to a similarity criterion and then to infer a concept that is shared among of these objects [15, 33].
Also, unsupervised learning includes algorithms that aim at providing a representation from high-dimensional to low-dimensional spaces, while preserving the initial information of data and offering a more efficient computation. These techniques, called *Dimensionality Reduction Methods*, focus mainly on confronting R. Belman's phenomenon known as "the curse of dimensionality." This phenomenon is essentially the observation that increasing the number of variables in multivariate problems requires exponentially increasing the amount of computational resources.

Reinforcement Learning involves performing actions to achieve a goal [30]. In reinforcement learning, an agent learns by trial and error to perform an action to receive a reward, thereby yielding an efficient method to develop goal-directed action strategies. Reinforcement learning was inspired by related phycological theories and is strongly related to the basal ganglia in the brain. Reinforcement learning methodologies are related to problems where the learning agent does not *a priori* know what it must do. Thus, the agent must discover an action policy for maximizing the "expected gain" defined by the rewards that the agents get in a given state. Reinforcement learning differs from supervised learning because in reinforcement learning neither input/output pairs are presented, nor sub-optimal actions explicitly corrected, but the agents at a specific time t fall into a state and upon this information select an action. As a consequence, the agent receives for its action a reinforcement signal or reward. Hence, as a given state has no optimal action, the challenge of a reinforcement learning algorithm is to find a

balance between exploration of possible actions and exploitation of its current knowledge in order to maximize its reward. Additionally, the reinforcement learning algorithm has to face the challenge to discover new actions not tried in the past and thus to explore the state space. Taking these facts into account, it is clear that there is not a predefined recipe to give answers to the above dilemma. The environment of this type of learning is typically formulated as a finite-state Markov Decision Process and reinforcement learning algorithms are highly related to dynamic programming techniques. Most of these algorithms are based on estimating value functions, i.e. functions of pairs of state-action that estimate how good it is for the agent to be in a given state (or how good it is to perform a given action in a given state). The notion of "how good" is defined in terms of future rewards that can be expected or, more precisely, in terms of expected returns. Of course, the rewards the agent may expect to receive in the future depend on what actions it will take.

In the next section, we will analyze further only the statistical supervised learning paradigms that are utilized mostly in RS and related algorithms in order to extract inferences from user preferences on a set of items.

3.3 Statistical Learning

The learning process faces problems that are partially unknown, too complex or too noisy. As we mentioned in the previous section, statistical learning techniques deal with the difficult learning process of finding desired dependence-relation for an infinite domain using a finite amount of given data [9, 35, 36]. In fact, there is an infinite number of such relations. Thus, the problem is to select the relation (i.e., to infer the dependence) which is the most appropriate. The answer to this problem is provided by the principle of *Occam's razor* (also known as *principle of parsimony*): "Entities should not be multiplied beyond necessity." This principle means that one should not increase the number of entities unnecessarily or make further assumptions than are needed to explain anything. In general, one should pursue the simplest hypothesis available. Thereby, when many solutions are available for a given problem we should select the simplest one because according to the interpretation of Occam's razor, "the simplest explanation is the best" [9, 35, 36].

Consequently, we have to define what is considered as the simplest solution. In order to define what a simple solution is, we need to use prior knowledge of the problem to be solved. Often, this prior knowledge is the *principle of smoothness*. Specifically, the smoothness principle states that physical changes do not occur instantaneously, but all changes take place in a continuous way. If two points x_1, x_2 are close, then so should be the corresponding outputs y_1, y_2. For example, if we have two points, one of which is red and the other one is green, then if we will get a new point which is very close to the red one, the principle of smoothness says that this point will most probably be red. This means that the function (which describes

the relation among location of points and color) does not change abruptly, but its change takes place in a continuous way. *Therefore, the learning problem could be described as the search to find a function which solves our task.*

Taking into consideration the previous facts, two main paradigms are developed in statistical inference: the particular parametric paradigm (*parametric inference*) and the general paradigm (*nonparametric inference*).

3.3.1 Classical Parametric Paradigm

The classical parametric paradigm aims at creating simple statistical methods of inference [36]. This approach of statistical inference was influenced by Fisher's work in the area of discriminant analysis. The problem was to infer a functional dependence by a given collection of empirical data. In other words, the investigator of a specific problem knows the physical law that generates the stochastic properties of the data in the form of a function to be determined up to a finite number of parameters. Thus, the problem is reduced to the estimation of the parameters using the data. Fisher's goal was to estimate the model that generates the observed signal. More specifically, Fisher proposed that any signal X could be modeled as the sum of two components, namely a *deterministic* component and *random* component:

$$Y = f(x, a) + \varepsilon. \tag{3.1}$$

In this model, $f(x, a)$ is the deterministic part defined by values of a function which is determined up to a limited number of parameters. On the other hand, ϵ corresponds to the random part describing noise added to the signal, which is defined by a known density function. Thus, the goal was the estimation of the unknown parameters in the function $f(x, a)$. For this purpose, Fisher adopted the *Maximum Likelihood Method*. The classical parametric paradigm could be called as *Model Identification* as the main philosophical idea is the traditional goal of science to discover or to identify an existing law of nature that generates the statistical properties of data.

The classical parametric paradigm is based on three beliefs. The first belief states that the number of free parameters, that define a set of functions linear to them and contain a good approximation to the desired function, is small. This belief is based on Weiestrass's theorem, according to which "any continuous function can be approximated by polynomials at any degree of accuracy" [36].

The second belief refers to the Central Limit Theorem which states that "the sum of a large number of variables, under wide conditions, is approximated by normal (Gaussian) law." Therefore, the underlying belief is that for most real life problems, if randomness is the result of interaction among large numbers of random components, then the statistical law of the stochastic element is the normal law [36].

The third belief states that the maximum likelihood method as an induction engine is a good tool for estimating parameters of models even for small sample sizes. This

belief was supported by many theorems that are related with conditional optimality of the method [36].

However, the parametric paradigm demonstrates shortcomings in all of the beliefs that it was based on. In real life multidimensional problems, it seems naive to consider that we could define a small set of functions that contained the desired functions, because it is known that, if the desired function is not very smooth, the desired level of accuracy causes an exponential increase of the number of free terms with an increasing amount of variables. Additionally, real life problems cannot be described by only classical distribution functions, while it has also been proven that the maximum likelihood method is not the best one even for simple problems of density estimation [36].

3.3.2 General Nonparametric—Predictive Paradigm

The shortcomings of the parametric paradigm for solving high-dimensional problems have led to the creation of new paradigms as an attempt to analyze the problem of generalization of statistical inference for the problem of pattern recognition. This study started by Glivenko, Cantelli and Kolmogorov [36], who proved that the *empirical* distribution function always converges to the *actual* distribution function. The main philosophical idea of this paradigm arises from the fact that there is no reliable a priori information about the desired function that we would like to approximate. Due to this fact, we need a method to infer an approximation of the desired function utilizing the given data under this situation. This means that this method should be the best for the given data and a description of conditions should be provided in order to achieve the best approximation to the desired function that we are looking for.

As a result of the previous reasons, the need was created to state the general principle of inductive inference which is known as the principle of *Empirical Risk Minimization* (ERM). The principle of ERM suggests a decision rule (an *indicator function*) that minimizes the so-called "empirical risk", i.e. *the number of training errors* [9, 36].

The construction of a general type of learning machine started in 1962, where Rosenblatt [2, 36] proposed the first model of a learning machine known as *perceptron*. The underlying idea of the perceptron was already known in the literature of neurophysiology. However, Rosenblatt was the first to formulate the physiological concepts of learning with reward and punishment stimulus of perceptron as a program for computers and showed with his experiments that this model can be generalized. He proposed a simple algorithm of constructing a rule for separating data into categories using given examples. The main idea was to choose appropriate coefficients of each neuron during the learning process. The next step was done in 1986 with the construction of the *back-propagation* algorithm for simultaneously finding the weights for multiple neurons of a neural network.

At the end of the 1960s, Vapnik and Chervonenkis started a new paradigm known as *Model Predictions* or *Predictive Paradigm* [36]. The focus here is not on an

accurate estimation of the parameters or on the adequacy of a model on past observations, but on the predictive ability, for example the capacity of making good predictions for new observations. The classical paradigm (Model for Identification) looks for a parsimonious function $f(x, a)$ belonging to a predefined set. On the other hand, in the predictive paradigm the aim is not to approximate the true function (the optimal solution) $f(x, a_0)$, but to get a function $f(x, a)$ which gives as accurate predictions as possible. Hence, the goal is not to discover the hidden mechanism, but to perform well. This is known as "black box model" [34], which illustrates the above conception while keeping the same formulation of the classical paradigm $y = f(x, a) + \varepsilon$.

For example in the *pattern recognition problem* (also known as the *classification problem*), understanding the phenomenon (the statistical law that causes the stochastic properties of data) would be a very complex task. In the predictive paradigm, models such as artificial neural networks, support vector machines etc. attempt to achieve a good accuracy in predictions of events without the need of the identification-deep understanding of the events which are observed. The fundamental idea behind models that belong to the predictive paradigm is that the problem of estimating a model of events is "*hard*" or "*ill-posed*" as it requires a large number of observations to be solved "well" [36]. According to Hadamard, a problem is "*well-posed*" if its solution

1. exists,
2. is unique, and
3. is stable.

If the solution of the problem violates at least one of the above requirements, then the problem is considered *ill-posed*.

Consequently, finding a function for the estimation problem has led to the derivation of bounds on the quality of any possible solution. In other words, these bounds express the *generalization* ability of this function. This resulted in the adoption of the inductive principle of *Structural Risk Minimization* which controls these bounds. This theory was constructed by Vapnik and Chervonenkis and its quintessence is the use of what is called a capacity concept for a set of events (a set of indicator functions). Specifically, Vapnik and Chervonenkis introduce the *Vapnik Capacity (VC) Dimension* which characterizes the variability of a set of indicator functions implemented by a learning machine. Thus, the underlying idea to controlling the generalization ability of the learning machine pertains to "achieving the smallest bound on the test error by controlling (minimizing) the number of the training errors; the machine (the set of functions) with the smallest VC-dimension should be used" [9, 36].

Thus, there is a trade-off between the following two requirements:

- to minimize the number of training errors and
- to use a set of functions with small VC-dimension.

On the one hand, the minimization of the number of training errors needs a selection of function from a wide set of functions. On the other hand, the selection of this function has to be from a narrow set of functions with small VC-dimension.

As a result, the selection of the indicator function that we will utilize to minimize the number of errors, has to compromise the accuracy of approximation of training data and the VC-dimension of the set of functions. The control of these two contradictory requirements is provided by the *Structural Risk Minimization principle* [9, 36].

3.3.3 Transductive Inference Paradigm

The next step beyond the model prediction paradigm was introduced by Vladimir Vapnik with the publication of the *Transductive Inference Paradigm* [34]. The key ideas behind the transductive inference paradigm arose from the need to create efficient methods of inference from small sample sizes. Specifically, in transductive inference an effort is made to estimate the values of an unknown predictive function at a given restricted subset of its domain in which we are interested and not in the entire domain of its definition. This led Vapnik to formulate the *Main Principle* [9, 34, 36]:

> If you possess a restricted amount of information for solving some problem, try to solve the problem directly and never solve a more general problem as an intermediate step. It is possible that the available information is sufficient for a direct solution, but may be insufficient to solve a more general intermediate problem.

The main principle constitutes the essential difference between newer approaches and the classical paradigm of statistical inference based on use of the maximum likelihood method to estimate a number of free parameters. While the classical paradigm is useful in simple problems that can be analyzed with few variables, real world problems are much more complex and require large numbers of variables. Thus, the goal when dealing with real life problems that are by nature of high dimensionality is to define a less demanding (i.e. less complex) problem which admits well-posed solutions. This fact involves finding values of the unknown function reasonably well only at given points of interest, while outside of the given points of interest that function may not be well-estimated.

The setting of the learning problem in the predictive paradigm, which is analyzed in the next section, uses a two step procedure. The first step is the *induction stage*, in which we estimate the function from a given set of functions using an induction principle. The second step is the *deduction stage*, in which we evaluate the values of the unknown function only at given points of interest. In other words, the solution given by the inductive principle derives results first from particular to general (inductive step) and then from general to particular (deductive step).

On the contrary, the paradigm of transductive inference forms a solution that derives results directly from particular (training samples) to particular (testing samples).

In many problems, we do not care about finding a specific function with good generalization ability, but rather are interested in classifying a given set of examples

(i.e. a test set of data) with minimum possible error. For this reason, the inductive formulation of the learning problem is unnecessarily complex.

Transductive inference embeds the unlabeled (test) data in the decision making process that will be responsible for their final classification. Transductive inference "works because the test set can give you a non-trivial factorization of the (discrimination) function class" [7]. Additionally, the unlabeled examples provide information on the prior information of the labeled examples and "guide the linear boundary away from the dense region of labeled examples" [39].

For a given set of labeled data points $Train - Set = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, with $y_i \in \{-1, 1\}$ and a set of test data points $Test-Set = \{x_{n+1}, x_{n+2}, \ldots, x_{n+k}\}$, where $x_i \in \mathbb{R}^d$, transduction seeks among the feasible corresponding labels the one $y^*_{n+1}, y^*_{n+2}, \ldots, y^*_{n+k}$ that has the minimum number of errors.

Also, transduction would be useful among other ways of inference in which there are either a small amount of labeled data points available or the cost for annotating data points is prohibitive. Hence, the use of the ERM principle helps in selections of the "best function from the set of indicator functions defined in $\mathbb{R}^d$, while transductive inference targets only the functions defined on the working set $Working - Set = Train - Set \bigcup Test - Set$," which is a discrete space.

To conclude, the goal of inductive learning (classical parametric paradigm and predictive paradigm) is to generalize for any future test set, while the goal of transductive inference is to make predictions for a specific working set. In inductive inference, the error probability is not meaningful when the prediction rule is updated very abruptly and the data point may be not independently and identically distributed, as, for example, in data streaming. On the contrary, Vapnik [36] illustrated that the results from transductive inference are accurate even when the data points of interest and the training data are not independently and identically distributed. Therefore, the predictive power of transductive inference can be estimated at any time instance in a data stream for both future and previously observed data points that are not independently and identically distributed. In particular, *empirical findings suggest that transductive inference is more suitable than inductive inference for problems with small training sets and large test sets* [39].

3.4 Formulation of the Learning Problem

In here, we will follow the formulation of the problem of learning from data for the predictive paradigm [9, 36, 37], in accordance with concepts developed in the Vapnik and Chervonenkis learning theory. The general learning problem of estimation of an unknown dependence (function) between input and output of a system using a limited number of observations constitutes of three components, as is illustrated in Fig. 3.1:

1. The *Generator* (sampling distribution) produces random vectors $x \in \mathbb{R}^d$, drawn independently from a fixed probability density $P(x)$ which is unknown.

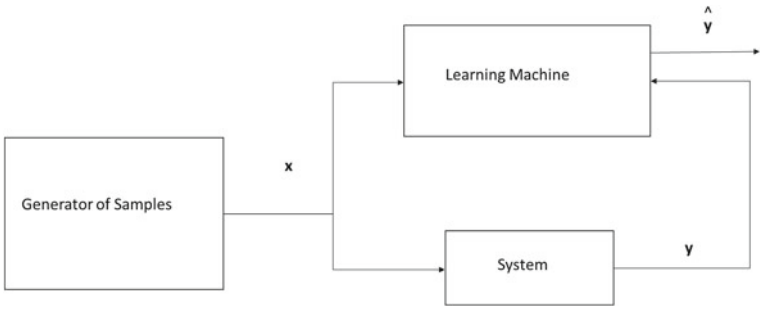


Fig. 3.1 Learning from examples

2. The *System* (also known as the supervisor) produces an output value y for every input vector x according to the fixed conditional density $P(x|y)$ which is also unknown.
3. The *Learning Machine* is capable of implementing a set of approximating functions $f(x, a)$, which depend on a set of parameters $a \in \Lambda$, where Λ is a set of allowed parameter values.

Hence, the learning problem can be defined as follows: Given a set of n training samples $(x_i, y_i), i = 1, \ldots, n$, which constitute n independent identically distributed random observations produced according to an unknown joint probability density function (pdf): $P(x, y) = P(x)P(y|x)$, select a function from the given set of approximating functions $f(x, a)$ which best approximates the System response.

We have to mention that the set of approximating functions of the Learning Machine should be chosen a priori and before the learning process is started. In the best case, the selection of a set of approximating functions reflects prior knowledge about the System. This selection is dependent on the particular application at hand and lies outside the scope of the learning problem. Common sets of approximating functions may be radial based functions, multi-layer perceptrons, wavelets etc.

In order to measure the quality of an approximation function and to *select* the best available approximation, one may measure the loss or discrepancy $L(y, f(x, a))$ between the System and the Learning Machine outputs for a given point x. In other words, a *Loss Function* is defined as a non-negative function that measures the quality of the approximating function f.

The expected value of the loss function is called the *expected or prediction risk functional*:

$$R(a) = \int L(y, f(x, a))P(x, y)\,dx\,dy. \tag{3.2}$$

Thereby, learning is the process of estimating the function $f(x, a)$ which minimizes the risk functional $R(a)$ over the set of functions $f(x, a)$ supported by the Learning

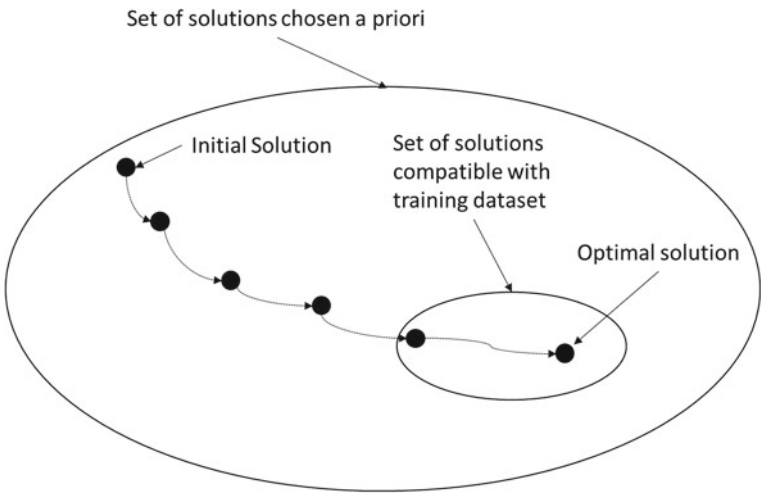


Fig. 3.2 Learning as a searching of the desired function

Machine using only the training data, as the joint probability distribution $P(x, y)$ is not known.

We cannot necessarily expect to find the optimal solution $f(x, a_0)$, so we denote with $f(x, a)$ as the estimate of the optimal solution obtained with finite training data using some learning procedure. This is illustrated in Fig. 3.2.

The learning process is divided into the following main problems:

- The problem of Pattern Recognition—Classification
- The problem of Regression Estimation
- The Problem of Density Estimation.

For each of these problems, we have different corresponding Loss functions as the output y differs. However the goal of minimizing the risk functional based only on training data, is common for all learning problems. In the next section, we will give the formulation and the loss functions for the problem of *Classification* [9, 36, 37], which is of interest in here.

3.5 The Problem of Classification

The output y of the System in the classification problem takes on only two symbolic values $y \in \{0, 1\}$, which correspond to the input x belonging to one or the other of two classes. Within the framework of RS, the two classes could correspond, for example, to items with low rating value (e.g. rating value in $\{1, 2\}$) and items with high rating value (e.g. rating value in $\{3, 4, 5\}$), respectively. Consequently, the output of the Learning Machine needs only to take on the two values 0 and 1. Thereby, the set

of functions $f(x, a)$, $a \in \Lambda$, which take these two values become a set of indicator functions. The following loss function for this problem measures the classification error:

$$L(y, f(x, a)) = |y - f(x, a)| \tag{3.3}$$

or

$$L(y, f(x, a)) = \begin{cases} 0 & \text{if } y = f(x, a) \\ 1 & \text{if } y \neq f(x, a) \end{cases} \tag{3.4}$$

Using the loss function, the risk functional 3.2 provides the probability of classification errors to occur. As a result, learning becomes the problem of finding the indicator function $f(x, a_0)$ which minimizes the probability of misclassification using only the training data.

It is useful to mention that, if the density $P(x, y)$ is known, the given learning task (classification in this case) can be solved by minimizing Eq. 3.2 directly, without a need for training data.

This implies that the density estimation problem is the most general of all learning problems and, therefore, the most difficult to solve with a finite amount of data. This remark has led Vapnik to define the Main Principle [36], to which we referred in a previous section. According to the main principle, we do not need to solve a given learning problem by indirectly solving a harder problem such as that of density estimation as an intermediate step. This principle is rarely followed under other approaches, such as artificial neural networks [2, 18] where the classification problem is solved via density estimation.

3.5.1 Empirical Risk Minimization

With the above analysis taken into account, we can estimate the indicator function $f(x, a_0)$ which minimizes the *empirical risk* (also known as the *training error*). In Eq. 3.2, the expected risk functional $R(a)$ is replaced by the *empirical risk functional*:

$$R_{emp}(a) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i, a)). \tag{3.5}$$

This inductive principle is called as *Empirical Risk Minimization* [36] and is utilized to approximate the function $f(x, a_0)$ which minimizes the risk in Eq. 3.2 by the function $f(x, a_n)$ which minimizes the empirical risk in Eq. 3.5.

The ERM principle chooses the function $f(x, a)$ that minimizes the empirical risk or the average loss for the training data. The key problem according to Vapnik [9, 36, 37] for a learning process based on the ERM principle is to ensure consistency of the learning process that minimizes the empirical risk. This will guarantee that

the minimization of the empirical risk will converge to the true risk which cannot be computed. Vapnik and Chervonenkis [35, 36] proved that the necessary and sufficient conditions for consistency of the ERM principle is that the ratio of the VC-entropy of the set of indicator functions on a sample of size n approaches to zero as the number of observed data n approaches infinity. VC entropy of the set of indicator functions provides a measure of the expected diversity of the set of indicator functions with respect to a sample of a given size, generated from some (unknown) distribution. This definition of entropy is given in Vapnik [35, 36] in the context of statistical learning theory and should not be confused with Shannon's entropy commonly used in information theory. Thus, the ERM principle is intended for dealing with large sample sizes.

Another important result of the VC theory on the generalization ability of the learning machine from a set of totally bounded non-negative functions $f(x, a)$ is the following Inequality 3.6. This inequality is useful in model selection, as it provides an upper limit for complexity for a given sample size and confidence level, with no assumptions about the type of approximating function and noise level in the data.

Specifically, the inequality for the classification problem of two classes is:

$$R(a) \leq R_{emp}(a) + \frac{\varepsilon(n)}{2}\left(1 + \sqrt{1 + \frac{4R_{emp}(a)}{\varepsilon(n)}}\right) \tag{3.6}$$

with probability $1-\eta$ simultaneously for all functions $f(x, a)$, including the function $f(x, a_0)$ and

$$\varepsilon(n) = 4\frac{h\left(\ln\frac{2n}{h} + 1\right) - \ln(\eta)}{n}, \tag{3.7}$$

such that h is a VC-dimension and n data points of training set.

As stated previously, the VC dimension h of a set of indicator functions is the maximum number of samples for which all possible binary classifications can be induced (without error). For example, h will be the VC dimension of a set of indicator functions, if there exist h samples that can be shattered by this set of functions but there are no $h-1$ samples that can be shattered by this set of functions. Here, shattering is the process of finding all possible 2^h ways with correctly assigned labels. The ERM principle works well when a large number of observed data is available, since the law of large numbers states that

$$\lim_{n\to\infty} R_{emp}(a) = R(a). \tag{3.8}$$

From the above Inequality 3.6, we observe that, when $\frac{n}{h}$ is large, the second summand on the right side of 3.6, which corresponds to the confidence interval for the empirical risk, approaches zero. Then, the true expected risk $R(a)$ is close to the value of the empirical risk $R_{emp}(a)$.

However, if $\frac{n}{h}$ is small, as with a limited number of observed data, there is no guarantee for a solution based on expected risk minimization because a small empirical risk $R_{emp}(a)$ requires a small true expected risk $R(a)$.

We have to mention that the empirical risk $R_{emp}(a)$ in Inequality 3.6, depends on a specific function from the set of functions, whereas the second term depends on the VC dimension of the set of functions. Hence, minimization of the upper bound of the true expected risk $R(a)$ in Inequality 3.6 needs to minimize both terms of the right hand side of the inequality. This means that it is necessary to make the VC dimension a controlling variable. More precisely, we need to find the set of functions with optimal VC dimension for a given training data. Thus, Vapnik proposed a new principle known as *Structural Risk Minimization*.

3.5.2 Structural Risk Minimization

Structural Risk Minimization (SRM) attempts to minimize the true expected risk $R(a)$, while paying attention to minimizing the Empirical Risk $R_{emp}(a)$ along with the VC dimension of the set of functions. Thus, the VC dimension defines the complexity of the set of functions and will not be confused with the number of free parameters or degree of freedoms [9, 36, 37]. For a finite training sample of size n, there exists an optimal element of a structure providing minimum of prediction risk. The SRM principle seeks to minimize the expected risk while avoiding underfitting and overfitting using function complexity control [9, 36]. Underfitting occurs when the selected function is not adequate to approximate the training data. On the other hand, there is overfitting when the selected function is so complex that it approximates excessively the training data and results into being insufficient to generalize well. Thus, the SRM principle provides a complexity ordering of the approximating functions. More specifically, "the SRM principle suggests a trade-off between the quality of approximation and the complexity of the approximating function" [36].

The empirical risk $R_{emp}(a)$ decreases as the VC dimension increases. As the VC dimension h is low compared to the number of training examples n, the confidence interval is narrow. This is illustrated in Fig. 3.3 (adapted from [36]).

The inductive principle SRM provides a formal mechanism for choosing an optimal complexity of a set of functions for a finite sample. SRM has been originally proposed and applied for classification [36]. However, it is applicable to any learning problem, in which the risk functional has to be minimized.

Let the set S of functions $f(x, a)$, $a \in \Lambda$ have a structure that consists of nested subsets $S_i, i = 1, 2, \ldots$ of totally bounded non-negative functions

$$S_1 \subset S_2 \subset S_3 \subset \cdots \subset S_n \subset \ldots, \tag{3.9}$$

such that the VC dimension $h_i, i = 1, 2, \ldots$ of the subsets satisfies

$$h_1 \leq h_2 \leq h_3 \leq \cdots \leq h_n \leq \ldots. \tag{3.10}$$

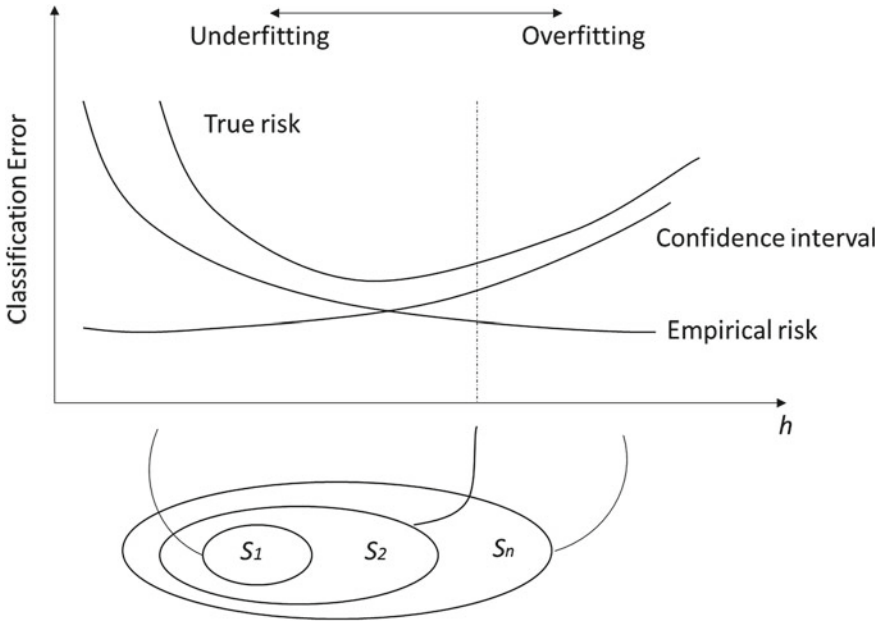


Fig. 3.3 Upper bound on expected true risk $R(a)$ and empirical risk $R_{emp}(a)$ as a function of VC dimension h

Then, the SRM principle chooses an optimal subset of the structure as a solution to the learning problem. This particular optimal subset has a specific VC dimension that yields the minimal guaranteed (i.e. lowest upper) bound on the true (expected) risk.

There are two strategies to implement the SRM inductive principle in learning algorithms and these are related to the two terms that participate in the upper bound of 3.6:

- The first strategy is to minimize the empirical risk $R_{emp}(a)$ for a fixed VC dimension h and
- the second strategy is to minimize the VC dimension, which involves minimization of the confidence interval term in 3.6 for the problem of classification, while the empirical risk $R_{emp}(a)$ remains constant (small).

3.6 Support Vector Machines

The *Support Vector Machine* (SVM) is a supervised classification system that finds an optimal hyperplane which separates data points that will generalize best to future data [4, 9, 11, 12, 35, 37]. Such a hyperplane is the, so-called, maximum margin hyperplane and maximizes the distance to the closest points from each class.

The SVM mainly relies on the following assumptions: In pattern recognition applications, usually we map the input vectors into a set of new variables (features), which are selected according to a priori assumptions about the learning problem. These features, rather than the original inputs, are then used by the learning machine. This type of feature selection often has the additional goal of controlling complexity for approximation schemes, where complexity is dependent on input dimensionality. In other words, the feature selection process has the goal to reduce redundancy in the data in order to reduce the problem complexity. On the contrary, SVM transform data into a high-dimensional space which may convert complex classification problems (with complex decision surfaces) into simpler problems that can use linear discriminant functions. SVM do not place any restriction on the number of basis functions (features) used to construct a high-dimensional mapping of the input data points. Consequently, the dimensionality of data points (feature vectors), that describe our data, does not influence significantly the learning process. Hence, it is not necessary to apply techniques of dimensionality reduction as in the classical paradigm in which the dimensionality of the data points is a critical factor.

Moreover, linear functions with constraints on complexity are used to approximate or discriminate the input data points in the high-dimensional space. SVM use linear estimators to perform approximation. This in contrast with artificial neural networks which depend on nonlinear approximations applied directly on the input space. Nonlinear estimators can potentially provide a more compact representation of the approximation function; however, they suffer from two serious drawbacks: lack of complexity measures and lack of optimization approaches that would provide a globally optimal solution.

Additionally, the linear approximating function that corresponds to the solution of the dual quadratic optimization problem is given in the kernel representation rather than in the typical basis function representation. The solution in the kernel representation is written as a weighted sum of the support vectors. The support vectors are a subset of the training data corresponding to the solution of the learning problem.

Also, SVM are based on using only those training patterns that are near the decision surface assuming they provide the most useful information for classification. SVM are based on statistical learning theory that we have analyzed in previous sections and implement the structural risk minimization (SRM) inductive principle in order to effectively generalize data sets of limited size. Specifically, SVM define a special structure on a set of equivalence classes. In this structure, each element is indexed by the margin size.

For these reasons, we select the use of SVM to address our problem. In the remainder of this section, the basic theory of SVM will be given. Additionally, we will present *One Class SVM* that are utilized in the new approach to the recommendation process proposed in here.

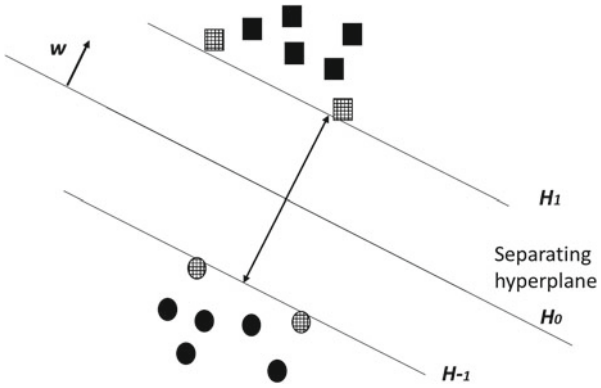


Fig. 3.4 Two class classification. Support vectors are indicated with crosses

3.6.1 Basics of Support Vector Machines

Let $X = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x \in \mathbb{R}^d$, $y \in \{-1, 1\}$ be a set of d-dimensional feature vectors corresponding to data points (e.g. recommended items).

3.6.1.1 Linearly Separable Data

In the simple case of separable training data, there exists a hyperplane decision function which separates the positive from the negative examples

$$f(x) = (w \cdot x) - b, \tag{3.11}$$

with appropriate parameters w, b [12, 37].

Let $d_{positive}$ be the shortest distance from the separating hyperplane to the closest positive example and $d_{negative}$ the corresponding shortest distance from the separating hyperplane to the closest negative example. The *margin* of the hyperplane is defined as $D = d_{positive} + d_{negative}$. Consequently, the *optimal hyperplane* is defined as the hyperplane with maximal margin. The optimal hyperplane separates the data without error. Then, if all the training data satisfy the above requirements, we have:

$$\begin{array}{ll} (w \cdot x_i) - b \geq 1 & \text{if } y_i = 1 \quad \text{hyperplane } H_1 \\ (w \cdot x_i) - b \leq -1 & \text{if } y_i = -1 \quad \text{hyperplane } H_{-1} \end{array} \tag{3.12}$$

The above inequalities can be combined in the following form:

$$y_i[(w \cdot x_i) - b] \geq 1, i = 1, \ldots, n. \tag{3.13}$$

Thereby, the optimal hyperplane can be found by solving the primal optimization problem:

$$\begin{aligned} &\text{minimize } \Phi(w) = \tfrac{1}{2}\|w\|^2 = \tfrac{1}{2}(w^T \cdot w) \\ &\text{subject to } y_i[(w \cdot x_i) - b] \geq 1, i = 1, \ldots, n. \end{aligned} \tag{3.14}$$

In order to solve this optimization problem, we form the following Lagrange functional:

$$L(w, b) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} a_i[y_i f(x_i) - 1], \tag{3.15}$$

where the a_i are Lagrange multipliers with $a_i \geq 0$. The Lagrangian $L(w, b)$ must be minimized with respect to the w and b so that the saddle point w_0, b_0 and a_i^0 satisfy the conditions:

$$\begin{aligned} \frac{\partial L(w_0, b_0)}{\partial b} &= \sum_{i=1}^{n} a_i^0 y_i = 0 \\ \frac{\partial L(w_0, b_0)}{\partial w} &= w_0 - \sum_{i=1}^{n} y_i a_i^0 x_i = 0. \end{aligned} \tag{3.16}$$

From the above equations, we obtain

$$\sum_{i=1}^{n} a_i^0 y_i = 0, a_i^0 \geq 0 \tag{3.17}$$

and

$$w_0 = \sum_{i=1}^{n} y_i a_i^0 x_i, a_i^0 \geq 0. \tag{3.18}$$

Also, the solution must satisfy the Karush-Kuhn-Tucker conditions:

$$a_i^0 y_i[(w_0 \cdot x_i) - b_0] - 1 = 0, i = 1, \ldots, n. \tag{3.19}$$

The Karush-Kuhn-Tucker conditions are satisfied at the solution of any constrained optimization problem whether it is convex or not and for any kind of constraints, provided that the intersection of the set of feasible directions with the set of descent directions coincides with the intersection of the set of feasible directions for linearized constraints with the set of descent directions. The problem for SVM is convex and for convex problems the Karush-Kuhn-Tucker conditions are necessary and sufficient for w_0, b_0 and a_i^0 to be a solution. Thus, solving the SVM problem is equivalent to finding a solution to the Karush-Kuhn-Tucker conditions. Note that there is a Lagrange multiplier a_i for every training point.

The data points for which the above conditions are satisfied and can have nonzero coefficients a_i^0 in the expansion of 3.18 are called *support vectors*. As the support vectors are the closest data points to the decision surface, they determine the location of the optimal separating hyperplane (see Fig. 3.4).

Thereby, we can substitute Eq. 3.18 back into Eq. 3.15 and, taking into account the Karush-Kuhn-Tucker conditions, we obtain the functional

$$W(a) = \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_i y_j (x_i \cdot x_j). \tag{3.20}$$

Consequently, we arrive at the, so-called, dual optimization problem:

$$\begin{aligned} &\text{maximize} \quad W(a) = \sum_{i=1}^{n} a_i - \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_i y_j (x_i \cdot x_j) \\ &\text{subject to} \quad b \sum_{i=1}^{n} a_i y_i = 0, \\ &\qquad\qquad a_i \geq 0. \end{aligned} \tag{3.21}$$

We can obtain the optimal hyperplane as a linear combination of support vectors (SV)

$$\begin{aligned} f(x) &= \sum_{i=1}^{n} y_i a_i^0 (x_i \cdot x) + b_0 \\ &= \sum_{i \in \text{SV}} y_i a_i^0 (x_i \cdot x) + b_0. \end{aligned} \tag{3.22}$$

The VC dimension h of a set of hyperplanes with margin $b_0 = \frac{1}{\|w_0\|^2}$ has the upper bound

$$h \leq min\{\| w_0 \|^2, d\} + 1. \tag{3.23}$$

The optimal separating hyperplane can be found following the SRM principle. Since the data points are linearly separable, one finds a separating hyperplane that has minimum empirical risk (i.e., zero empirical error) from the subset of functions with the smallest VC-dimension.

3.6.1.2 Linearly Nonseparable Data

Usually, data collected under real conditions are affected by outliers. Sometimes, outliers are caused by noisy measurements. In this case, outliers should be taken into consideration. This means that the data are not linearly separable and some of the training data points fall inside the margin. Thus, the SVM has to achieve two contradicting goals. On one hand, the SVM has to maximize the margin and on the other hand the SVM has to minimize the number of nonseparable data [4, 12, 37].

We can extend the previous ideas of separable data to nonseparable data by introducing a slack positive variable $\xi_i \geq 0$ for each training vector. Thereby, we can modify Eq. 3.12 in the following way

$$\begin{array}{lll} (w \cdot x_i) - b \geq 1 - \xi_i & \text{if } y_i = 1 & \text{hyperplane } H_1 \\ (w \cdot x_i) - b \leq -1 + \xi_i & \text{if } y_i = -1 & \text{hyperplane } H_{-1} \\ & \xi_i \geq 0, i = 1, \ldots, n. & \end{array} \tag{3.24}$$

The above inequalities can be combined in the following form:

$$y_i[(w \cdot x_i) - b] \geq 1 - \xi_i, i = 1, \ldots, n. \tag{3.25}$$

Obviously, if we take ξ_i large enough, the constraints in Eq. 3.25 will be met for all i. For an error to occur, the corresponding ξ_i must exceed unity, therefore the upper bound on the number of training errors is given by $\sum_{i=1} \xi_i$. To avoid the trivial solution of large ξ_i, we introduce a penalization cost C in the objective function in Eq. 3.25, which controls the degree of penalization of the slack variables ξ_i, so that, when C increases, fewer training errors are permitted. In other words, the parameter C controls the tradeoff between complexity (VC-dimension) and proportion of non-separable samples (empirical risk) and must be selected by the user. A given value for C implicitly specifies the size of margin. Hence, the SVM solution can then be found by (a) keeping the upper bound on the VC-dimension small and (b) by minimizing an upper bound on the empirical risk so the primal optimization formulation (1-Norm Soft Margin or the Box Constraint) becomes:

$$\begin{array}{ll} \text{minimize} & \Phi(\xi) = \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \xi_i \\ \text{subject to} & y_i[(w \cdot x_i) - b] \geq 1 - \xi_i, i = 1, \ldots, n. \end{array} \tag{3.26}$$

Following the same formalism of Lagrange multipliers leads to the same dual problem as in Eq. 3.21, but with the positivity constraints on a_i replaced by the constraints $0 \leq a_i \leq C$. Correspondingly, we end up with the Lagrange functional:

$$L(w, b, \xi) = \frac{1}{2} \parallel w \parallel^2 + + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} a_i[y_i((w \cdot x_i) + b) - 1 - \xi_i] - \sum_{i=1}^{n} r_i \xi_i \tag{3.27}$$

with $_i \geq 0$ and $r_i \geq 0$. The optimal solution has to fulfil the Karush-Kuhn-Tucker conditions:

$$\begin{array}{l} \frac{\partial L(w_0, b_0, \xi_0)}{\partial b} = \sum_{i=1}^{n} a_i^0 y_i = 0 \\ \frac{\partial L(w_0, b_0, \xi_0)}{\partial w} = w_0 - \sum_{i=1}^{n} y_i a_i^0 x_i = 0 \\ \frac{\partial L(w_0, b_0, \xi_0)}{\partial \xi_i} = C - a_i - r_i = 0. \end{array} \tag{3.28}$$

Substituting back the Karush-Kuhn-Tucker conditions, into the initial Lagrange functional we obtain the following objective function:

$$L(w, b, \xi) = \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_i y_j (x_i \cdot x_j). \tag{3.29}$$

This objective function is identical to that for the separable case, with the additional constraints $C - a_i - r_i = 0$ and $r_i \geq 0$ to enforce $a_i \leq C$, while $\xi_i \neq 0$ only if $r_i = 0$ and therefore $a_i = C$. Thereby the complementary Karush-Kuhn-Tucker conditions become

$$\begin{aligned} &a_i[y_i((x_i \cdot w) + b) - 1 + \xi_i] = 0, \quad i = 1, \ldots, n, \\ &\xi_i(a_i - C) = 0, \qquad\qquad\qquad\quad i = 1, \ldots, n. \end{aligned} \tag{3.30}$$

The Karush-Kuhn-Tucker conditions imply that non-zero slack variables can only occur when $a_i = C$. The points with non-zero slack variables are $\frac{1}{\|w\|} - \xi_i$, as their geometric margin is less than $\frac{1}{\|w\|}$. Points for which $0 < a_i < C$ lie at the target distance of $\frac{1}{\|w\|}$ from the hyperplane. Hence, the dual optimization problem is formulated as follows:

$$\begin{aligned} \text{maximize} \quad & W(a) = \sum_{i=1}^{n} a_i - \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_i y_j (x_i \cdot x_j) \\ \text{subject to} \quad & \sum_{i=1}^{n} a_i y_i = 0, \\ & 0 \leq a_i \leq C. \end{aligned} \tag{3.31}$$

Thus, from the primal solution (w_0, b_0, ξ_i^0), it can be shown that the optimal solution has to fulfil the following Karush-Kuhn-Tucker optimality conditions:

$$\begin{aligned} a_i = 0 \quad &\Rightarrow y_i[(w \cdot x_i) - b] \geq 1 \text{ and } \xi_i = 0 \text{ ignored vectors} \\ 0 < a_i < C \Rightarrow &\ y_i[(w \cdot x_i) - b] = 1 \text{ and } \xi_i = 0 \text{ error support vectors} \\ a_i = C \quad &\Rightarrow y_i[(w \cdot x_i) - b] \leq 1 \text{ and } \xi_i > 0 \text{ margin support vectors} \end{aligned} \tag{3.32}$$

and

$$w_0 = \sum_{i=1}^{n} a_i^0 y_i x_i. \tag{3.33}$$

The above equations indicate one of the most significant characteristics of SVM: since most patterns lie outside the margin area, their optimal a_i are zero. Only those training patterns x_i which lie on the margin surface that is equal to the a priori chosen penalty parameter C (margin support vectors), or inside the margin area (error support vectors) have non-zero a_i and are named *support vectors SV* [4, 12, 37].

The optimal solution gives rise to a decision function for the classification problem which consists of assigning any input vector x to one of the two classes according to the following rule:

$$f(x) = sign\left(\sum_{i \in \text{SV}} y_i a_i^0 (x_i \cdot x) + b_0\right). \tag{3.34}$$

According to Eq. 3.23, in an (infinite-dimensional) Hilbert space, the VC-dimension h of the set of separating hyperplanes with margin $b_0 = \frac{1}{\|w_0\|^2}$ depends only on $\| w_0 \|^2$. An effective way to construct the optimal separating hyperplane in a Hilbert space *without explicitly mapping the input data points into the Hilbert space* can be done using *Mercer's Theorem*. The inner product $(\Phi(x_i) \cdot \Phi(x_j))$ in some Hilbert space (feature space) H of input vectors x_i and x_j can be defined by a symmetric positive definite function $K(x_i, x_j)$ (called *kernel*) in the input space X

$$(\Phi(x_i) \cdot \Phi(x_j)) = K(x_i, x_j). \tag{3.35}$$

In other words, for a non-linearly separable classification problem, we map the data x from input space $X \in \mathbb{R}^d$ to some other (possibly infinite dimensional) Euclidean space H where the data are linearly separable, using a mapping function which we will call Φ:

$$\Phi : X \in \mathbb{R}^d \mapsto H. \tag{3.36}$$

Thus, we replace all inner products $(x_i \cdot x)$ by a proper $K(x_i, x_j)$ and the dual optimization problem for the Lagrangian multipliers is formulated as follows:

$$\begin{aligned} &\text{maximize} \quad W(a) = \sum_{i=1}^{n} a_i - \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_i y_j K(x_i, x_j) \\ &\text{subject to} \quad \sum_{i=1}^{n} a_i y_i = 0, \\ &\qquad\qquad 0 \le a_i \le C. \end{aligned} \tag{3.37}$$

Different kernel functions $K(x_i, x_j)$ result in different description boundaries in the original input space. In here, we utilize the gaussian or the polynomial kernel functions. These are of the respective forms:

$$K(x_i, x_j) = \exp\left\{-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right\}. \tag{3.38}$$

and

$$K(x_i, x_j) = ((x_i - x_j) + 1)^k, \tag{3.39}$$

where k is the order of the polynomial kernel function. Thus, the corresponding decision function is

$$f(\mathbf{x}) = sign\left(\sum_{i \in \text{SV}} y_i a_i^0 K(x_i, x) + b_0\right). \tag{3.40}$$

We have to note that we get linear decision functions (Eq. 3.40) in the feature space that are equivalent to nonlinear functions (Eq. 3.14). The computations do not involve the Φ function explicitly, but depend only on the inner product defined in H feature space which in turn is obtained efficiently from a suitable kernel function. The learning machines which construct functions of the form of Eq. 3.40 are called *Support Vector Machines (SVM)*.

3.6.2 Multi-class Classification Based on SVM

Let X be a labeled dataset containing n data points from K classes, $\omega_1, \omega_2, \ldots, \omega_K$. The goal is to classify new data points from the same distribution in one of the K classes. The conventional way is to decompose the K-class problem into a series of two-class problems and construct several binary classifiers. This problem of constructing and combining K binary classifiers, such as SVM, constitutes an on on-going research issue [1, 3].

The first approach to the problem follows a *One Against All* (OAA) strategy. The OAA approach constructs K binary SVM with the i_{th} one separating class i from all the remaining classes. Given a data point x to classify, all the K SVM are evaluated and the label of the class that has the largest value of the decision function is selected.

The second approach is the *One Against One* (OAO) or *Pairwise* strategy. The OAO approach is constructed by training binary SVM to classify between pairs of classes. Thus, the OAO model consists of $\frac{K(K-1)}{2}$ binary SVM for a K-class classification problem. Each of the $\frac{K(K-1)}{2}$ SVM gives one vote for its of the target class and, eventually, the given input sample x is assigned to the class with highest number of votes [22].

Neither the OAA approach nor the OAO approach significantly outperform one another in terms of classification accuracy. Their difference mainly lies in the training time, testing speed and the size of the trained classifier model. Although the OAA approach only requires K binary SVM, its training is computationally more expensive because each binary SVM is optimized on all the n training samples. Contrarily, the OAO approach has $\frac{K(K-1)}{2}$ binary SVM to train; however, each SVM is trained on $\frac{2n}{K}$ samples. The overall training speed is significantly faster than that of the OAA approach. As for the total size of the classifier model, the OAO has much fewer support vectors than the OAA.

In addition to the previous, the combination of classifiers in the voting scheme of the OAO approach can result in the possibility of ties or contradictory votings [15]. Also, in OAA voting can lead in many rejections although the accuracy on the accepted data points is very good. In order to overcome these problems a lot of alternative methods have been proposed. In [31], it is proposed to fit a simple logistic function on the outputs and combine them with the maximum combining rule in the case of the OAA approach. This does not improve over the voting method with rejections, but no data points are rejected. Also, in [14] they make use of

error-correcting output codes, which changes the definition of the class a single classifier has to learn.

Moreover, there are methods that cast the output of the SVM classifier into a confidence measure, such as posterior probability, since standard SVM do not provide such probabilities. In [24, 27], the authors developed a post-processing method for mapping the outputs of a single SVM into posterior probabilities.

The results presented in [38] on a comparison of several multi-class SVM methods indicate that all these approaches are fundamentally very similar. However, the authors conclude that the OAO approach is more practical, because the training process is faster. This is corroborated further in [1], where the authors conclude that the OAO and ECOC approaches are more accurate than the OAA approach. In LIBSVM [6], which is utilized in here for muticlass SVM classification, the OAO approach is followed.

3.7 One-Class Classification

One-class classification problems arise in many real world machine learning applications [17, 19–21, 25, 26], where a single class of patterns has to be distinguished against the remainder of the pattern space. A main characteristic of this special type of machine learning problems is that the class to be recognized, i.e. the *target class*, occupies only a negligible volume in pattern space when compared to the volume occupied by the complementary space. This setting is encountered within the broader framework of class imbalance problems where inductive learning systems attempt to recognize the positive instances of a target concept. The target class, also referred to as the *target concept*, is represented by only a few available positive patterns in contrast to the vast complementary space of negative patterns. Identifying the nature of the class imbalance problem is of crucial importance within the field of machine learning which specifically relates to (a) the degree of class imbalance which can be measured by the differences in the prior class probabilities, (b) the complexity of the concept represented by the positive patterns, (c) the overall size of the training set and (d) the classifier involved.

In [20], the authors attempted to unify the relative research by focusing on the nature of the class imbalance problem. Their findings indicate that the higher the degree of class imbalance, the higher the complexity of the concept and the smaller the overall training set, the greater the effect of class imbalances in classifier performance. This conclusion is justified by the fact that high complexity and imbalance levels as well as small training set sizes give rise to very small subclusters that classifiers fail to classify correctly. Their experiments involved several artificially generated domains of varying concept complexity, training set size and degree of imbalance. The results of those experiments revealed that the C5.0 classifier was the most sensitive compared against MLP and SVM which demonstrated the best overall performance as they were not very sensitive to the problem.

In [13], it is stated that classification problems with uneven class distributions present several difficulties during training as well as during the evaluation process of the classifiers. The context within which the authors conducted their experiments was the customer insolvency problem which is characterized by (a) very uneven distributions for the two classes of interest, namely the solvent and insolvent class of customers (b) small number of instances within the insolvent class (minority class) and (c) different misclassification costs for the two classes. In order to assess the effect of imbalances in class distributions on the accuracy of classification performance, several classifiers were employed such as Neural Networks (MLP), Multinomial Logistic Regression, Bayesian Networks (hill climbing search), Decision Tree (pruned C4.5), SVM and Linear Logistic Regression. The classification results based on the True Positive ratio which represents the ability of the classifiers in recognizing the minority (positive) class ($TP = Pr\{predicted\ Minority|actually\ Minority\}$) demonstrate poor efficiency. More specifically the best overall classifier was MLP while SVM and Logistic Linear Regression treated all minority samples as noise.

The authors in [8] mention that in cases where the number of patterns originating from the majority class greatly outnumber the number of patterns from the minority class, certain discriminative learners tend to overfit. As an alternative approach the authors propose the utilization of recognition-based learning paradigms often referred to as novelty detection approaches where the model built by the classifiers is based on samples from the target (minority) class alone. Their findings suggest that one—class (recognition based) learning, under certain conditions such as multimodality of the domain space, may, in fact, be superior to discriminative approaches such as decision trees or neural networks. Similar results are also presented in [28] where they demonstrate the optimality of one-class SVM [32] over two-class ones in certain important imbalanced-data domains. Additionally, they argue that one-class learning is related to aggressive feature selection methods which are more practical since feature selection can often be too expensive to apply.

In [23], the authors show that the novelty detection approach is a viable solution to the class imbalance problem. Specifically, the authors conducted experiments using SVM-based classifiers, on classification problems where the class imbalance was extreme, and found that novelty detectors are more accurate than balanced and unbalanced binary classifiers. Their findings also demonstrate that novelty detectors are more effective when the two classes under consideration exhibit a non-symmetrical class relationship. This situation arises in practical classification problems when each class does not consist of homogeneous patterns. Specifically, a problem is called non-symmetrical when only one-class is of interest and everything else belongs to another class. The one-class classification approach was also reported as a remedy for class imbalance problems by [16] where they claim that one-class classification can be considered as an alternative solution at the algorithmic level.

3.7.1 One-Class SVM Classification

A classifier based on one-class support vector machines (*One-Class SVM*) is a supervised classification system that finds an optimal hypersphere which encompasses within its bounds as many training data points as possible. The training patterns originate only from one-class, namely the class of positive patterns. In the context of item-based recommendation, the class of positive patterns is interpreted as the class of desirable items for a particular user. A one-class SVM classifier attempts to obtain a hypersphere of minimal radius into a feature space H using an appropriate kernel function that will generalize best on future data. This means that the majority of incoming data pertaining to the class of positive patterns will fall within the bounds of the learnt hypersphere. In this section we will describe the method for one-class SVM proposed by [29].

Assuming that we have a set X of d dimensional data points x_i $X = \{x_1, x_2, \ldots, x_n\}$, where $x \in \mathbb{R}^d$, $i = \{1, 2, \ldots, n\}$. Also, we have a function (map) $\Phi : \mathbb{R}^d \mapsto H$ that maps the data points into a higher dimensional feature (inner product) space H such that the inner product in the image of Φ can be computed by evaluating a kernel $K(x_i, x_j) = (\Phi(x_i) \cdot \Phi(x_j))$.

The purpose of one-class SVM is to construct a function (hyperplane) f that holds most of the data points in its positive side, which means that f takes the value $+1$ in the "small" region of data points and -1 elsewhere. The goal of the proposed method is to map the data points into the feature space through the kernel function and to separate them from the origin with maximum margin.

The following quadratic problem can be formulated:

$$\begin{aligned} &\text{minimize} \quad \tfrac{1}{2}\|w\|^2 + \tfrac{1}{\nu l}\sum_{i=1}^{n}\xi_i - \rho \\ &\text{subject to} \quad (w \cdot \Phi(x_i)) \geq \rho - \xi_i, i = 1, \ldots, n, \\ &\qquad\qquad \text{where } \xi_i \geq 0 \text{ and } \nu \in (0, 1]. \end{aligned} \tag{3.41}$$

In order to solve this optimization problem, we form the following Lagrange functional:

$$L(w, \rho, \xi) = \frac{1}{2} \parallel w \parallel^2 + \frac{1}{\nu l}\sum_{i=1}^{n}\xi_i - \rho - \sum_{i=1}^{n} a_i[(w \cdot \Phi(x_i)) - \rho + \xi_i] - \sum_{i=1}^{n} r_i\xi_i \tag{3.42}$$

with $a_i \geq 0$ and $r_i \geq 0$.

The optimal solution has to fulfil the Karush-Kuhn-Tucker conditions:

$$\begin{aligned} \tfrac{\partial L(w_0,\rho_0,\xi_0)}{\partial \rho} &= \sum_{i=1}^{n} a_i^0 - 1 = 0 \\ \tfrac{\partial L(w_0,\rho_0,\xi_0)}{\partial w} &= w_0 - \sum_{i=1}^{n} a_i^0 \Phi(x_i) = 0 \\ \tfrac{\partial L(w_0,\rho_0,\xi_0)}{\partial \xi_i} &= \tfrac{1}{\nu l} - a_i - r_i = 0 \end{aligned} \tag{3.43}$$

Substituting back into the initial Lagrange, functional we obtain the following objective function:

$$\begin{aligned} L(w, \rho, \xi) &= \sum_{i=1}^{n} a_i - \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j (\Phi(x_i) \cdot \Phi(x_j)) \\ &= \sum_{i=1}^{n} a_i - \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j K(x_i, x_j). \end{aligned} \tag{3.44}$$

Hence, the dual optimization problem is formulated as follows:

$$\begin{aligned} &\text{maximize} \quad W(a) = \sum_{i=1}^{n} a_i - \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j K(x_i, x_j) \\ &\text{subject to} \quad \sum_{i=1}^{n} a_i = 1, \\ &\qquad\qquad 0 \leq a_i \leq \tfrac{1}{\upsilon l}. \end{aligned} \tag{3.45}$$

The optimal solution gives rise to a decision function of the following form:

$$f(x) = \sum_{i \in \text{SV}} a_i^0 K(x_i, x_j) + \rho_0 \tag{3.46}$$

such that

$$f(x) = \left\{ \begin{array}{l} +1, \text{ if x } \in \text{ X} \\ -1, \text{ if x } \in \overline{X}, \end{array} \right\} \tag{3.47}$$

where

$$0 \leq a_i \leq \frac{1}{\upsilon n}, \sum_{j=1}^{n} a_i = 1. \tag{3.48}$$

The corresponding data points that the a_i coefficients are non-zero, so that $w_0 = \sum_{i=1}^{n} a_i^0 \Phi(x_i)$, are the margin support vectors and define the decision function.

The parameter υ controls the trade-off between two goals. On one hand, the goal is that the decision function be positive for most of the data points that belong to the training set. On the other hand, the goal is to also keep the complexity (VC dimension) small, which depends on w according to Eq. 3.23. In [29], it was shown that $\upsilon = \frac{1}{nC}$ is an upper bound for the fraction of the target class data points outside the estimated region and a lower bound on the fraction of the numbers of support vectors. If $\upsilon = 0$ involves that the penalization of errors becomes infinite $C = \infty$ and no target data points are allowed outside the estimated region. If $\upsilon = 1$ then the kernel function of Eq. 3.46 reduces to a Parzen windows estimate of the underlying density. If $\upsilon < 1$ ensures that the density of the data points will be represented only from those that constitute the subset of support vectors.

Assume, that we are given $X(u_a) \subseteq X$, which is the set of positive samples that a particular user (active user, u_a) provides to the system as an initial estimate of the kind of items (files) that he/she considers desirable. In other words, $X(u_a)$ is a subset of indices that are rated as preferable items by the active user. The One-Class SVM classifier subsequently learns a hypersphere in the feature space that encompasses as many as possible of the given positive samples. The induction principle which forms the basis for the classification of new items is stated as follows: a new sample is assigned to the class of desirable patterns if the corresponding feature vector lies within the bounds of the learnt hypersphere, otherwise it is assigned to the class of non-desirable patterns [25].

3.7.2 Recommendation as a One-Class Classification Problem

The CF approach constitutes the major technique utilized in most of RS. As stated in the previous section, such systems are hindered mainly by the *New-Item Problem* and the *Sparsity Problem*. These problems arise as a consequence of the fact that it is difficult to collect a sufficient amount of ratings for the majority of our items. Thus, we need methods that need to utilize only a small fraction of the total set of items that are positively rated and confront the problem of missing negative (non-desirable) items. An efficient approach would be to decompose the initial recommendation problem into the following form:

1. firstly, identify only the desirable items from the large amount of all possible items and
2. secondly, assign a corresponding rating degree to these.

Thus, a natural solution to the problem of identification of only appropriate items is to adopt the one class classification approach in order to identify the desirable items for a particular user.

The main problem dominating the design of an efficient multimedia RS is the difficulty faced by its users when attempting to articulate their needs. However, users are extremely good at characterizing a specific instance of multimedia information as preferable or not. This entails that it is possible to obtain a sufficient number of positive and negative examples from the user in order to employ an appropriate machine learning methodology to acquire a user preference profile. Positive and negative evidence concerning the preferences of a specific user are utilized by the machine learning methodology so as to derive a model of how that particular user valuates the information content of a multimedia file. Such a model could enable a RS to classify unseen multimedia files as desirable or non-desirable according to the acquired model of the user preferences. Thus, the problem of recommendation is formulated as a binary classification problem where the set of probable classes would include two class instances, C_+ = prefer/like and C_- = not prefer/dislike. However, the burden of obtaining a sufficient number of positive and negative examples from a user is not negligible. Additionally, users find it sensibly hard to explicitly express

what they consider as non desirable since the reward they will eventually receive does not outweigh the cost undertaken in terms of time and effort. It is also very important to mention that the class of desirable patterns occupies only a negligible volume of the patterns space since the multimedia instances that a particular user would characterize as preferable are only few compared to the vast majority of the non-desirable patterns.

This fact justifies the *highly unbalanced nature of the recommendation problem*, since non-targets occur only occasionally and their measurements are very costly. Moreover, if there were available patterns from the non-target class, they could not be trusted in principle as they would be badly sampled, with unknown priors and ill-defined distributions. In essence, non-targets are weakly defined since they appear as any kind of deviation or anomaly from the target objects. Since samples from both classes are not available, machine learning models based on defining a boundary between the two classes are not applicable. Therefore, a natural choice in order to overcome this problem is to build a model that either provides a statistical description for the class of the available patterns or a description concerning the shape/structure of the class that generated the training samples. *This insight has led us to reformulate the problem of recommendation as a one-class classification problem, where the only available patterns originate from the target class to be learnt*. Specifically, the class to be considered as the target class is the class of desirable patterns while the complementary space of the universe of discourse corresponds to the class on non-desirable patterns. Otherwise stated, our primary concern is to derive an inductive bias which will form the basis for the classification of unseen patterns as preferable or not. In the context of building an item-based RS, available training patterns correspond to those multimedia instances that a particular user assigned to the class of preferable patterns. The recommendation of new items is then performed by utilizing the one-class classifier for assigning unseen items in the database either in the class of desirable patterns or in the complementary class of non-desirable patterns.

The general setting of the recommendation problem, where there is a unique class of interest and everything else belongs to another class, manifests its extremely non-symmetrical nature. Additionally, the probability density for the class of target patterns may be scattered along the different intrinsic classes of the data. For example, the universe of discourse for our music piece RS is a music database which is intrinsically partitioned into 10 disjoint classes of musical genres. Thus, the target class of preferable patterns for a particular may be formed as a mixing of the various musical genres in arbitrary proportions. The non-symmetrical nature of the recommendation problem is an additional fact validating its formulation as a one-class learning problem.

Another important factor that leads toward the selection of one-class learning as a valid paradigm for the problem of recommendation is that the related misclassification costs are analogously unbalanced. The quantities of interest are the false positive rate and the false negative rate. The false positive rate expresses how often a classifier falsely predicts that a specific pattern belongs to the target class of patterns while it originated from the complementary class. The false negative rate expresses how

often a classifier falsely predicts that a specific pattern belongs to the complementary class of patterns while it originated from the target class. In the context of designing a music piece RS, the cost related to the false positive rate is of greater impact than the cost related to the false negative rate. False positives result in recommending items that a particular user would classify as non-desirable and, thus, effect the quality of recommendation. In contrast, false negatives result in not recommending items that a particular user would classify as desirable. Thus, it is of vital importance to minimize the false positive rate which results in improving the accuracy of recommendation.

References

1. Allwein, E.L., Schapire, R.E., Singer, Y.: Reducing multiclass to binary: a unifying approach for margin classifiers. J. Mach. Learn. Res. **1**, 113–141 (2001). doi:10.1162/15324430152733133
2. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
3. Bredensteiner, E.J., Bennett, K.P.: Multicategory classification by support vector machines. Comput. Optim. Appl. **12**(1–3), 53–79 (1999). doi:10.1023/A:1008663629662
4. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Discov. **2**(2), 121–167 (1998). doi:10.1023/A:1009715923555
5. Camastra, F., Vinciarelli, A.: Machine Learning for Audio, Image and Video Analysis: Theory and Applications. Springer, London (2007)
6. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001). Software http://www.csie.ntu.edu.tw/cjlin/libsvm
7. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning. The MIT Press, Cambridge (2006)
8. Chawla, N.V., Japkowicz, N., Kotcz, A.: Editorial: special issue on learning from imbalanced data sets. SIGKDD Explor. Newsl. **6**(1), 1–6 (2004). doi:10.1145/1007730.1007733
9. Cherkassky, V., Mulier, F.M.: Learning from Data: Concepts, Theory, and Methods. Wiley-IEEE Press (2007)
10. Cord, M., Cunningham, P.: Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval. Springer, Berlin (2008)
11. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995). doi:10.1023/A:1022627411411
12. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, Cambridge (2000)
13. Daskalaki, S., Avouris, N.: Evaluation of classifiers for an uneven class distribution problem. Appl. Artif. Intell. **20**, 381–417 (2006)
14. Dietterich, T.G., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. J. Artif. Intell. Res. **2**(1), 263–286 (1994)
15. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley, New York (2001)
16. Garcia, V., Sanchez, J.S., Mollineda, R.A., Alejo, R., Sotoca, J.: The class imbalance problem in pattern classification and learning. In: Proceedings of CEDI 2007, pp. 283–291 (2007)
17. Harmeling, S., Dornhege, G., Tax, D., Meinecke, F., Müller, K.R.: From outliers to prototypes: ordering data. Neurocomputing **69**(13–15), 1608–1618 (2006). doi:10.1016/j.neucom.2005.05.015
18. Haykin, S.: Neural Networks, 2nd edn. Prentice Hall, Upper Saddle (1999)
19. Japkowicz, N.: One-class classification. Ph.D. thesis, Delft University of Technology (2001)
20. Japkowicz, N., Stephen, S.: The class imbalance problem: a systematic study. Intell. Data Anal. **6**(5), 429–449 (2002)

21. Juszczak, P.: Learning to recognise. a study on one-class classification and active learning. Ph.D. thesis, Delft University of Technology (2006)
22. Kreßel, U.H.G.: Pairwise Classification and Support Vector Machines, pp. 255–268. MIT Press, Cambridge (1999)
23. Lee, H.J., Cho, S.: Application of LVQ to novelty detection using outlier training data. Pattern Recognit. Lett. **27**(13), 1572–1579 (2006). doi:10.1016/j.patrec.2006.02.019
24. Lin, H.T., Lin, C.J., Weng, R.C.: A note on Platt's probabilistic outputs for support vector machines. Mach. Learn. **68**(3), 267–276 (2007). doi:10.1007/s10994-007-5018-6
25. Manevitz, L.M., Yousef, M.: One-class SVMs for document classification. J. Mach. Learn. Res. **2**, 139–154 (2002)
26. Moya, M.R., Koch, M.W., Hostetler, L.D.: One-class classifier networks for target recognition applications. In: Proceedings of World Congress on Neural Networks, pp. 797–801. Portland (1993)
27. Platt, J.: Advances in Large Margin Classifiers, chap. Probabilities for SV Machines, pp. 61–274. MIT Press, Cambridge (2000)
28. Raskutti, B., Kowalczyk, A.: Extreme re-balancing for SVMs: a case study. SIGKDD Explor. Newsl. **6**(1), 60–69 (2004). doi:10.1145/1007730.1007739
29. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Comput. **13**(7), 1443–1471 (2001). doi:10.1162/089976601750264965
30. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, Cambridge (2000)
31. Tax, D., Duin, R.: Using two-class classifiers for multiclass classification. In: Proceedings of 16th International Conference on Pattern Recognition 2002, vol. 2, pp. 124–127 (2002)
32. Tax, D.M.J.: Concept-learning in the absence of counter-examples:an auto association-based approach to classification. Ph.D. thesis, The State University of New Jersey (1999)
33. Theodoridis, S., Koutroumbas, K.: Pattern Recognition, 4th edn. Academic Press, Waltham (1999)
34. Vapnik, V.N.: Estimation of Dependences Based on Empirical Data: Springer Series in Statistics. Springer, Secaucus (1982)
35. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1995)
36. Vapnik, V.N.: Statistical Learning Theory. Wiley, New York (1998)
37. Vapnik, V.N.: An overview of statistical learning theory. IEEE Trans. Neural Netw. **10**(5), 988–999 (1999)
38. Wei Hsu, C., Lin, C.J.: A comparison of methods for multi-class support vector machines. IEEE Trans. Neural Netw. **13**, 415–425 (2002)
39. Zhu, X.: Semi-supervised learning literature survey. Technical Report, University of Wisconsin, Department of Computer Science (2008). http://pages.cs.wisc.edu/~jerryzhu/research/ssl/semireview.html/

Chapter 4
Content Description of Multimedia Data

Abstract The rapid growth of multimedia technologies has led to the creation of large collections of various types of multimedia data accessed and controlled by a wide range of users, systems, applications and services. Here, the term *multimedia data* refers mainly to data contained in audio, image, video, and text files. Such data are characterized by their large quantities, high complexity and format diversity. Thus, intelligent methods, applications, systems, and services are needed that are able to process the content in massive multimedia data collections and extract meaningful and useful information. This chapter presents the process of content description of multimedia data.

4.1 Introduction

The ever-expanding range of web-based and mobile applications that use audio, image, video, and text file collections have triggered significant research efforts into the direction of development of advanced tools for effective retrieval and management of multimedia data [3, 16, 19–23, 27, 28].

The processing tasks of multimedia data are two-fold. On one hand, the aim is to enhance the description of the content of multimedia data by annotating *meta-data* to them [7, 28]. On the other hand, methodologies, techniques and processes are developed that make use of the meta-information that has been extracted from the multimedia data.

In general, two categories of meta-data are common: The first category includes *content-independent*, also referred to as *knowledge-based*, meta-data. These meta-data are not concerned directly with the content (i.e., the raw multimedia data), but rather are related to it. Thus, these meta-data are not extracted automatically in the form of a function of the multimedia data content, but rather require efforts to be made by the multimedia data collection administrator and/or its users towards their creation and maintenance. Usually, this kind of information (meta-data) is text-based or categorical and utilized to offer a *semantic* description of multimedia data. An example of this kind of meta-data for music file organization is the ID3 format, an extension to the popular mp3 format which allows the user to add specific tags

A.S. Lampropoulos and G.A. Tsihrintzis, *Machine Learning Paradigms*,
Intelligent Systems Reference Library 92, DOI 10.1007/978-3-319-19135-5_4

such as song title, album name, artist or group name, etc. Two of the main drawbacks of these meta-data are the following: (1) they reflect the subjectivity of the annotator and (2) the annotation process is prone to be inconsistent, incomplete, ambiguous, and very difficult to become automated.

The second category of meta-data includes *content-dependent*, also referred to as *content-based*, meta-data. These meta-data are extracted as a function of the multimedia data content via well-defined numerical computations. This meta-information (meta-data) is based mainly on signal processing methodologies for multimedia data [26] and aims at capturing the perceptual saliency of content. For this purpose, use is made of methodologies from audio and [4, 24] and image processing [14, 15] in order to extract information that describes the content in an objective way. This processing is usually made by an *extractor* and leads to the transformation of raw multimedia data into an essential level of information which is known as *feature extraction*. The *features* consist a number of measures or variables or parameters that allow a compact description of (aspects of) the raw data. In other words, the features allow a computational representation of (aspects of) the multimedia content. A number of features are organized in a vector form known as *feature vector*. Thereby, for each multimedia item (audio/image/video/text) of a multimedia collection, the values for each feature are extracted and used to construct the corresponding feature vector which stands for the *identity/signature* of the initial multimedia file.

Regarding the development of methodologies, techniques and processes that utilize the meta-information for organizing and manipulating the multimedia data, machine learning techniques and algorithms can offer an efficient paradigm for addressing these issues and providing an insight into the domain from which the data are drawn [1, 5]. Machine learning techniques offer the possibility of integration of both low-level, content-based features addressing the more detailed perceptual aspects and high-level semantic features underlying the more general conceptual aspects of multimedia data. Hence, classification, pattern recognition and information retrieval based on the machine learning paradigm play an important role for finding and managing the latent correlation between low-level features and high-level concepts of large collections of multimedia data. Thereby, the use of machine learning methodologies provides an efficient way to organize [2, 19] and access multimedia data collections in a manner that surpasses the limitations of conventional databases and text-based retrieval systems. Machine learning is capable of coping with the challenge to bridge the gap between the semantic information enclosed in knowledge-based meta-data, that represent the relatively more important interpretations of multimedia objects as perceived by users, with the low-level content-based features, that are directly related to aspects of multimedia content. As it is feasible to apply statistical learning methods and create similarity measures by using the statistical properties of low-level features, a learning machine can be trained to model *perceptual* aspects of users represented in high-level conceptual information by using the information contained in the low-level content-based features [18].

4.2 MPEG-7

MPEG-7 [8, 9, 11] is the ISO/IEC 15938 standard developed by the Moving Pictures Expert Group. MPEG-7 aims at standardizing the description of multimedia content data. It defines a standard set of descriptors that can be used to describe various types of multimedia information. The standard is not aimed at any particular application area, instead it is designed to support as broad a range of applications as possible. Thus, it defines a library of elementary meta-data descriptors, which can be grouped to hierarchical description schemes. Additionally, it provides a language *Description Definition Language* (DDL) that allows the standard to be available in extensions. The tools provided for capturing multimedia characteristics include not only basic features such as timbre, colors, etc., but also provide the ability to extract high level abstract features such as objects, time and interaction which require human annotation.

4.2.1 Visual Content Descriptors

MPEG-7 standard [8, 10, 11, 17] specifies a set of descriptors, each defining the syntax and the semantics of an elementary visual low-level feature e.g., color or shape. A brief overview of each descriptor is presented below, while more details can be found in [8, 17].

In order to extract visual low-level features, we utilized the MPEG-7 eXperimentation Model (MPEG-7 XM) [10, 12]. There are three subsets of low-level visual descriptors:

- **Color** is one of the most important visual features in image and video content analysis. Color features are relatively robust to changes in the background colors and are independent of image size and orientation.
- **Texture** refers to the visual patterns that have or lack properties of homogeneity, that result from the presence of multiple colors or intensities in the image. It is a property of virtually any surface, including clouds, trees, bricks, hair, and fabric. It contains important structural information of surfaces and their relationship to the surrounding environment.
- **Shape** In many image database applications, the shape of image objects provides a powerful visual clue for similarity matching. Typical examples of such applications include binary images with written characters, trademarks, pre-segmented object contours and 2-D/3-D virtual object boundaries. In image retrieval, it is usually required that the shape descriptor be invariant to scaling, rotation, and translation. Depending on the application, shape information can be 2-D or 3-D in nature. In general, 2-D shape description can be divided into two categories, namely contour-based and region-based. The former uses only boundary information of objects, suitable to describe objects that have similar contour characteristics. The latter

uses the entire shape region to extract a meaningful description which is most useful when objects have similar spatial distributions of pixels in objects.

4.2.1.1 Visual Color Descriptors

Scalable Color descriptor (SC) is a color histogram in the HSV color space, which is uniformly quantized into 256 bins according to the tables provided in the normative part. Histogram values are nonlinearly quantized using the Haar transform, down to a preferred level. The binary presentation is scalable in terms of number of bins (256 in this study) and number of bitplanes.

Color Structure descriptor (CS) captures both global color content similar to a color histogram and information about the local spatial structure of the content. The spatial organization of colors in local neighborhoods is determined with a square structuring element, the size of which is determined from the dimensions of the input image. Descriptor produces a histogram.

Color Layout descriptor (CL) specifies a spatial distribution of colors for high-speed retrieval and browsing. Descriptor are extracted from an (8×8) array of local dominant colors determined from the 64 (8×8) blocks the image is divided into. Descriptors are matched using a tailored similarity metric.

4.2.1.2 Visual Texture Descriptors

The *Edge Histogram* descriptor (EH) captures the spatial distribution of edges. Four directions of edges (0, 45, 90, 135) are detected in addition to non-directional ones. The input image is divided into 16 non-overlapping blocks and a block-based extraction scheme is applied to extract the types of edges and calculate their relative populations, resulting in a 80-dimensional vector.

The *Homogeneous Texture* descriptor (HT) filters the image with a bank of orientation and scale tuned filters that are modeled using Gabor functions. The first and second moments of the energy in the frequency domain in the corresponding sub-bands are then used as the components of the texture descriptor.

4.2.1.3 Visual Shape Descriptors

Region-Based Shape Descriptor using Angular Radial Transformation (ART) belongs to the class of moment invariants methods for shape description. This descriptor is suitable for shapes that can be best described by shape regions rather than contours. The main idea behind moment invariants is to use region-based moments which are invariant to transformations, as the shape feature. The MPEG-7 ART descriptor employs a complex Angular Radial Transformation defined on a unit disk in polar coordinates to achieve this goal. Coefficients of ART basis functions are

quantized and used for matching. The descriptor is very compact (140 bits/region) and also very robust to segmentation noise.

Contour-Based Shape Descriptor is based on curvature scale-space (CCS) representations of contours and also includes of eccentricity and circularity values of the original and filtered contours. A CCS index is used for matching and indicates the heights of the most prominent peak, and the horizontal and vertical positions on the remaining peaks in the so-called CSS image. The average size of the descriptor is 122 bits/contour. Contour-Based Shape Descriptor allows to discriminate shapes which have similar region but different contour properties.

3-D Shape Descriptor or Shape Spectrum Descriptor is useful to compare natural or virtual 3-D objects. The descriptor is based on a shape spectrum concept. Roughly speaking, the shape spectrum is defined as the histogram of a shape index, computed over the entire 3-D surface. The shape index itself measures local convexity of each local 3-D surface. Histograms with 100 bins are used, each quantized by 12 bits.

4.2.2 Audio Content Descriptors

The MPEG-7 Audio Framework [6, 11, 13] consists of seventeen Descriptors, representing spectral and temporal features. They play an important role in describing audio material and therefore provide a basis for the construction of higher-level audio applications. The low-level audio descriptors can be categorized into the following groups:

- Basic: Audio Waveform (AWF), Audio Power (AP).
- Basic Spectral: Audio Spectrum Envelop (ASE), Audio Spectrum Centroid (ASC), Audio Spectrum Spread (ASS), Audio Spectrum Flatness (ASF).
- Basic Signal Parameters: Audio Fundamental Frequency (AFF) of quasi-periodic signals and Audio Harmonicity (AH).
- Temporal Timbral descriptors: Log Attack Time (LAT), Temporal Centroid (TC).
- Timbral Spectral descriptors: Harmonic Spectral Centroid (HSC), Harmonic Spectral Deviation (HSD), Harmonic Spectral Spread (HSS), Harmonic Spectral Variation (HSV) and Spectral Centroid.
- Spectral Basis: Audio Spectrum Basis (ASB) and Audio Spectrum Projection (ASP).

4.2.2.1 Basic Descriptors

The two basic audio descriptors are temporally sampled scalar values for general use, applicable to all kinds of signals. The AWF descriptor describes the audio waveform envelope (minimum and maximum), typically for display purposes. It is used to describe the minimum and maximum sampled amplitude values reached by audio signal within the same period.

The AP describes the temporally-smoothed instantaneous power of samples in the frame, (the mean of 10 ms frame is the fundamental resolution of this descriptor). In other words it is a temporally measure of signal content as a function of time and offers a quick summary of a signal in conjunction with other basic spectral descriptors.

4.2.2.2 Basic Spectral Descriptors

The basic spectral audio descriptors all share a common basis, all deriving from the short term audio signal spectrum (analysis of frequency over time). They are all based on the ASE Descriptor, which is a logarithmic-frequency spectrum. This descriptor provides a compact description of the signal spectral content and represents the similar approximation of logarithmic response of the human ear. The ASE is a vector that describes the short-term power spectrum of an audio signal. The resolution of the ASE ranges between $\frac{1}{16}$ of an octave and 8 octaves. Consequently, the wide range of the ASE allows a suitable selection of level of spectral description information in terms of spectral resolution of the logarithmic bands. The lowest band *loEdge* is normally 62.5 Hz and the width of the spectrum is then 8 octaves to the *hiEdge* of 16 kHz. This which was chosen as a realistic limit of human hearing. The ASE is computed by power spectrum based on a Fast Fourier Transform (FFT) of the frame of audio signal samples. Usually, a frame size of 30 ms is utilized together with a Hamming window function. Zero padding of the frame is used to allow for discrimination power of two FFT sizes.

The ASC descriptor describes the center of gravity of the ASE. This Descriptor is an economical description of the shape of the power spectrum. It is an indicator as to whether the spectral content of a signal is dominated by high or low frequencies. The ASC Descriptor could be considered as an approximation of perceptual *sharpness* of the signal.

The ASS descriptor complements the ASC descriptor and equals the second moment of the ASE. Specifically, this decriptor indicates whether the signal content, as it is represented by the power spectrum, is concentrated around its centroid or spread out over a wider range of the spectrum. This gives a measure which allows the distinction of noise-like sounds from tonal sounds.

The ASF describes the flatness properties of the spectrum of an audio signal for each of a number of frequency bands. Thus, the signal is divided into nominal quarter-octave resolution, logarithmically spaced, overlapping frequency bands and the spectral flatness is computed for each band. Spectral flatness is the ratio of the geometric mean to the arithmetic mean of spectral power within a band. When this vector indicates a high deviation from a flat spectral shape for a given band, this is indicative of the presence of tonal components.

4.2.2.3 Signal Parameters

The signal parameters constitute a simple parametric description of the audio signal. This group includes the computation of an estimate for the fundamental frequency

(F0) of the audio signal. The fundamental frequency is by itself a research area in audio signal processing. For the purposes of the MPEG-7 standard there is no normative algorithm for estimation of the fundamental frequency of the audio signal as the scope of the standard is to remain open to future techniques that may become available.

However, the specific requirement of the standard from all of these algorithms is to provide a measure of periodicity for the signal analysis interval. The parameters required by the standard are the *loLimit* and *hiLimit* which correspond to the lower and upper limits of the frequency range in which F0 is contained. A measure of confidence on the presence of periodicity in the analysed part is a value belongs to [0, 1] in which 0 indicates non-periodic signal. These measures are stored in the *Weight* field of a *Series Of Scalar* that stores a temporal series of AFF descriptors. Thus, the AFF descriptor provides estimates of the fundamental frequency in segments in which the audio signal is assumed to be periodic. This measure can be used by sound matching algorithms as a weight for handling portions of a signal that are not clearly periodic.

The AH represents the harmonicity of a signal, allowing distinction between sounds with a harmonic spectrum (e.g., musical tones or voiced speech [e.g., vowels]), sounds with an inharmonic spectrum (e.g., metallic or bell-like sounds) and sounds with a non-harmonic spectrum (e.g., noise, unvoiced speech, or dense mixtures of instruments). The AH provides two measures of the harmonic properties of a signal spectrum. These are the Harmonic Ratio (HR) and the Upper Limit of Harmonicity (ULH). The HR gives the ratio of the harmonic components in the total power spectrum. The ULH is the frequency in the spectrum beyond which the spectrum cannot be considered that possess harmonic content.

4.2.2.4 Timbral Descriptors

Timbral descriptors aim at describing perceptual features of instrument sounds. Timbre refers to features that allow one to distinguish two sounds that are equal in pitch, loudness and subjective duration. These descriptors are taking into account several perceptual dimensions at the same time in a complex way.

Timbral Temporal Descriptors The Timbral Temporal descriptors describe temporal characteristics of segments of sounds, and are especially useful for the description of musical timbre (characteristic tone quality independent of pitch and loudness). The Timbral Temporal descriptors are used only within an audio segment and are intended to compute parameters of the signal envelope. The signal envelope describes the energy change of the signal over time and is generally equivalent to the known ADSR (Attack, Delay, Sustain, Release) phases and the corresponding time limits of a musical sound. Attack is the length of time required to reach its initial maximum volume. Decay is the time taken for the volume to reach a second volume level known as sustain level. The sustain level is the volume level at which sound sustains after the decay phase. Usually, the sustain level is lower than the attack volume. Release, is the time it takes for the volume to reduce

to zero. However, it is not necessary for a sound signal to include all four phases. Temporal Timbral descriptors describe the signal power function over time. The power function is estimated as a local mean square value of the signal amplitude value within a running window.

The Log Attack Time (LAT) descriptor characterizes the "attack" of a sound, the time it takes for the signal to rise from silence to its maximum amplitude. The LAT is utilized for the description of onsets of single sound samples from different musical instruments. In MPEG-7, the LAT is defined as the decimal base logarithm of the duration of the attack phase. This feature signifies the difference between a sudden and a smooth sound.

The Temporal Centroid (TC) descriptor computes a time-based centroid as the time average over the energy envelope of the signal. This descriptor can be utilized to distinguish between a decaying piano note and a sustained organ note, when the lengths and the attacks of the two notes are identical.

Timbral Spectral Descriptors The Timbral Spectral descriptors are spectral features extracted in a linear-frequency space. The analysis of harmonic structure is particularly useful to capture the perception of musical timbre. Pitched musical instruments illustrate a high degree of harmonic spectral quality.

The Harmonic Spectral Centroid (HSC) descriptor is defined as the average, over the signal duration, of the amplitude-weighted mean of the frequency of the bins (the harmonic peaks of the spectrum) in the linear power spectrum. HSC has a semantic similar to the other centroid descriptors such as the Audio Spectrum Centroid (ASC), but applies only to the harmonic (non-noise) parts of the musical tone and is used in distinguishing musical instrument timbres. It is has a high correlation with the perceptual feature of "sharpness" of a sound.

Other timbral spectral descriptors are the following:

The Harmonic Spectral Deviation (HSD) measures the spectral deviation of the harmonic peaks from the envelopes of the local envelopes.

The Harmonic Spectral Spread (HSS) measures the amplitude-weighted standard deviation (Root Mean Square) of the harmonic peaks of the spectrum, normalized by the HSC.

The Harmonic Spectral Variation (HSV) is the normalized correlation between the amplitude of the harmonic peaks between two subsequent time-slices of the signal.

4.2.2.5 Spectral Basis Representations

The Spectral Basis descriptors represent low-dimensional projections of a high-dimensional spectral space to aid compactness and recognition. These descriptors are utilized for audio classification and indexing applications.

The Audio Spectrum Basis (ASB) is a series of (potentially time-varying and/or statistically independent) basis functions that are derived from the Singular Value Decomposition (SVD) or Independent Component Analysis (ICA) of a normalized power spectrum.

The Audio Spectrum Projection descriptor is used together with the ASB descriptor and represents low-dimensional features of a spectrum after projection upon a reduced-rank basis. Together, the descriptors may be used to view and to represent compactly the independent subspaces of a spectrogram. Often these independent subspaces correlate strongly with different sound sources.

4.3 MARSYAS: Audio Content Features

MARSYAS [24, 25] is a framework which provides a set of content-based features for audio files. In this section, we will present a summary of those features that are utilized in this dissertation for the content-based analysis of music files. Specifically, each music piece is represented by a 30-dimensional objective feature vector, which forms a mathematical abstraction attempting to encapsulate the information content of the audio signal contained in each music file. More specifically, the objective (i.e., computed directly from the audio signal) music characteristics can be categorized into three different types of features that are identified as *(a) Music Surface-*, *(b) Rhythm-*, and *(c) Pitch*-related features.

Each file, which has a duration of 30 s, is used as input to a feature extraction module. Specifically, short time audio analysis is used in order to break the signal into small, possibly overlapping temporal segments of duration of 50 ms (covering the entire duration of 30 s) and process each segment separately. These segments are called "analysis windows" or "frames" and need to be short enough for the frequency characteristics of the magnitude spectrum to be relatively stable. On the other hand, the term "texture window" describes the shortest window (minimum amount of sound) that is necessary to identify music texture. The texture window is set equal to 30 s in our system.

The actual objective features used in our system are the running mean, median and standard deviation of audio signal characteristics computed over a number of analysis windows. The feature vector constituents appear in Table 4.1.

4.3.1 Music Surface Features

For the purpose of pattern recognition/classification of music files, we use the statistics of the spectral distribution over time of the corresponding audio signals and represent the "musical surface" [4, 24, 25]. Some of these statistics are defined next.

- **Spectral Centroid**: This feature reflects the *brightness* of the audio signal and is computed as the balancing point (centroid) of the spectrum. It can be calculated as

$$C = \frac{\sum_{n=0}^{N-1} M_t[n] \cdot n}{\sum_{n=0}^{N-1} M_t[n]} \tag{4.1}$$

where $M_t[n]$ is the magnitude of the Fourier transform at frame t and frequency bin n.

- **Spectral Rolloff**: This feature describes the spectral *shape* and is defined as the frequency $R = R(r)$ that corresponds to r% of the magnitude distribution. It can be seen as a generalization of the spectral centroid, as the spectral centroid is the roll-off value that corresponds to $r = 50\,\%$ of the magnitude distribution. In our system, we used a roll-off value $r = 95\,\%$ which has been experimentally determined. N is the length of the discrete signal stored in vector x.

$$\sum_{n=0}^{R} M_t[n] = r \cdot \sum_{n=0}^{N-1} M_t[n] \tag{4.2}$$

- **Spectral Flux**: This feature describes the *evolution* of frequency with time and is computed as the difference of the magnitude of the short-time Fourier transform between the current and the previous frame. Therefore, the spectral flux is a measure of local spectral change, given by the equation

$$SF = \sum_{n=0}^{N-1} (N_t[n] - N_{t-1}[n])^2 \tag{4.3}$$

where $N_t[n]$ and $N_{t-1}[n]$ is the normalized magnitude of the short-time Fourier transform at window t and $t-1$, respectively.

- **Zero-Crossings**: A zero-crossing occurs when successive samples in a digital signal have different signs. The corresponding feature is defined as the number of time-domain zero-crossings in the signal. This feature is useful in detecting the *amount of noise* in a signal and can be calculated as

$$Z_n = \sum_m |\mathrm{sgn}[x(m)] - \mathrm{sgn}[x(m-1)]| \cdot w(n-m) \tag{4.4}$$

where

$$\mathrm{sgn}[x(n)] = \begin{cases} 1, & \mathrm{x(n)} \geq 0 \\ 0, & \mathrm{x(n)} < 0 \end{cases} \tag{4.5}$$

and

$$w(m) = \begin{cases} \frac{1}{2}, & 0 \leq m \leq N-1 \\ 0, & \text{otherwise.} \end{cases} \tag{4.6}$$

- **Short-Time Energy Function**: The short-time energy of an audio signal $x(m)$ is defined as

$$E_n = \frac{1}{N} \sum_m [x(m) \cdot w(n-m)]^2 \tag{4.7}$$

where

$$w(m) = \begin{cases} 1, & 0 \leq m \leq N-1 \\ 0, & \text{otherwise.} \end{cases} \tag{4.8}$$

In Eqs. 4.4 and 4.8, $x(m)$ is the discrete-time audio signal, n is the time index of the short time energy and $w(m)$ is a rectangular window. This feature provides a convenient representation of the *temporal evolution* of the audio signal amplitude variation.

- **Mel-Frequency Cepstral Coefficients (MFCCs)**: These coefficients are designed to capture *short-term spectral features*. After taking the logarithm of the amplitude spectrum obtained from the short-time Fourier transform of each frame, the frequency bins are grouped and smoothed according to the Mel-frequency scaling, which has been designed in agreement with human auditory perception. MFCCs are generated by decorrelating the Mel-spectral vectors with a discrete cosine transform.

4.3.2 Rhythm Features and Tempo

Rhythm features characterize the movement of music signals over time and contain information as *regularity of the tempo*. The feature set for representing rhythm is extracted from a *beat histogram*, that is a curve describing beat strength as a function of tempo values, and can be used to obtain information about the complexity of the beat in the music file. The feature set for representing rhythm structure is based on detecting the most salient periodicities of the signal and it is usually extracted from the beat histogram. To construct the beat histogram, the time domain amplitude envelope of each band is first extracted by decomposing the music signal into a number of octave frequency band. Then, the envelopes of each band are summed together followed by the computation of the autocorrelation of resulting sum envelop. The dominant peaks of the autocorrelation function, corresponding to the various periodicities of the signal envelope, are accumulated over the entire sound file into a beat histogram, in which each bin corresponds to the peak lag. The rhythmic content features are then extracted from the beat histogram and, generally, include the relative amplitude of the first and the second histogram peak, the ratio of the amplitude of the second peak divided by the amplitude of the first peak, the periods of the first and second peak, and the overall sum of the histogram.

4.3.3 Pitch Features

The pitch features describe *melody* and *harmony* information in a music signal. A pitch detection algorithm decomposes the signal into two frequency bands and amplitude envelopes are extracted for each frequency band where the envelope extraction is performed via half-way rectification and low-pass filtering. The envelopes are summed and an enhanced autocorrelation function is computed so as to reduce the effect of integer multiples of the peak of frequencies to multiple pitch detection. The dominant peaks of the autocorrelation function are accumulated into pitch histograms and the pitch content features extracted from the pitch histograms. The pitch content features typically include the amplitudes and periods of maximum peaks in the histogram, pitch intervals between the two most prominent peaks, and the overall sums of the histograms.

Table 4.1 summarizes the objective feature vector description.

Table 4.1 Feature vector of MARSYAS

Feature ID	Feature name
1	Mean centroid
2	Mean rolloff
3	Mean flux
4	Mean zero-crossings
5	STD of centroid
6	STD of rolloff
7	STD of flux
8	STD of zero-crossings
9	Low energy
[10 . . . 19]	MFCCs
20	Beat A0
21	Beat A1
22	Beat RA
23	Beat P1
24	Beat P2
25	Beat sum
26	Pitch FA0
27	Pitch UP0
28	Pitch FP0
29	Pitch IP0
30	Pitch sum

References

1. Camastra, F., Vinciarelli, A.: Machine Learning for Audio, Image and Video Analysis: Theory and Applications. Springer, London (2007)
2. Cord, M., Cunningham, P.: Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval. Springer, Heidelberg (2008)
3. Divakaran, A.: Multimedia Content Analysis Theory and Applications. Springer, Heidelberg (2008)
4. Foote, J.: Content-based retrieval of music and audio. In: Proceedings of Storage and Retrieval for Image and Video Databases (SPIE), vol. 3229, pp. 138–147 (1997)
5. Gong, Y., Xu, W.: Machine Learning for Multimedia Content Analysis. Springer, NEC Laboratories America Inc., New York (2007)
6. Kim, H.G., Moreau, N., Sikora, T.: MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval. Wiley, New York (2005)
7. Kompatsiaris, Y., Hobson, P.: Semantic Multimedia and Ontologies: Theory and Applications. Springer, Heidelberg (2008)
8. Manjunath, B.S., Salembier, P., Sikora, T.: Introduction to MPEG-7: Multimedia Content Description Interface. Wiley, New York (2002)
9. Martinez, J.M.: MPEG-7: overview of MPEG-7 description tools, part 2. IEEE Multimed. **9**(3), 83–93 (2002). doi:10.1109/MMUL.2002.1022862
10. MPEG-7: Visual experimentation model (xm), version 10.0 iso/iec/jtc1/sc29/wg11, doc. n4063 (2001)
11. MPEG-7: MPEG-7 overview, version 10.0 iso/iec/jtc1/sc29/wg11, doc. n6828 (2004)
12. MPEG-7: Multimedia content description interface, reference software iso/iec/15938-6 (2005). Software http://standards.iso.org/ittf/PubliclyAvailableStandards/c035364_ISO_IEC_15938-6(E)_Reference_Software.zip
13. MPEG-7: Mpeg-7 audio reference software toolkit iso/iec/15938-4 part 6 (2007). Software http://mpeg7.doc.gold.ac.uk/mirror/index.html
14. Nixon, M.S., Aguado, A.S.: Feature Extraction and Image Processing. Academic Press, London (2008)
15. Petrou, M., Bosdogianni, P.: Image Processing: The Fundamentals. Wiley, New York (1999)
16. Petrushin, V.A., Khan, L.: Multimedia Data Mining and Knowledge Discovery. Springer, Heidelberg (2007)
17. Sikora, T.: The mpeg-7 visual standard for content description—an overview. IEEE Trans. Circuits Syst. Video Technol. **11**(6), 696–702 (2001)
18. Sotiropoulos, D.N., Lampropoulos, A.S., Tsihrintzis, G.A.: Musiper: a system for modeling music similarity perception based on objective feature subset selection. User Model. User-Adapt. Interact. **18**(4), 315–348 (2008)
19. Thuraisingham, B.M.: Managing and Mining Multimedia Databases. CRC Press Inc., Boca Raton (2001)
20. Tsihrintzis, G.A., Jain, L.C. (eds.): Multimedia Services in Intelligent Environments—Advanced Tools and Methodologies, Studies in Computational Intelligence, 120th edn. Springer, Berlin (2008)
21. Tsihrintzis, G.A., Jain, L.C.: Multimedia Services in Intelligent Environments—Integrated Systems. Studies in Computational Intelligence. Springer, Berlin (2010)
22. Tsihrintzis, G.A., Virvou, M., Howlett, R.J., Jain, L.C. (eds.): New Directions in Intelligent Interactive Multimedia. Studies in Computational Intelligence, 142nd edn. Springer, Berlin (2008)
23. Tsihrintzis, G.A., Virvou, M., Jain, L.C. (eds.): Multimedia Services in Intelligent Environments—Software Development Challenges and Solutions. Studies in Computational Intelligence. Springer, Berlin (2010)
24. Tzanetakis, G.: Manipulation, analysis and retrieval systems for audio signals. Ph.D. thesis, Princeton University (2002)

25. Tzanetakis, G., Cook, P.: Marsyas: a framework for audio analysis. Organ. Sound **4**(3), 169–175 (2000)
26. Vaseghi, S.V.: Multimedia Signal Processing: Theory and Applications in Speech, Music and Communications. Wiley, New York (2007)
27. Virvou, M., Jain, L.C. (eds.): Intelligent Interactive Systems in Knowledge-based Environments. Studies in computational intelligence, 104th edn. Springer, Berlin (2008)
28. Wu, J.K.K., Hong, D., Kankanhalli, M.S., Lim, J.H.: Perspectives on Content-Based Multimedia Systems. Kluwer Academic Publishers, Norwell (2000)

Chapter 5
Similarity Measures for Recommendations Based on Objective Feature Subset Selection

Abstract In this chapter, we present a content-based RS for music files, called *MUSIPER*, in which individualized (*subjective*) music similarity perception models of the system users are constructed from *objective* audio signal features by associating different music similarity measures to different users. Specifically, our approach in developing MUSIPER is based on investigating certain subsets in the *objective* feature set and their relation to the *subjective* music similarity perception of individuals.

5.1 Introduction

Our starting point has been the fact that each individual perceives differently the information features contained in a music file and assigns different degrees of importance to music features when assessing similarity between music files. This leads to the hypothesis that different individuals possibly assess music similarity via different feature sets and there might even exist certain features that are entirely unidentifiable by certain users. On the basis of this assumption, we utilize relevance feedback from individual users in a neural network-based incremental learning process in order to specify that feature subset and the corresponding similarity measure which exhibit the maximum possible accordance with the user's music similarity perception. The approach followed in MUSIPER can be followed with any type of multimedia data. For example, a variation of MUSIPER has been implemented and evaluated to function with image data [5].

5.2 Objective Feature-Based Similarity Measures

In Chap. 4, we described in detail the procedure which computes the (Marsyas 30-dimensional feature vector) feature vector, the constituents of which are utilized as the objective features to rank similarity between music pieces. In this

The acronym MUSIPER stands for MUsic SImilarity PERception.

A.S. Lampropoulos and G.A. Tsihrintzis, *Machine Learning Paradigms*,
Intelligent Systems Reference Library 92, DOI 10.1007/978-3-319-19135-5_5

section, we describe the process through which the values of the constituents of the corresponding feature vectors are combined to generate a single value that represents the degree of similarity between two music pieces. Simply listing all features for a pair of music pieces and determining their overlap is not sufficient to model music similarity perception [3, 7]. In our approach, this is achieved through the definition of an appropriate similarity measure that exhibits the intrinsic ability to combine the constituent values of the heterogeneous feature vector into the corresponding similarity value between two music pieces. Moreover, the required similarity measure ought to involve a substantially plastic learning ability that would serve the primary purpose of MUSIPER, that is the ability to construct efficient user models that reflect the information provided by their corresponding users.

Radial Basis Function Networks (RBFNs) can serve as an ideal computational equivalent of the previously described similarity measure as they are capable of realizing essentially any non-linear mapping between spaces of high dimensions and, therefore, approximating the non-linear function that maps the set of heterogeneous feature vector values into a single similarity value [1, 2]. Moreover the back propagation learning rule empowers them with the learning capability that justifies these computational models as suitable user model constructors.

Our approach was based on investigating links between objective audio signal features and subjective music similarity perception. More specifically, we hypothesized that there exist certain subsets of the original feature vector that could be more salient for a certain individual as he/she valuates the perceived similarity of two music pieces. For this reason, we utilized a number of neural networks forced to function on the basis of different feature subsets of the original feature vector and, thus, realize different similarity measures. Each feature subset corresponded to a different type of audio features or their combinations. A detailed description of the specific feature subsets realized in MUSIPER is included in the following section.

5.3 Architecture of MUSIPER

Modeling the subjective similarity perception of a certain individual may be computationally realized by the development of an appropriate similarity measure providing the degree of resemblance between two music pieces as a real value in the [0, 1] interval. Thus, the user modeling functionality embedded in our system consists of developing similarity measures which would approximate the similarity values that would be assigned by a specific user to pairs of music pieces. From a mathematical point of view a similarity measure may be interpreted as a continuous non-linear mapping ($F : R^n \times R^n \rightarrow [0, 1], n \leq 30$) from the space of objective features to the [0, 1] interval of similarity degrees which naturally leads us to the choice of Radial Basis Functions Networks (RBFN's) that are capable of implementing arbitrary nonlinear transformations of the input space. Moreover, the adopted incremental learning procedure lies in the core of the training process where the

internal network parameters are modified according to the back propagation rule in response to the user supplied similarity values concerning certain pairs of music pieces.

5.4 Incremental Learning

The overall system architecture is based on the adapted incremental learning technique depicted in Fig. 5.1. Specifically, our scheme consists of the following steps:

1. Seed the search with the *target* music piece corresponding to an existing music piece in the system database. This step uses an *offline process*, where the feature extractor extracts the set of values for the complete set of 30 features. Afterwards, a predefined number of subsets from the original feature vectors set, $C_1,\ldots,C_M$, (e.g. $M = 11$ neural networks in the MUSIPER) are assessed for their ability to capture the subjective music piece similarity perception of a specific user. These subsets of the feature vector are fed into the corresponding neural networks, which constitute database searchers (DBSearchers) running in parallel and realizing M different similarity measures (Fig. 5.1).

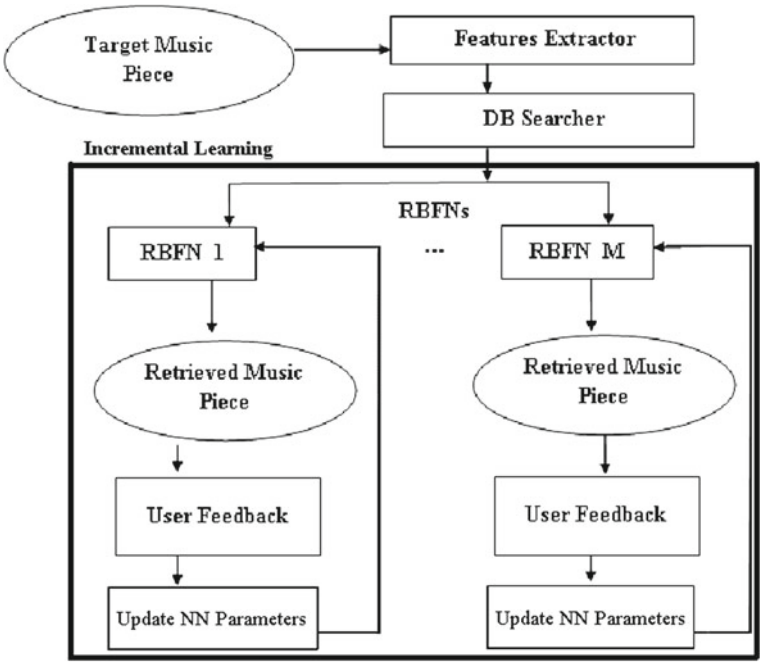


Fig. 5.1 MUSIPER architecture

2. Each neural network retrieves the most similar music piece according to the similarity measure it realizes.
3. The user valuates the set of the retrieved music pieces and ranks the degree of similarity between the retrieved music pieces and the target music piece according to his/her own perception. The user supplied similarity values are stored in matrix E where $E = [e_0, e_1, \ldots, e_n]^T$.
4. This information is subsequently used by the system in order to adjust the neural network parameters stored in matrix W where $W = [w_0, w_1, \ldots, w_n]^T$ as it is implied by Eq. 5.13. This latter parameter refinement involves the adaptation of the entire neural network parameter set and constitutes the fundamental part of the adopted training scheme.
5. The procedure is repeated for a preset number of times, during which the network performance is recorded. In the end, we identify the neural network and the corresponding feature subset that exhibited the most effective performance in modeling the music similarity perception of the specific user.

5.5 Realization of MUSIPER

The topology of a RBFN involves a set of three layers as shown in Fig. 5.2, where the first layer constitutes the input layer, the second layer is a hidden layer and the third layer is the output layer. We must clarify that the number of hidden nodes for each neural network is the same and does not depend on the number of the selected features for that network as it was experimentally set to five in order to ensure that the networks converge. Generally, the input, hidden and output layers contain a set of $p+1$ (equal to the number of signals/features to be processed), N and only one node, respectively. Each input node is connected with every hidden layer node and each hidden layer node is connected with the output node. All these connections are called *synapses* and are given associated *synaptic weights*. From the system theoretic point of view, the transfer function between the input and the hidden layer is non-linear, while the transfer function between the hidden and the output layer is linear

In more detail, the output layer node realizes the function:

$$y_{out}(n) = u_{out}(n), \tag{5.1}$$

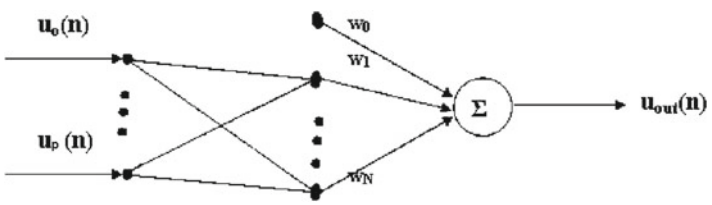


Fig. 5.2 RBFN architecture

where

$$u_{out}(n) = \sum_{j=0}^{N} w_j \cdot y_j(n). \tag{5.2}$$

In Eq. 5.2, w_j is the connection weight between the jth hidden layer node and the output node, $y_j(n)$ is the output of the jth node in the hidden layer corresponding to the nth input pattern, and $y_{out}(n)$ is the output of the output node after the nth input pattern has been presented to the network and represents the similarity value between two music pieces M_A, M_B. The jth node of the hidden layer realizes the following function:

$$y_0(n) = 1 \tag{5.3}$$

and

$$y_j(n) = \exp - \left\{ \frac{\left\| \mathbf{v}^n - \mu_j(n) \right\|^2}{2 \cdot \sigma_j^2(n)} \right\}, \quad 1 \leq j \leq N. \tag{5.4}$$

The zeroth node of the hidden layer can be considered as a bias term which is added to the overall output of the network. Moreover, $\mathbf{v}^n$ is the nth input pattern, $\mu_j(n)$ and $\sigma_j(n)$ are the center and the spread, respectively, of the radial basis function realized by the jth node in the hidden layer, and $y_j(n)$ is the output of the jth hidden node. The parameter n indicates that we refer to the time instant after the nth input pattern has been presented to the network. On the other hand, the term $\left\| \mathbf{v}^n - \mu_j(n) \right\|$ corresponds to the Euclidean distance between the vectors $\mathbf{v}^n$ and $\mu_j(n)$, where $\mathbf{v}^n$, $\mu_j(n) \, \epsilon \, \mathbb{R}^p$.

Now, assume that C_A and C_B are the feature vector values that have been extracted from the music pieces M_A and M_B, where $C_A = [C_{A_1}, \ldots, C_{A_p}]$ and $C_B = [C_{B_1}, \ldots, C_{B_p}]$. According to the previous definitions of C_A and C_B, the input to the neural network is denoted by $\mathbf{v} = [v_1, \ldots, v_p]$, where

$$v_i = \left| C_{A_i} - C_{B_i} \right|, \qquad 1 \leq i \leq p. \tag{5.5}$$

The fundamental adjustable parameters of the RBFN are those related to the radial basis functions realized by the nodes of the hidden layer and the connection weights between the hidden nodes and the output node of the network. Thus, the set of the RBFN parameters includes the parameter set of each of the N nodes of the hidden layer (and the corresponding radial basis functions Φ_j, $1 \leq j \leq N$) that can be presented in the general form $\Phi_j(\mathbf{v}; \mu_j; \sigma_j)$, together with the weight vector $\mathbf{w} = [w_0, w_1, \ldots, w_N]$. Each of the used radial basis functions performs a mapping $\Phi : \mathbb{R}^p \longrightarrow \mathbb{R}$ where Φ_j is given by the equation

$$\Phi_j = \exp\left(- \frac{\left\| \mathbf{v} - \mu_j \right\|^2}{2 \cdot \sigma_j^2} \right). \tag{5.6}$$

The approximation ability of the RBFN lies in the adaptability of its parameters, which allows us to train the RBFN to *learn* (approximate) essentially any desirable similarity measure. The appropriate parameter values can be determined by using a training set of input patterns that will force the RBFN parameters and the corresponding input-output relation to attain the appropriate form. Specifically, the set of training patterns is comprised of pairs of music feature vectors and the corresponding similarity values as subjectively perceived by a certain user.

The training process of the RBFN can be reduced to a set of two distinct training stages. The first stage can be considered as a *pre*training process in which the weight values are the only parameters to be modified. This stage serves as a weight initialization procedure that corrects a random initial parameter setting. The input pattern set used in the first training stage contains a number of elements equal to the number of nodes in the hidden layer of the network. Thus, we need a set of pairs of music feature vectors and corresponding similarity values that reflect the objective degree of similarity. We made use of a static objective similarity measure during the first training stage, so as to achieve the weight values that are most appropriate for modeling the subjective perception of a specific user.

After the end of the first training stage, the parameter modification procedure changes so that the entire parameter set $(\mu_j, \sigma_j, \mathbf{w})$ is adjusted with the presentation of every training pattern. In this way, the network behavior is refined in order to approximate the desirable subjective similarity perception.

In order to compute the initial weight values, we must first consider the training input pattern set which consists of $N + 1$ music piece feature vectors, namely $(C_A^0, C_B^0), (C_A^1, C_B^1), \ldots, (C_A^N, C_B^N)$. The corresponding similarity values are denoted by the variables $e_0, e_1, \ldots, e_N$ where

$$e_k = \left\| C_A^k - C_B^k \right\|, \quad 0 \leq k \leq N. \tag{5.7}$$

The input vectors for the corresponding feature vectors are: $\mathbf{v}^0, \mathbf{v}^1, \ldots, \mathbf{v}^N$, where:

$$v_i^k = \left| C_{A_i}^k - C_{B_i}^k \right|, \quad 0 \leq k \leq N, \quad 1 \leq i \leq p. \tag{5.8}$$

Each of the desired similarity values $e_0, e_1, \ldots, e_N$ must equal the network output after the presentation of the corresponding difference feature vectors $\mathbf{v}^0, \mathbf{v}^1, \ldots, \mathbf{v}^N$. Thus, we get the set of equations

$$e_k = w_0 + \sum_{j=1}^{N} \Phi_j(\left\| \mathbf{v}^k - \mu_j \right\| ; \sigma_j) \cdot w_j. \tag{5.9}$$

Equation 5.7 is readily put into the matrix form:

$$\begin{bmatrix} e_0 \\ e_1 \\ \vdots \\ e_N \end{bmatrix} = \begin{bmatrix} 1 & \Phi_1(\left\|\mathbf{v}^0 - \mu_1\right\|;\sigma_1) & \cdots & \Phi_N(\left\|\mathbf{v}^0 - \mu_N\right\|;\sigma_N) \\ 1 & \Phi_1(\left\|\mathbf{v}^1 - \mu_1\right\|;\sigma_1) & \cdots & \Phi_N(\left\|\mathbf{v}^1 - \mu_N\right\|;\sigma_N) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \Phi_1(\left\|\mathbf{v}^N - \mu_1\right\|;\sigma_1) & \cdots & \Phi_N(\left\|\mathbf{v}^N - \mu_N\right\|;\sigma_N) \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_N \end{bmatrix} \tag{5.10}$$

or in the abbreviated form:

$$E = \Phi W \tag{5.11}$$

where $E = [e_0, e_1, \ldots, e_n]^T$, $W = [w_0, w_1, \ldots, w_n]^T$ and

$$\Phi_{r,c} = \begin{cases} 1, & 0 \le r \le N, c = 1 \\ \Phi_c(\|\mathbf{v}^r - \mu_c\|;\sigma_c), & 0 \le r \le N, 1 \le c \le N. \end{cases} \tag{5.12}$$

Clearly, the initial weight vector can derived from the equation:

$$W = \Phi^{-1}E. \tag{5.13}$$

However, the matrix Φ in the previous equations is usually substituted by the matrix $(\Phi + \lambda I)$ where $\lambda \ge 0$ so that matrix $(\Phi + \lambda I)$ is always invertible. Thus, Eq. 5.13 becomes:

$$W = (\Phi + \lambda I)^{-1}E. \tag{5.14}$$

5.5.1 Computational Realization of Incremental Learning

In order to complete the description of the training process of the RBFN, we refer to the second training stage. This stage coincides with the adopted incremental learning procedure, during which the entire parameter set is modified simultaneously. Our description at this point will be restricted to giving the equations for modifying each of the network parameters as they are derived via application of the back-propagation algorithm. We have:

$$w_0(n+1) = w_0(n) + \triangle w_0(n), \tag{5.15}$$

where $\triangle w_0(n)$ is the correction for the w_0 weight constituent after the presentation of the nth training pattern.

Similarly,

$$w_j(n+1) = w_j(n) + \triangle w_j(n), \quad 1 \leq j \leq N, \tag{5.16}$$

where $\triangle w_j(n)$ is the correction for the w_j weight constituent after the presentation of the nth training pattern. Also:

$$\mu_{j_i}(n+1) = \mu_{j_i}(n) + \triangle \mu_{j_i}(n), \quad 1 \leq j \leq N, \tag{5.17}$$

where $\triangle \mu_{j_i}(n)$ is the correction of the ith constituent of the jth function center and

$$\sigma_j(n+1) = \sigma_j(n) + \triangle \sigma_j(n), \quad 1 \leq j \leq N, \tag{5.18}$$

where $\triangle \sigma_j(n)$ is the correction of the jth function spread.

The correction values are given by the following equations

$$\triangle w_0(n) = n_1 \cdot e(n), \tag{5.19}$$

where

$$e(n) = e_n - y_{out}(n) \tag{5.20}$$

is the network error at the presentation of the nth training pattern, e_n is the desired similarity value and $y_{out}(n)$ is the network response with respect to the input pattern $\mathbf{v}^n$.

Similarly:

$$\triangle w_j(n) = n_2 \cdot e(n) \cdot \exp\left(-\frac{\left\|\mathbf{v}^n - \mu_j(n)\right\|^2}{2 \cdot \sigma_j^2(n)}\right), \tag{5.21}$$

$$\triangle \mu_{j_i}(n) = n_3 \cdot w_j(n) \cdot \exp\left(-\frac{\left\|\mathbf{v}^n - \mu_j(n)\right\|^2}{2 \cdot \sigma_j^2(n)}\right) \cdot \frac{v_i^n - \mu_{j_i}(n)}{2 \cdot \sigma_j^2(n)}, \tag{5.22}$$

$$\triangle \sigma_j(n) = n_4 \cdot w_j(n) \cdot \exp\left(-\frac{\left\|\mathbf{v}^n - \mu_j(n)\right\|^2}{2 \cdot \sigma_j^2(n)}\right) \cdot \frac{\left\|v^n - \mu_j(n)\right\|^2}{2 \cdot \sigma_j^3(n)}, \tag{5.23}$$

where n_1, n_2, n_3, n_4 are the corresponding learning rates for the parameters.

5.6 MUSIPER Operation Demonstration

MUSIPER has been developed as a desktop application whose main graphical user interface window appears in Fig. 5.3. The application window can be thought of as being divided into two regions by a vertical line. This line separates and rounds up

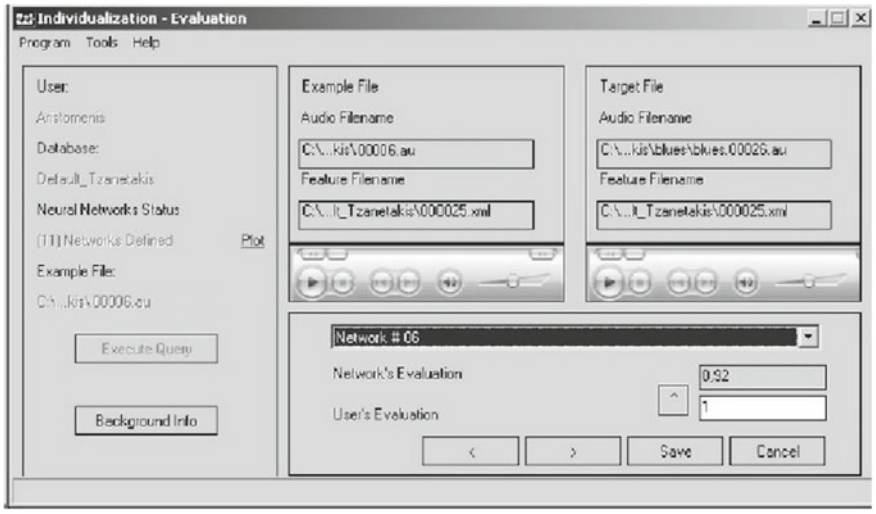


Fig. 5.3 MUSIPER interface

the different kinds of user activities that are supported by the system. Specifically, the left region accumulates the user interface components that are related to the music pieces database selection and the neural network manipulation. Moreover, this part provides the basic query conduction operations that incorporate the target music piece specification.

The user must provide his/her own similarity estimation for each one of the music pieces retrieved by the corresponding neural networks. This functionality is provided in the right part of the application window and allows the user to listen to pairs of music pieces by interacting with two different components of Media Player and subsequently typing in his/her own similarity estimation value. The user interaction session ends when the user evaluates the music similarity for the complete set of training examples for the total amount of training stages.

A very important functionality provided by MUSIPER is that its user is not obligated to train MUSIPER in a single session. Instead, there is the capability of saving the training session at any instant and continuing at a future moment after loading his/her training profile.

5.7 MUSIPER Evaluation Process

MUSIPER was evaluated by one hundred (100) users belonging to different age groups and having various music style preferences and related music education levels. As a test database, we used a collection of one thousand (1000) western music pieces.

Table 5.1 Statistics of evaluation stage I

Overall favorite music genre	Pop 54 %
Age range	18 to 61 ($\mu = 29, \sigma^2 = 5, 44$)
CDs owned	10 to 200 ($\mu = 113, \sigma^2 = 235, 11$)
Hours spent per week listening to music	1 to 15 ($\mu = 7, \sigma^2 = 2, 77$)
Get music from filesharing P2P	MP3 (47 %)
Get music from internet music stores	MP3 (17 %)
Get music from music stores	CDs (36 %)
Play musical instrument	41 %, mostly the guitar or the piano
Professionally involved in music	4 %
Participation in evaluation of other systems	56 %

This collection is publicly accessible and has been used as a test bed for assessing the relative performance of various musical genre classification algorithms [4, 6]. The collection contains one hundred (100) pieces from each of the following ten (10) classes of western music: *blues, classical, country, disco, hiphop, jazz, metal, pop, regge,* and *rock*. Each piece has a duration of thirty (30) seconds.

The evaluation process consisted of three stages: At first, each participant was asked to complete a questionnaire regarding user background information, which was collected for statistical purposes and is summarized in Table 5.1.

In the second step of the evaluation process, each participant was given a predefined set of 11 *pre-trained* neural networks with corresponding feature subsets as in Table 5.2. The process of selecting the feature subsets was not arbitrary. More specifically, first we defined feature subsets containing features from only one semantic category. The corresponding networks are identified with numbers 2, 5, and 7, respectively. Next, we considered combinations of the previous features from various semantic categories in order to identify combinations of features from different semantic categories which appear to be efficient. Corresponding networks included those identified with numbers 6, 8, 9 and 10. For example, neural networks identified with numbers 6 and 10 combine features from two (namely, pitch and beat related features) and three (combination of beat, pitch and MFCC features) semantic categories, respectively. Both networks are examples of feature combinations that exhibit poor efficiency in modeling music similarity perception. In contrast, neural networks identified with numbers 8 and 9 combine features from two semantic categories, namely, MFCC and pitch related features and music surface and pitch related features, respectively. These networks are examples of feature combinations that exhibit high efficiency in modeling music similarity perception. The participants were asked to feed back into the system a *perceived* degree of similarity to the target piece of the returns of each network. Each participant fed back into the system for a total number of six (6) training stages (epochs). Specifically, during the course

Table 5.2 Feature subsets per neural network

Network IDs	Feature subsets	Feature IDs
1	Complete feature set	[1 . . . 30]
2	All beat-related features	[20 . . . 25]
3	All mean-, standard deviation- and low energy-related features	[1 . . . 9]
4	All MFCC-related features	[10 . . . 19]
5	All pitch-related features	[26 . . . 30]
6	All beat- and pitch-related features	[20 . . . 30]
7	All mean-, standard deviation-, MFCC- and low energy-related features	[1 . . . 19]
8	All MFCC- and pitch-related features	[10 . . . 19], [26 . . . 30]
9	All mean-, standard deviation-, MFCC-, pitch- and low energy-related features	[1 . . . 19], [26 . . . 30]
10	All beat-, MFCC- and pitch-related features	[10 . . . 30]
11	Mean and standard deviation of zero-crossings and low energy features	[4, 8, 9]

of each training stage, the user listened to a previously selected music piece which served as the target song of the query and the corresponding most similar music pieces were retrieved from the database by each neural network. Next, the user was directed to supply his/her own similarity perception estimate for each one of the 11 pairs of songs by typing in a similarity value in the [0, 1] interval. At this point, the user was given the option to adapt the estimated similarity value provided by the system. After completing all of the six training stages for every RBFN by providing a total of 66 similarity values, each user conducted a save operation in order to update the record of RBFN performance history and the newly estimated adjustable parameter values.

Finally, completion of the neural network training stage was followed by the third evaluation stage during which each participant was prompted to provide some information concerning the overall training and retrieval performance of the system.

5.8 System Evaluation Results

The second stage of the evaluation process revealed that:

1. During the training session of each user, there were neural networks whose relevant performance in approximating the music similarity perception of that particular user was consistently better than that of the remaining neural networks. Figures 5.4 and 5.5 illustrate typical examples of this fact, as seen from the plots of the time evolution of the error rates of the various networks. More specifically in Fig. 5.4 it is clear that the feature subsets with identification numbers (IDs) {4, 9, 11} correspond to the neural networks with the best performance while in Fig. 5.5 the best neural networks are those functioning on the basis of the feature subsets with IDs {4, 8, 9}.
   We evaluated the retrieval performance of our system for each user taking as a measure of accuracy the mean error rate of the best neural network for that user. The best network for a user was the one that exhibited the lowest error rate. Table 5.3 summarizes the best neural network and corresponding mean error rate for a set of 25 users.

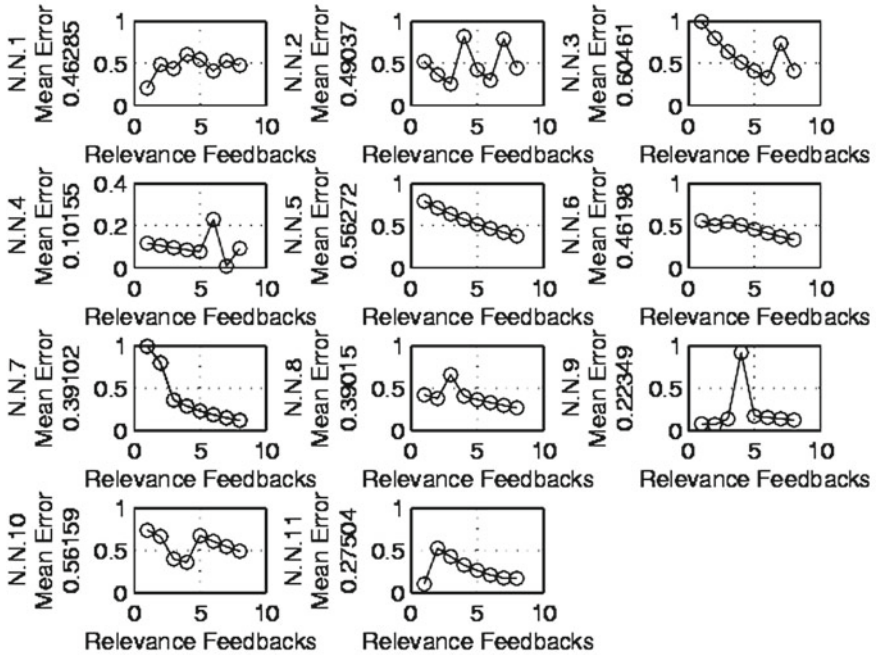


Fig. 5.4 Typical user behavior I

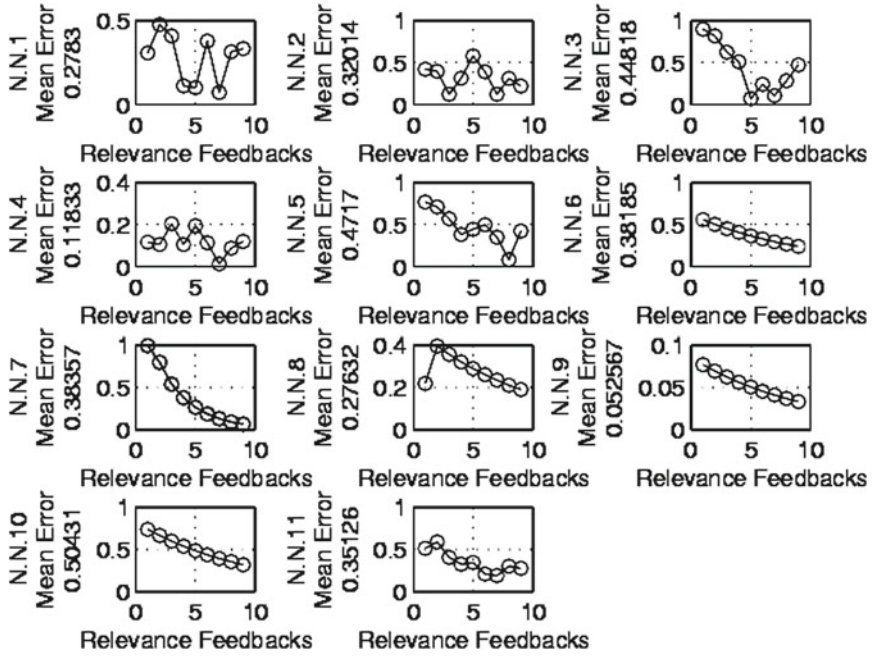


Fig. 5.5 Typical user behavior II

A further justification for the existence of certain neural networks, the ability of which in modeling the subjective similarity perception of a user was consistently better than that of the other networks is provided by comparing corresponding precision evolutions. Specifically, precision is computed as in Eq. 5.24 for the set of 11 neural networks. Figures 5.6 and 5.7 illustrate the precision evolution (precision value versus number of relevance feedbacks) for the complete set of neural networks for a specific user whose feedbacks produced the results in Fig. 5.12. Clearly, the neural network identified by number 7 demonstrates the best overall performance in modeling that specific user's similarity perception, since the corresponding precision value sequence reaches a stable saturation point after 9 retrievals/feedbacks. A similar saturated behavior is observed for the neural network identified by number 8, the precision of which saturates to a considerably lower value.

2. A second justification of the user modeling ability of our system lies in the observation that, even when certain neural network retrievals were assessed by the user as unsatisfactory, the similarity values estimated by the neural networks were quite close to the perceived similarity values provided by the user. A typical example of this observation is presented in Figs. 5.8 and 5.9, which compare the similarity values estimated by each neural network to the corresponding simi-

Table 5.3 Best neural network per user

User no	Mean error rate	NNs IDs
1	0.052567305	9
2	0.068916158	7
3	0.080797066	8
4	0.093290161	7
5	0.099306121	7
6	0.101553771	4
7	0.105191415	4
8	0.106715587	9
9	0.110695984	4
10	0.11238125	9
11	0.121527443	11
12	0.122296543	4
13	0.123571513	1
14	0.125683542	9
15	0.128536126	9
16	0.129964756	2
17	0.130741697	9
18	0.134278904	11
19	0.147017443	1
20	0.155501906	2
21	0.164345701	11
22	0.165889964	9
23	0.167231585	7
24	0.171717066	7
25	0.174795191	8

larity values provided by a user for a set of 6 relevance feedbacks. Specifically, it is observed that, even though neural networks identified by the numbers 1, 4 and 11 fail to provide efficient retrievals, their estimated similarity values are quite close to the ones provided by that user.

3. A third observation is that *no* single neural network and corresponding feature subset outperformed all networks in *all* training sessions. On the contrary, the system users are *clustered* by the eleven neural networks into 11 corresponding clusters as in Fig. 5.10. We observe that the neural networks numbered 5, 6 and 10 produce empty user clusters, which implies that the corresponding feature subsets fail to model the music similarity perception of any user. On the other hand, the neural networks numbered 9 and 7 produce clusters containing approximately

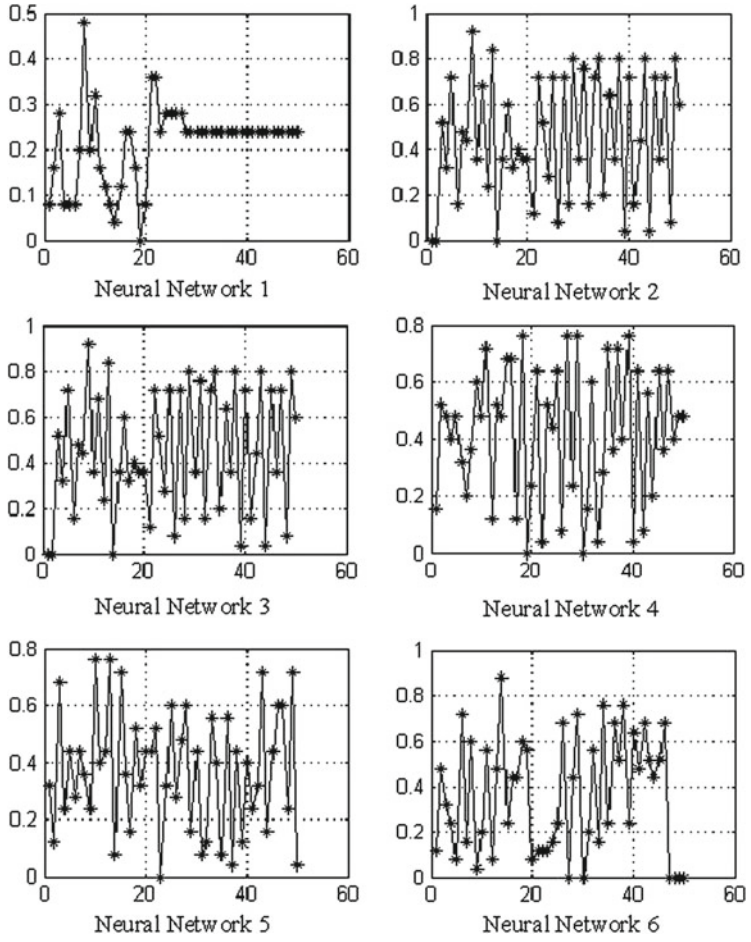


Fig. 5.6 Precision of neural networks 1–6 for the one user

27 and 18 % of the users. This difference in network performance lies with the qualitative differences of the corresponding feature subsets. Specifically, the feature subsets used by neural networks 9 and 7 describe both acoustic and psychoacoustic music information. This observation constitutes strong evidence justifying our initial hypothesis that relates feature subsets with the similarity perception of an individual user.

4. The convergence of the incremental learning process was examined and illustrated in Fig. 5.11. Specifically, the time evolution of the error rates of all the neural networks is shown over a total of 56 training cycles by the same user. During this training, a total of 8 different target music pieces were given and the system was trained for 7 epochs per given target.

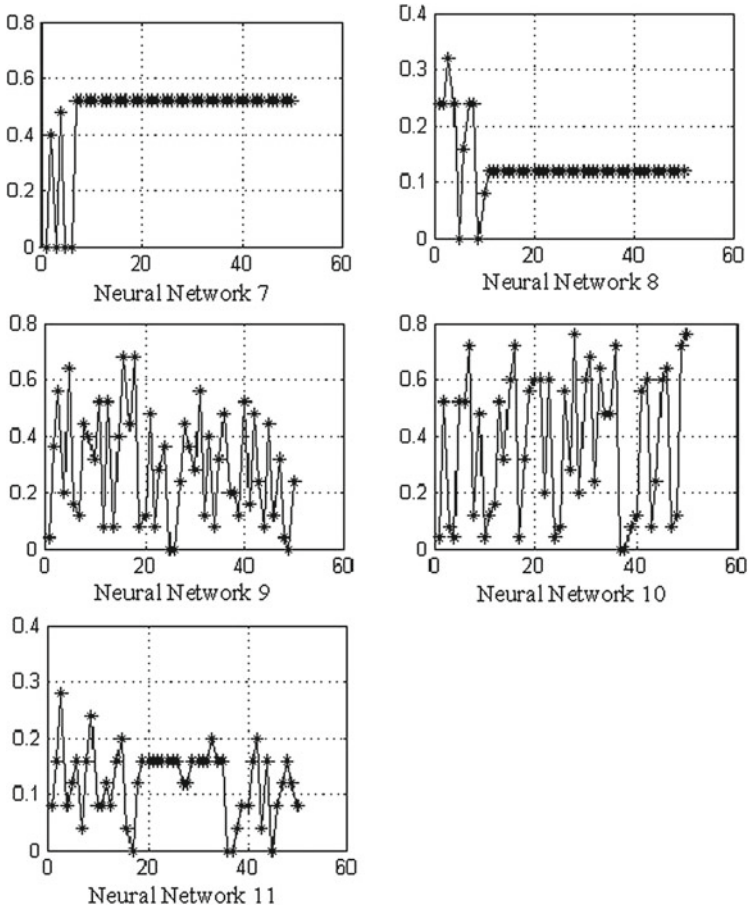


Fig. 5.7 Precision of neural networks 7–11 for the one user

5. It is important to observe the evolution of RBF centres and spreads during the incremental learning process (relevance feedbacks). Such observation reveals a smooth evolution of the centers and spreads of those RBFNs whose precision value saturated after a number of relevance feedbacks. We also noticed a strong correlation between the saturated precision value of a neural network and the corresponding evolution of its internal parameters. Specifically, when the precision saturated to a value and stabilized in its vicinity, the corresponding evolution of the network internal parameter values followed a smoother transition in subsequent feedbacks.
   The evolution of RBF centres and spreads during the incremental learning process is illustrated in Figs. 5.12, 5.13, 5.14 and 5.15 for four different users, respectively. Specifically, for each of the four users, the centre and spread evolution are shown for the neural network that exhibited the most effective per-

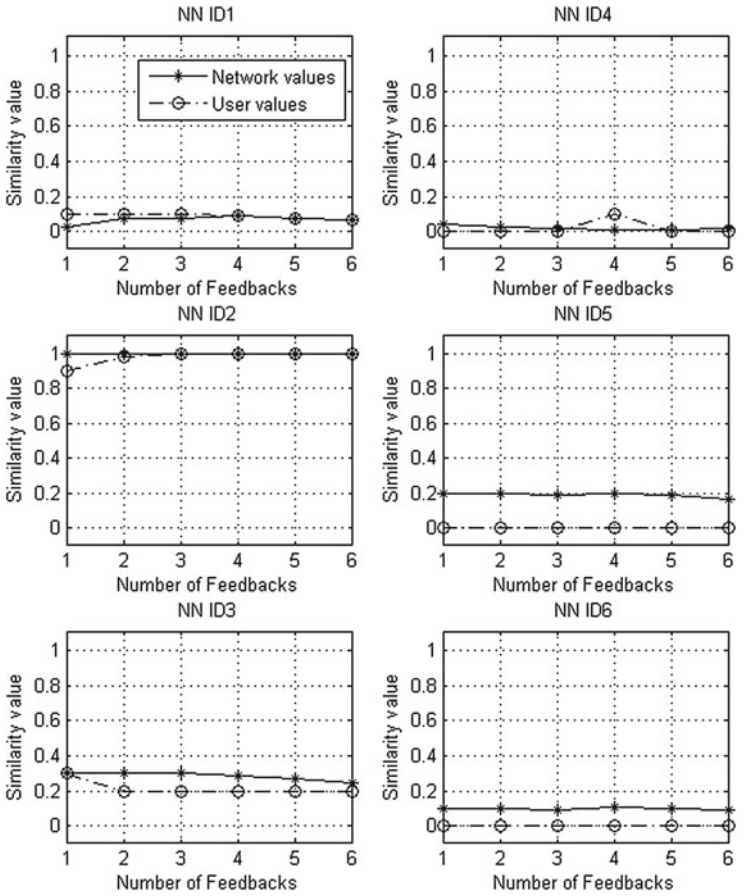


Fig. 5.8 Comparison between network-estimated and user-supplied similarity values

formance among all eleven networks in modeling the user music similarity perception. In each figure, the upper sub-figure shows Euclidean distance evolution of the RBF centres and spreads of the five nodes of the hidden network layer during the incremental learning process. The vertical and horizontal axes indicate Euclidean distance and number of relevance feedbacks, respectively. On the other hand, the lower sub-figure shows precision with respect to number of feedbacks, where precision is computed as

$$\text{precision} = \frac{\text{relevant music files retrieved in top } N \text{ returns}}{N}, \tag{5.24}$$

with $N = 25$.

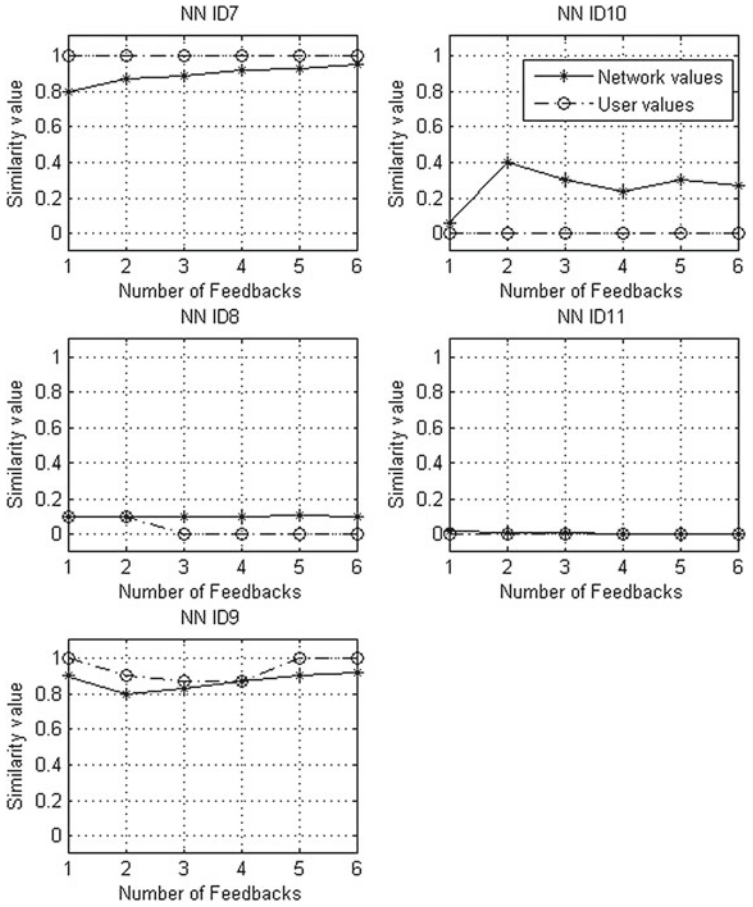


Fig. 5.9 Comparison between network-estimated and user-supplied similarity values

6. Further system tests were conducted, in which we experimented with various initialization patterns and presence of outliers in the user supplied relevance feedbacks. During the initialization process of MUSIPER (pretraining stage) there is no reason to expect that certain training patterns would be more "appropriate" to adjust the internal network parameters. This is because the system is initially trained so as to reflect the Euclidean distance between patterns. As the Euclidean distance is an *objective* similarity measure, there exist no preferable training patterns and any pair of target-retrieved feature vectors is equally appropriate for pretraining. The incremental learning process transforms the objective similarity measure provided by the Euclidean distance into a subjective one using the similarity values supplied by the user. Thus, the incremental learning

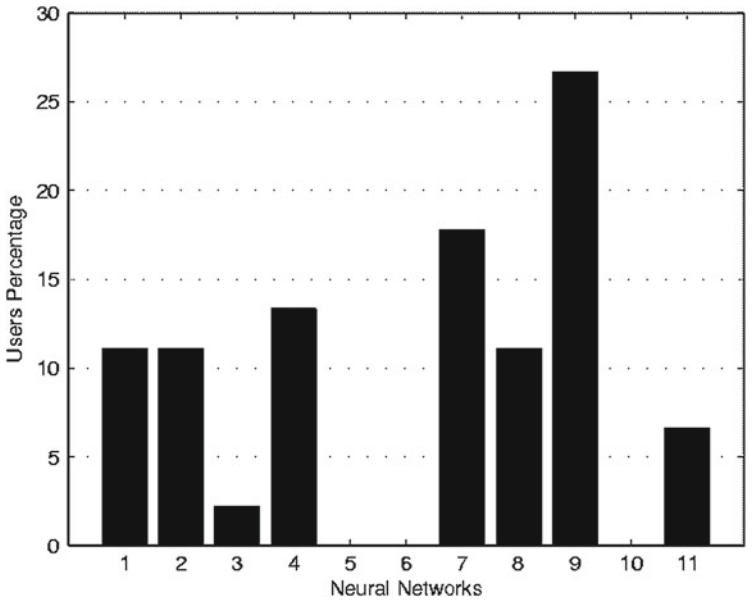


Fig. 5.10 Preferences distribution

procedure tunes the initial settings of the internal network parameters. However, the incremental learning process cannot be used for training from scratch as it would require a prohibitively high number of relevance feedbacks and system training time. These conclusions were experimentally corroborated.
On the other hand, the presence of outliers in the user-supplied feedbacks affects the training process by extending the time needed (number of training stages) for the neural networks to converge. However, as the number of relevance feedbacks increases the effect of outliers is gradually eliminated. This too was experimentally observed.

Table 5.4 summarizes the information collected during the third evaluation stage emphasizing the fact that the majority of the users observed the existence of certain neural networks whose retrievals were significantly better than the others. Moreover, most of the participants noticed a gradual improvement of the neural network responses from training stage to training stage.

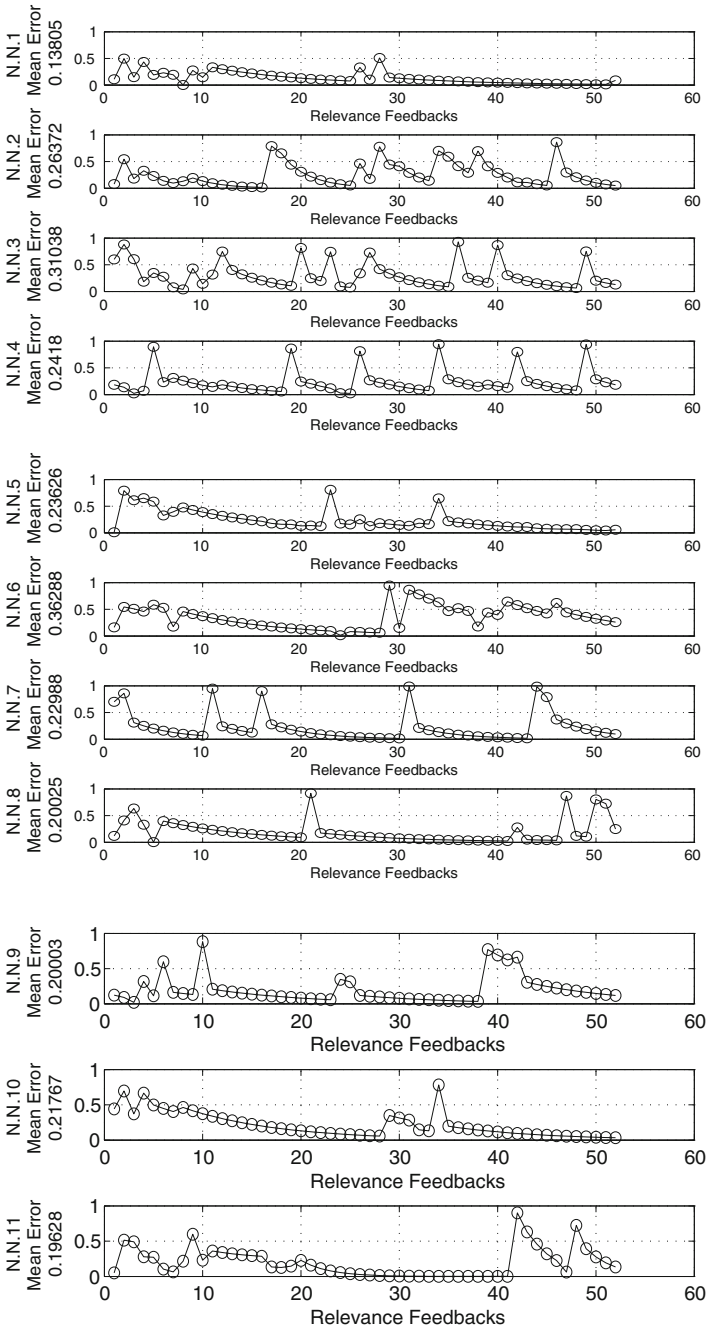


Fig. 5.11 Error rate convergence

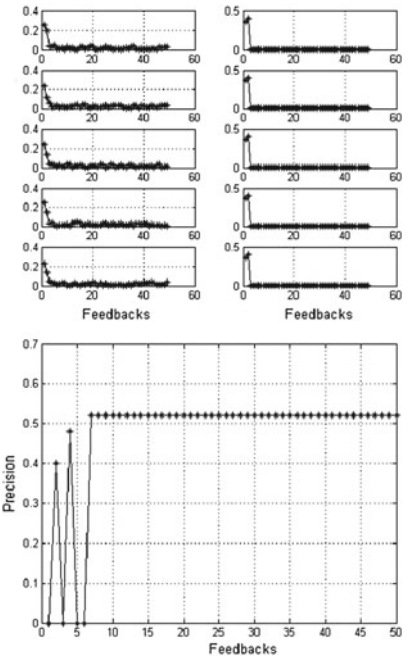


Fig. 5.12 Centers—spreads evolution precision of neural network—user 1

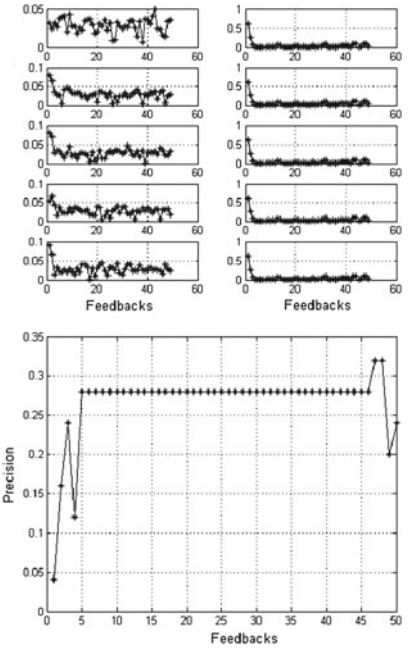


Fig. 5.13 Centers—spreads evolution precision of neural network—user 2

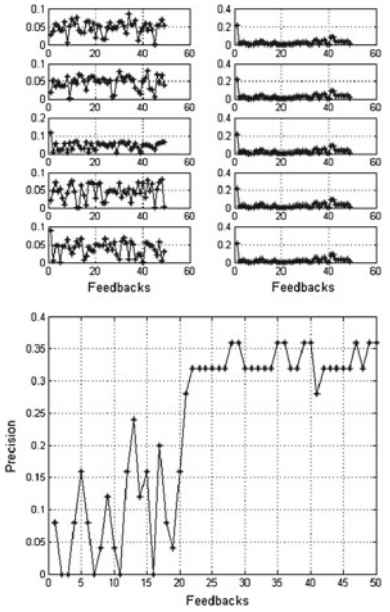


Fig. 5.14 Centers—spreads evolution precision of neural network—user 3

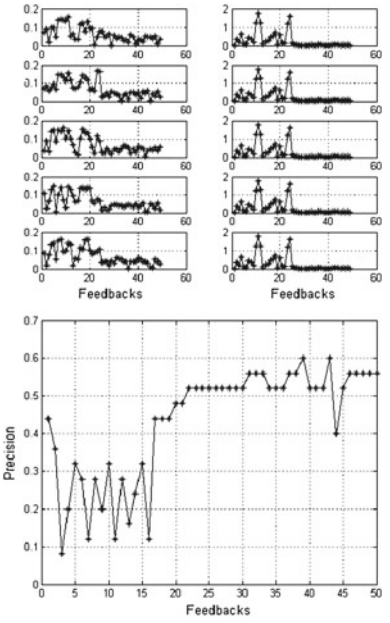


Fig. 5.15 Centers—spreads evolution precision of neural network—user 4

Table 5.4 Statistics of evaluation stage III

How long (in minutes) did you spent training the system?	27 mins on average (%)
Did you observe a difference in the retrievals returned by the various neural networks during the same training epoch? 1 (minimum difference) to 5 (maximum difference)	1 : 2 2 : 11 3 : 54 4 : 24 5 : 9
Did you observe an improvement in the retrievals returned by the various neural networks from training stage to training stage 1 (minimum improvement) to 5 (maximum improvement)	1 : 3 2 : 11 3 : 32 4 : 46 5 : 8
Did you observe any specific neural network that systematically returned better retrievals than the other networks 1 (minimum difference) to 5 (maximum difference)	1 : 2 2 : 22 3 : 34 4 : 36 5 : 6
Overall system assessment: 1 (Misleading) 2 (Not helpful) 3 (Good) 4 (Very good) 5 (Excellent)	2 17 27 32 22

References

1. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford, (1995)
2. Haykin, S.: Neural Networks, 2nd edn. Prentice Hall, Upper Saddle (1999)
3. Lampropoulos, A.S., Sotiropoulos, D.N., Tsihrintzis, G.A.: Individualization of music similarity perception via feature subset selection. In: Proceedings of IEEE International Conference on Systems, Man, and Cybernetics 2004. The Hague, The Netherlands (2004)
4. Li, T., Ogihara, M., Li, Q.: A comparative study on content based music genre classification. In: Proceedings of 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Toronto, Canada (2003)
5. Sotiropoulos, D.N., Lampropoulos, A.S., Tsihrintzis, G.A.: Multimedia Services in Intelligent Environments, chap. Individualization of Content-Based Image Retrieval Systems via Objective Feature Subset Selection, pp. 181–201. Springer (2008)
6. Tzanetakis, G., Cook, P.: Musical genre classification of audio signals. IEEE Trans. Speech Audio Process. **10**(5), 293–302 (2002)
7. Vignoli, F., Pauws, S.: A music retrieval system based on user driven similarity and its evaluation. In: Proceedings of 6th International Conference on Music Information Retrieval, pp. 272–279. London (2005)

Chapter 6
Cascade Recommendation Methods

Abstract In this chapter, we address the problem of recommendation by developing a two-level cascade classification architecture. The first-level classification step involves the incorporation of a one-class classifier which is trained exclusively on positive patterns. The one-class learning component of the first-level serves the purpose of recognizing instances from the class of desirable patterns as opposed to non-desirable patterns. On the other hand, the second-level classification step is based on a multi-class classifier, which is also trained exclusively on positive data. However, the second-level classifier is trained to discriminate among the various (sub-)classes from which positive patterns originate.

6.1 Introduction

It must be mentioned that, within the entire pattern space V, the class of negative/non-desirable patterns occupies a significantly larger volume in comparison to the volume occupied by the positive/desirable patterns. This is anticipated, as user preferences are concentrated within a small fraction of the universe of discourse. Thus, in the context of building an efficient RS, it is crucial that the RS be able to recognize the majority of instances that belong to the complementary space of non-desirable data. This particular problem is addressed by the first-level component of the cascade classification architecture by developing a one-class classifier trained exclusively on samples from the smaller class of desirable patterns. In other words, the purpose of the first-level classifier is to address the extremely *unbalanced* nature of the machine learning problem that arises when a content-based, item-oriented approach is adopted to address the problem of recommendation. On the other hand, the second-level classifier addresses the problem of discriminating among the various (sub-)classes of desirable patterns. The task of the second-level classifier may be formulated as a *balanced* multi-class machine learning problem, as the users' preferences are (more or less) evenly distributed over the classes of desirable patterns.

In order to formulate the problem of recommendation as a two-level machine learning problem, it is necessary to precisely define the training and testing procedures followed for both classifiers. Clearly, the training procedure of classifiers at

A.S. Lampropoulos and G.A. Tsihrintzis, *Machine Learning Paradigms*,
Intelligent Systems Reference Library 92, DOI 10.1007/978-3-319-19135-5_6

both levels is conducted exclusively on positive data. This is a very important aspect of our approach, as, during the training process, we completely ignore the larger class of non-desirable patterns. Negative (non-desirable) patterns are only used within the testing procedure to accurately measure the efficiency of the two-level classifier in predicting the class of unseen patterns.

6.2 Cascade Content-Based Recommendation

First-Level: One-Class SVM—Second-Level: Multi-Class SVM

Let $U = \{u_1, u_2, \ldots, u_m\}$ and $I = \{i_1, i_2, \ldots, i_n\}$ be the sets of users and items, respectively of the database from which our RS makes recommendations. Each itam in the database corresponds to a feature vector (e.g., a 30-dimensional MARSYAS feature vector) in a high-dimensional Euclidean vector space V. Each user assigns a unique rating value for each item in the database within the range of {0, 1, 2, 3}. Thus, user ratings define four disjoint classes of increasing degree of interest, namely C_0, C_1, C_2 and C_3. C_0 corresponds to the class of non-desirable/negative patterns, while the class of desirable/positive patterns may be defined as the union $(C_1 \cup C_2 \cup C_3)$ of C_1, C_2 and C_3. In order to indicate the user involvement in defining the four classes of interest, we may write that

$$\forall u \in U, \quad V = C_0(u) \cup C_1(u) \cup C_2(u) \cup C_3(u), \tag{6.1}$$

where

$$C_0(u) \cap C_1(u) \cap C_2(u) \cap C_3(u) = \emptyset. \tag{6.2}$$

More specifically, letting $R(u, i)$ be the rating value that the user u assigned to item i, the four classes of interest may be defined via the following equations:

$$\begin{aligned} C_0(u) &= \{i \in I : R(u, i) = 0\} \\ C_1(u) &= \{i \in I : R(u, i) = 1\} \\ C_2(u) &= \{i \in I : R(u, i) = 2\} \\ C_3(u) &= \{i \in I : R(u, i) = 3\} \end{aligned} \tag{6.3}$$

At this point, we need to mention that if $I(u)$ denotes the subset of items for which user u provided a rating, it follows that $\forall u \in U, I(u) = I$. Thus, the positive (desirable) and the negative (non-desirable) classes of patterns for each user may be defined as follows:

$$\begin{aligned} \forall u \in U, \quad \mathbf{P}(u) &= C_1(u) \cup C_2(u) \cup C_3(u) \\ \forall u \in U, \quad \mathbf{N}(u) &= C_0(u) \end{aligned} \tag{6.4}$$

The training/testing procedure for the classifiers at both levels involves partitioning each class of the desirable patterns for each user into K disjoint subsets such that:

$$\forall u \in U, j \in \{1,2,3\}, \quad C_j(u) = \bigcup_{k\in[K]} C_j(u,k), \tag{6.5}$$

where

$$\forall k \in [K], \quad |C_j(u,k)| = \frac{1}{K}|C_j(u)| \text{ such that } \bigcap_{k\in[K]} C_j(u,k) = \emptyset. \tag{6.6}$$

Letting $C_j(u,k)$ be the set of patterns from the positive class j that is used throughout the testing procedure, the corresponding set of training patterns will be denoted as $\widehat{C}_j(u,k)$ and the following equation holds:

$$\forall j \in \{1,2,3\}, \forall k \in [K], \quad \widehat{C}_j(u,k) \cup C_j(u,k) = C_j(u). \tag{6.7}$$

In other words, Eq. 6.7 defines the K-fold cross validation partitioning that is utilized to measure the performance accuracy of our cascade classification scheme. Let $P(u,k)$ and $N(u,k)$ be the sets of positive and negative patterns, respectively, as they are presented to the first-level classifier during the testing stage at fold k for a particular user u. We have:

$$P(u,k) = C_1(u,k) \cup C_2(u,k) \cup C_3(u,k) \tag{6.8}$$

$$N(u,k) = C_0(u,k) = C_0(u) = N(u) \tag{6.9}$$

In case the K-fold cross validation partitioning is not taken into consideration, the set of positive patterns for a particular user may be referred to as $P(u)$, so that $P(u) = C_1(u) \cup C_2(u) \cup C_3(u)$.

The training procedure of the first level of our cascade classification architecture aims at developing one one-class classifier per user. These one-class classifiers are trained to recognize those data instances that have originated from the positive class of patterns. In other words, each one-class classifier realizes a discrimination function denoted by $f_u(\mathbf{v})$, where $\mathbf{v}$ is a vector in V that is learnt from the fraction of training positive patterns. More specifically, if $f_{u,k}(\mathbf{v})$ is the discrimination function that corresponds to user u at fold k, then this function would be the result of training the one-class classifier on $\widehat{C}_1(u,k) \cup \widehat{C}_2(u,k) \cup \widehat{C}_3(u,k)$. The purpose of each discrimination function $f_u(\mathbf{v})$ is to recognize the testing positive patterns $P(u)$ against the complete set of negative patterns $N(u)$.

On the other hand, the training procedure of the second level of our cascade classification architecture aims at developing one multi-class classifier per user. This is achieved by training the multi-class classifier on the same set of positive data, $\widehat{C}_1(u,k) \cup \widehat{C}_2(u,k) \cup \widehat{C}_3(u,k)$, but this time with the purpose to discriminate among the various (sub-)classes of the data pertaining to the set $P(u,k)$. In other words, each second-level classifier realizes a discrimination function denoted by $g_u(\mathbf{v})$, the purpose of which is to partition the space of testing (positive) data $P(u)$ into the 3 corresponding subspaces, $C_1(u)$, $C_2(u)$ and $C_3(u)$, of desirable patterns. To explicitly indicate the discrimination function concerning user u at fold k, we use $g_{u,k}(\mathbf{v})$.

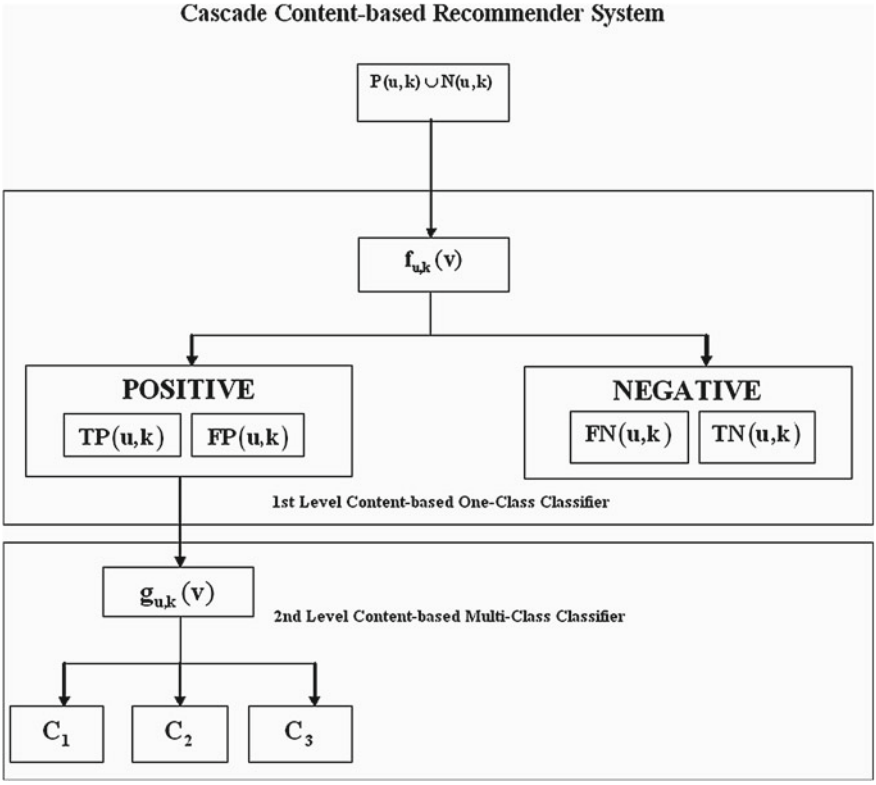


Fig. 6.1 Cascade content-based recommender

The recommendation ability of our system is based on its efficiency when predicting the rating value that a particular user assigned to an item which was not included in the training set. Having in mind that $P(u, k) \cup N(u, k)$ is the full set of testing data presented to the fist level of our cascade classification mechanism, the one-class component operates as a filter that recognizes the items that a user assigned to the class of desirable patterns. Specifically, the first-level discrimination function $f_{u,k}(\mathbf{v})$ for user u at fold k partitions the set of testing data into positive and negative patterns as illustrated in Fig. 6.1. In other words, the testing procedure concerning the first-level of our cascade classifier involves the assignment of a unique value within the set $\{-1, 1\}$ for each input element such that:

$$\forall u \in U,\ \forall k \in [K],\ \forall \mathbf{v} \in P(u, k) \cup N(u, k),\quad f_{u,k}(\mathbf{v}) \in \{-1, +1\}. \tag{6.10}$$

The subset of testing instances that are assigned to the class of desirable patterns are subsequently fed into the second-level classifier which assigns to them a particular rating value within the range of $\{1, 2, 3\}$. Specifically,

$$\forall u \in U, \forall k \in [K], \forall \mathbf{v} \in P(u,k) \cup N(u,k) : f_{u,k}(\mathbf{v}) = +1,\ g_{u,k}(\mathbf{v}) \in \{1,2,3\} \tag{6.11}$$

Let the true rating value concerning an object $\mathbf{v} \in V$ for a particular user u at fold k be $R_{u,k}(\mathbf{v})$ so that the following equation holds:

$$\forall u \in U,\ \forall k \in [K],\ \forall j \in \{0,1,2,3\}, \quad R_{u,k}(\mathbf{v}) = j \Leftrightarrow \mathbf{v} \in C_j(u,k). \tag{6.12}$$

The estimated rating value assigned by our system will be indicated as $\widehat{R}_{u,k}(\mathbf{v})$ and can be computed as in the following equation:

$$\widehat{R}_{u,k}(\mathbf{v}) = \begin{cases} 0,\ \forall \mathbf{v} \in P(u,k) \cup N(u,k) : f_{u,k}(\mathbf{v}) = -1; \\ 1,\ \forall \mathbf{v} \in P(u,k) \cup N(u,k) : f_{u,k}(\mathbf{v}) = +1 \text{ and } g_{u,k}(\mathbf{v}) = 1 \\ 2,\ \forall \mathbf{v} \in P(u,k) \cup N(u,k) : f_{u,k}(\mathbf{v}) = +1 \text{ and } g_{u,k}(\mathbf{v}) = 2 \\ 3,\ \forall \mathbf{v} \in P(u,k) \cup N(u,k) : f_{u,k}(\mathbf{v}) = +1 \text{ and } g_{u,k}(\mathbf{v}) = 3. \end{cases} \tag{6.13}$$

6.3 Cascade Hybrid Recommendation

First-Level: One-Class SVM—Second-Level: Collaborative Filtering

As stated previously, the problem addressed in this chapter is that of building an efficient RS in the complete absence of negative examples. Since negative examples are, in general, extremely difficult to obtain, we employed classification paradigms that operate exclusively on the basis of positive patterns. This justifies the incorporation of the one-class classification component within the first-level of our cascade classification architecture. Thus, the first classification level serves the purpose of filtering out the majority of the non-desirable patterns. However, as the ultimate purpose of any RS is to provide its users with high quality recommendations, a component is needed which predicts the true class of unseen patterns with high accuracy. This is the rationale behind the second classification level which takes as input the set of patterns that were assigned to the class of desirable items by the first classification level. In order to provide high quality recommendations, it is vital that the second-level classifier correctly discriminate among the various (sub-)classes of desirable patterns. Thus, the second-level classifier is a multi-class one.

A natural modification of our cascade classification architecture consists of replacing the second (multi-class) classification level with a CF component, as illustrated in Fig. 6.2. Having in mind that the first classification level realizes the broader distinction between positive and negative patterns, the subsequent CF component produces specific rating values within the range of {1, 2, 3}. Specifically, the CF methods that we utilized were:

- Pearson Correlation [1]
- Vector Similarity [1] and
- Personality Diagnosis [2].

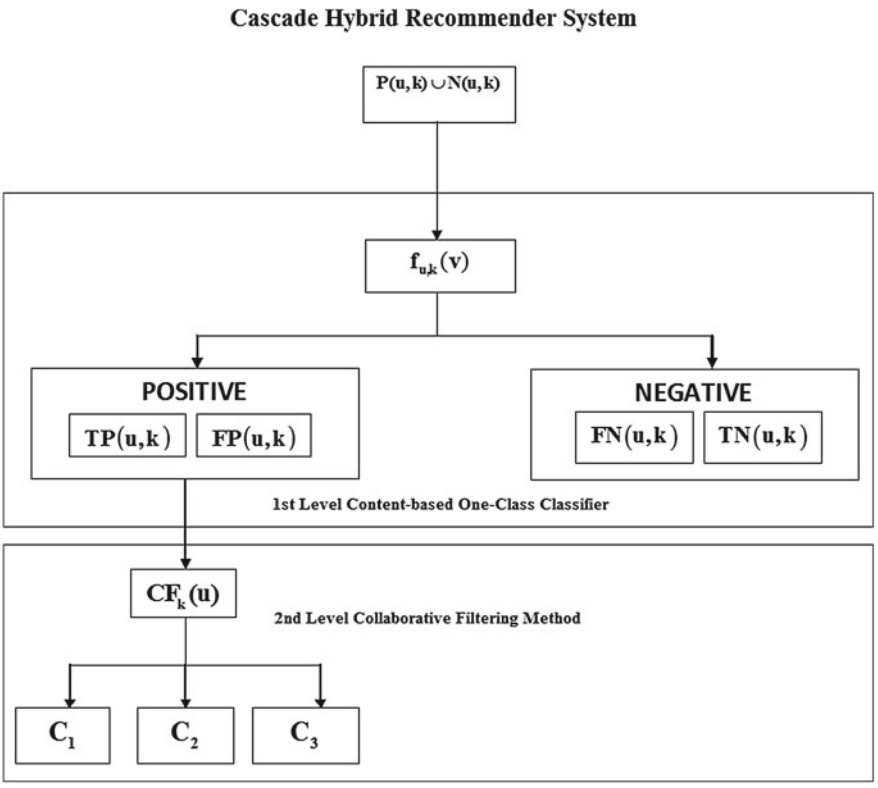


Fig. 6.2 Cascade hybrid recommender

Personality Diagnosis (PD) may be thought of as a hybrid between memory and model-based approaches of CF. Its main characteristic is that predictions have meaningful probabilistic semantics. Moreover, this approach assumes that preferences constitute a characterization of their underlying personality type for each user. Therefore, taking into consideration the active user's known ratings of items, it is possible to estimate the probability that he/she has the same personality type with another user. The personality type of a given user is taken to be the vector of "true" ratings for items the user has seen. A true rating differs by an amount of Gaussian noise from the actual rating given by a user. Given the personality type of a user A, PD finds the probability that the given user is of the same personality type as other users in the system and, subsequently, the probability that the user will like some new item [2].

The training and testing procedures concerning the second-level CF component are identical to the ones used for the multi-class classification component. Specifically, training was conducted on the ratings that correspond to the set of patterns $\widehat{C}_1(u,k) \cup \widehat{C}_2(u,k) \cup \widehat{C}_3(u,k)$. Accordingly, the testing procedure was conducted on the rating values that correspond to the set of testing patterns $\cup C_1(u,k) \cup C_2(u,k) \cup C_3(u,k)$.

6.4 Measuring the Efficiency of the Cascade Classification Scheme

The efficiency of the adapted cascade classification scheme was measured in terms of the *Mean Absolute Error* and the *Ranked Scoring* measures. The Mean Absolute Error (MAE) constitutes the most commonly measure used to evaluate the efficiency of RS. More formally, the MAE concerning user u at fold k may be defined as:

$$MAE(u,k) = \frac{1}{|P(u,k)| + |N(u,k)|} \sum_{\mathbf{v} \in P(u,k) \cup N(u,k)} |R_{u,k}(\mathbf{v}) - \widehat{R}_{u,k}(\mathbf{v})| \tag{6.14}$$

The Ranked Scoring (RSC) [1] assumes that the recommendation is presented to the user as a list of items ranked by their predicted ratings. Specifically, RSC assesses the expected utility of a ranked list of items by multiplying the utility of an item for the user by the probability that the item will be viewed by the user. The utility of an item is computed as the difference between its observed rating and the default or neutral rating d in the domain, which can be either the midpoint of the rating scale or the average rating in the dataset. On the other hand, the probability of viewing decays exponentially as the rank of items increases. Formally, the RSC of a ranked list of items $\mathbf{v_j} \in P(u_i,k) \cup N(u_i,k)$ sorted according to the index j in order of declining $R_{u_i,k}(\mathbf{v_j})$ for a particular user u_i at fold k is given by:

$$RS_{u_i,k} = \sum_{\mathbf{v_j} \in P(u_i,k) \cup N(u_i,k)} \max\{R_{u_i,k}(\mathbf{v_j}) - d), 0\} \times \frac{1}{2^{(j-1)(i-1)}} \tag{6.15}$$

Having in mind that the set of testing patterns for the first-level classifier at fold k is formed by the patterns pertaining to the sets $C_0(u)$, $C_1(u,k)$, $C_2(u,k)$ and $C_3(u,k)$, we may write that

$$|P(u,k)| = |C_1(u,k)| + |C_2(u,k)| + |C_3(u,k)| \tag{6.16}$$

and

$$|N(u,k)| = |C_0(u)|. \tag{6.17}$$

According to Eqs. 6.16 and 6.17, we may define the *true positive rate* (TPR), *false negative rate* (FNR), *true negative rate* (TNR), and *false positive rate* (FPR) concerning user u for the kth fold of the testing stage as follows:

$$TPR(u,k) = \frac{TP(u,k)}{|P(u,k)|} \tag{6.18}$$

$$FNR(u,k) = \frac{FP(u,k)}{|P(u,k)|} \tag{6.19}$$

$$TNR(u,k) = \frac{TN(u,k)}{|N(u,k)|} \tag{6.20}$$

and

$$FPR(u,k) = \frac{FP(u,k)}{|N(u,k)|} \tag{6.21}$$

It is important to note that the quantities defined in Eqs. 6.18, 6.19, 6.20, and 6.21 refer to the classification performance of the first-level classifier in the adapted cascade classification scheme. More specifically, True Positive, $TP(u,k)$, is the number of positive/desirable patterns that were correctly assigned to the positive class of patterns while False Negative, $TN(u,k)$, is the number of positive/desirable patterns that were incorrectly assigned to the negative class of patterns. Similarly, True Negative, $TN(u,k)$, is the number of negative/non-desirable patterns that were correctly assigned to the negative class of patterns, while False Positive, $FP(u,k)$, is the number of negative/non-desirable patterns that were incorrectly assigned to the positive class. More formally, having in mind Eq. 6.13, the above quantities may be described as follows:

$$TP(u,k) = \{\mathbf{v} \in P(u,k) : f_{u,k}(\mathbf{v}) = +1\} \tag{6.22}$$

$$FP(u,k) = \{\mathbf{v} \in N(u,k) : f_{u,k}(\mathbf{v}) = +1\} \tag{6.23}$$

$$TN(u,k) = \{\mathbf{v} \in N(u,k) : f_{u,k}(\mathbf{v}) = -1\} \tag{6.24}$$

$$FN(u,k) = \{\mathbf{v} \in P(u,k) : f_{u,k}(\mathbf{v}) = -1\} \tag{6.25}$$

Computing the mean value for the above quantities over different folds results in the following equations:

$$\overline{TPR}(u) = \frac{1}{K}\sum_{f \in F} TPR(u,k) \tag{6.26}$$

$$\overline{FNR}(u) = \frac{1}{K}\sum_{f \in F} FNR(u,k) \tag{6.27}$$

$$\overline{TNR}(u) = \frac{1}{K}\sum_{f \in F} TNR(u,k) \tag{6.28}$$

$$\overline{FPR}(u) = \frac{1}{K}\sum_{f \in F} FPR(u,k) \tag{6.29}$$

It is possible to bound the MAE for the complete two-level classifier according to its performance during the second stage of the multi-class classification scheme.

The best case scenario concerning the classification performance of the second-level (multi-class) classifier suggests that all the true positive patterns, which are passed to the second classification level, are correctly classified. Moreover, the best case scenario requires that all the false negative patterns of the first classification level originated from C_1. Thus, the following inequality holds:

$$\forall u \in U \; \forall k \in [K] \; MAE(u,k) \geq \frac{FN(u,k) + FP(u,k)}{|P(u,k)| + |N(u,k)|} \tag{6.30}$$

Given Eqs. 6.18, 6.19, 6.20, and 6.21 and letting

$$\lambda(u,k) = \frac{|P(u,k)|}{|N(u,k)|} = \frac{|P(u)|}{|N(u)|} = \lambda(u) \tag{6.31}$$

as the numbers of positive and negative patterns used during the testing stage do not change for each fold and for each user, inequality 6.30 may be written as

$$MAE(u,k) \geq FNR(u,k) \times \frac{\lambda(u)}{\lambda(u)+1} + FPR(u,k) \times \frac{1}{\lambda(u)+1}. \tag{6.32}$$

Given that

$$\overline{MAE}(u) = \frac{1}{K} \sum_{k \in [K]} MAE(u,k), \tag{6.33}$$

inequality 6.32 may be written as:

$$\overline{MAE}(u) \geq \overline{FNR}(u) \times \frac{\lambda(u)}{\lambda(u)+1} + \overline{FPR}(u) \times \frac{1}{\lambda(u)+1}. \tag{6.34}$$

If we consider the average value for the MAE over all users, we may write that:

$$\overline{MAE} = \frac{1}{|U|} \sum_{u \in U} \overline{MAE}(u). \tag{6.35}$$

This results in:

$$\overline{MAE} \geq \frac{1}{|U|} \sum_{u \in U} \overline{FNR}(u) \times \frac{\lambda(u)}{\lambda(u)+1} + \overline{FPR}(u) \times \frac{1}{\lambda(u)+1} \tag{6.36}$$

The worst case scenario concerning the classification performance of the second level is that the second-level (multi-class) classifier incorrectly assigns all true positive patterns to C_3 while they truly originated from C_1. In addition, all the false negative patterns originate from C_3 and all the false positive patterns are assigned to C_3. Thus, we may write the following inequality:

$$\forall u \in U \; \forall f \in [K] \; MAE(u,k) \leq \frac{3 \times FN(u,k) + 2 \times TP(u,k) + 3 \times FP(u,k)}{P(u,k) + N(u,k)} \tag{6.37}$$

Given Eqs. 6.18, 6.19, 6.20, 6.21 and 6.31, Eq. 6.37 may be written as:

$$MAE(u,k) \leq \frac{3 \times FNR(u,k) \times \lambda(u)}{\lambda(u)+1} + \frac{2 \times TPR(u,k) \times \lambda(u)}{\lambda(u)+1} + \frac{3 \times FPR(u,k)}{\lambda(u)+1} \tag{6.38}$$

Now, given Eq. 6.33, inequality 6.38 results in:

$$\overline{MAE(u)} \leq \frac{3 \times \overline{FNR}(u) \times \lambda(u)}{\lambda(u)+1} + \frac{2 \times \overline{TPR}(u) \times \lambda(u)}{\lambda(u)+1} + \frac{3 \times \overline{FPR}(u)}{\lambda(u)+1} \tag{6.39}$$

Thus, the average value for the MAE has an upper bound given by the following inequality:

$$\overline{MAE} \leq \frac{1}{|U|} \sum_{u \in U} \frac{3 \times \overline{FNR}(u) \times \lambda(u)}{\lambda(u)+1} + \frac{2 \times \overline{TPR}(u) \times \lambda(u)}{\lambda(u)+1} + \frac{3 \times \overline{FPR}(u)}{\lambda(u)+1}. \tag{6.40}$$

Inequalities 6.36 and 6.40 imply that the minimum value for the average MAE over all users is given as:

$$\min_{u \in U} \overline{MAE} = \frac{1}{|U|} \sum_{u \in U} \frac{\overline{FNR}(u) \times \lambda(u)}{\lambda(u)+1} + \frac{\overline{FPR}(u)}{\lambda(u)+1}. \tag{6.41}$$

Similarly, the maximum value for the average MAE over all users is given as:

$$\max_{u \in U} \overline{MAE} = \frac{1}{|U|} \sum_{u \in U} \frac{3 \times \overline{FNR}(u) \times \lambda(u)}{\lambda(u)+1} + \frac{2 \times \overline{TPR}(u) \times \lambda(u)}{\lambda(u)+1} + \frac{1}{|U|} \sum_{u \in U} \frac{3 \times \overline{FPR}(u)}{\lambda(u)+1}. \tag{6.42}$$

References

1. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 43–52. Morgan Kaufmann (1998)
2. Pennock, D.M., Horvitz, E., Lawrence, S., Giles, C.L.: Collaborative filtering by personality diagnosis: a hybrid memory and model-based approach. In: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, UAI'00, pp. 473–480. Morgan Kaufmann Publishers Inc., San Francisco (2000)

Chapter 7
Evaluation of Cascade Recommendation Methods

Abstract The experimental results provided in this chapter correspond to the testing stage of our system. The evaluation process compared three recommendation approaches: (a) the standard Collaborative Filtering methodologies, (b) the Cascade Content-based Recommendation methodology and (c) the Cascade Hybrid Recommendation methodology. To evaluate our system, we tested its performance as a RS for music files. In the following sections of this chapter, a detailed description is provided of the three types of experiments that were conducted in order to evaluate the efficiency of our cascade recommendation architecture.

7.1 Introduction

The evaluation process involved three recommendation approaches:

1. The first approach corresponds to the standard collaborative filtering methodologies, namely the Pearson Correlation, the Vector Similarity and the Personality Diagnosis.
2. The second approach corresponds to the Cascade Content-based Recommendation methodology which was realized on the basis of a two-level classification scheme. Specifically, we tested one-class SVM for the first level, while the second classification level was realized as a multi-class SVM.
3. Finally, the third approach corresponds to the Cascade Hybrid Recommendation methodology which was implemented by a one-class SVM classification component at the first level and a CF counterpart at the second level. Specifically, the third recommendation approach involves three different recommenders which correspond to the different CF methodologies that were embedded within the second level.

Three types of experiments were conducted in order to evaluate the efficiency of our cascade recommendation architecture.

- The first type of experiments is described in Sect. 7.2 and demonstrates the contribution of the one-class classification component at the first level of our cascade recommendation system. Specifically, we provide MAE and RSC measurements

A.S. Lampropoulos and G.A. Tsihrintzis, *Machine Learning Paradigms*,
Intelligent Systems Reference Library 92, DOI 10.1007/978-3-319-19135-5_7

concerning the mean overall performance of the standard collaborative filtering methodologies in the hybrid recommendation approach for the complete set of users. Additionally, we measure the relative performance of the Cascade Content-based Recommender against the performance of other recommendation approaches in order to identify the recommendation system that exhibits the best overall performance.

- The second type of experiments is described in Sect. 7.3 and demonstrates the contribution of the second (multi-class) classification level within the framework of the Cascade Content-based Recommendation methodology. The main purpose of this experimentation session is to reveal the benefit in recommendation quality obtained via the second (multi-class) classification level.

7.2 Comparative Study of Recommendation Methods

In this section, we provide a detailed description concerning the first type of experiments. Our primary concern focused on conducting a comparative study of the various recommendation approaches that were implemented. It is very important to assess the recommendation ability of each individual system in order to identify the one that exhibited the best overall performance. Specifically, the recommendation accuracy was measured in terms of the average MAE over all folds for the complete set of users. Our findings indicate that there was no recommendation approach that outperformed the other approaches for the complete set of users. This means that there were occasions for which the best recommendations for a particular user were given by the standard CF approach. On the other hand, there were occasions for which either the Cascade Content-based Recommender or the Cascade Hybrid Recommender provided more accurate predictions concerning the true user ratings.

Typical examples of the previously mentioned situations are illustrated in Figs. 7.1, 7.2 and 7.3. Specifically, Fig. 7.1 demonstrates that the best recommendation approach for User1 was the Cascade Content-based Recommender. In order of decreasing efficiency, the other recommendation approaches for User1 were the Cascade Hybrid Recommender and standard CF. Furthermore, Fig. 7.2 demonstrates that the best recommendation approach for User13 was the standard CF. The remaining recommendation approaches for this user were the Cascade Hybrid Recommender, which ranked second, and the Cascade Content-based Recommender, which ranked third. Finally, Fig. 7.3 demonstrates that the Cascade Content-based Recommender and the standard CF rank second and third, respectively, in terms of efficiency.

The most important finding which results from the first set of experiments is that the overall best recommendation approach over all users and folds was provided by the Cascade Hybrid Recommender. This fact is explicitly illustrated in Fig. 7.4 in which the hybrid approach presents the lowest average MAE taken over all users and folds during the testing stage. It is worth mentioning that the pure content-based and CF methodologies rank second and third, respectively, in terms of the overall recommendation accuracy. This is not an accidental fact, but is rather an immediate

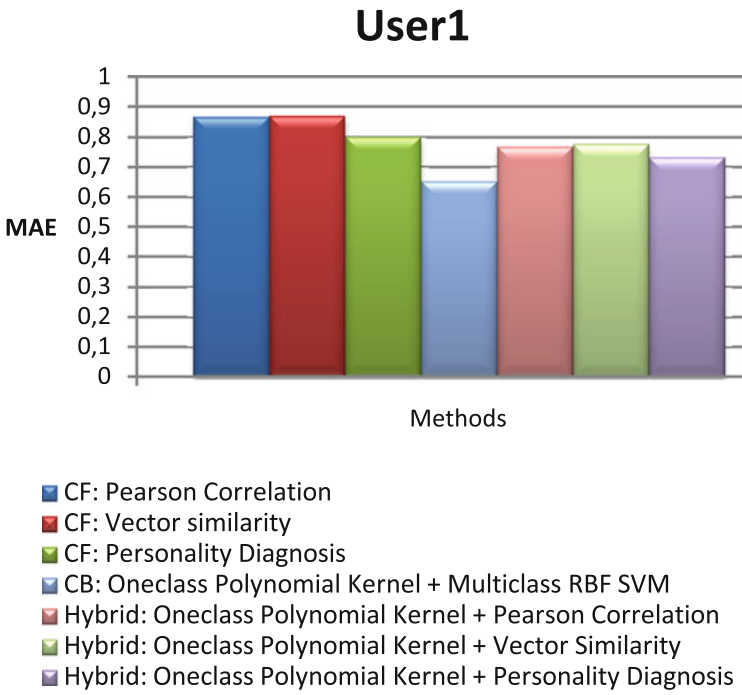


Fig. 7.1 Content-based Recommender is the best for user 1

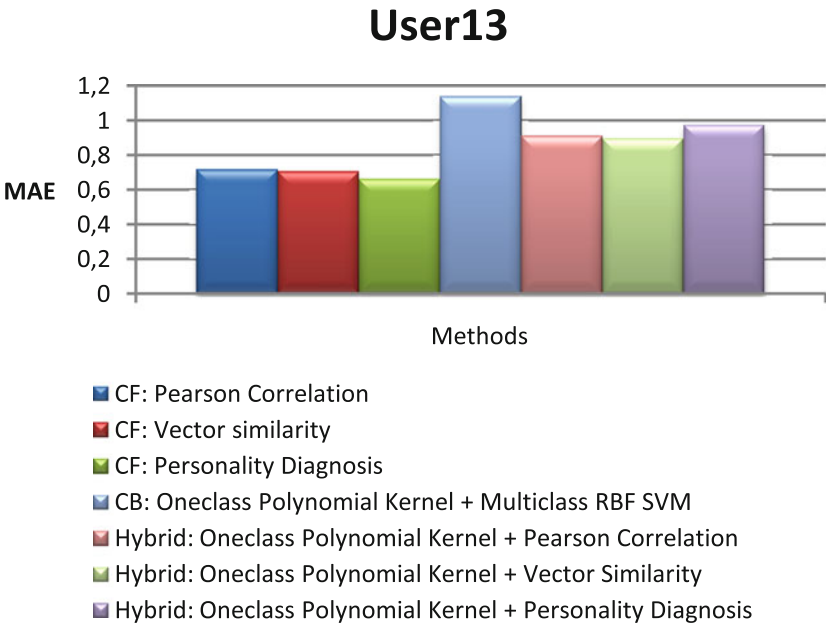


Fig. 7.2 Collaborative filtering Recommender is the best for user 13

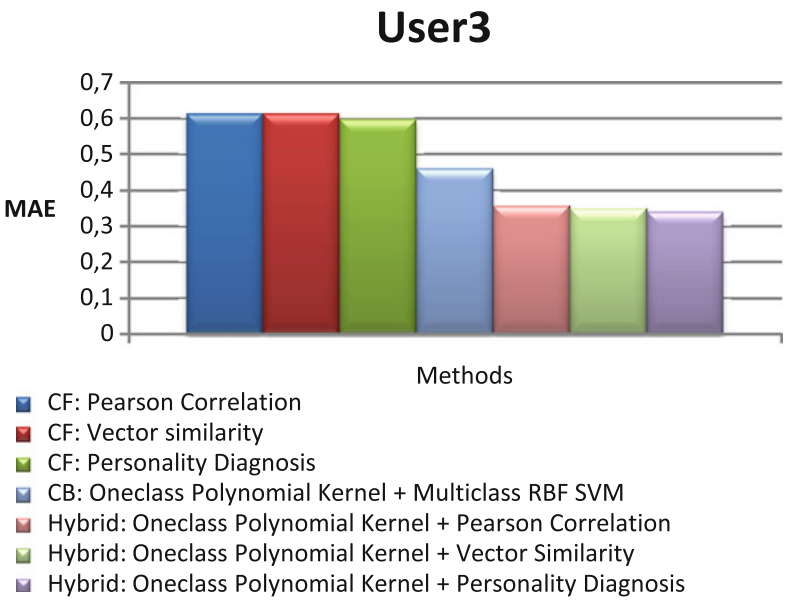


Fig. 7.3 Hybrid Recommender is the best for user3

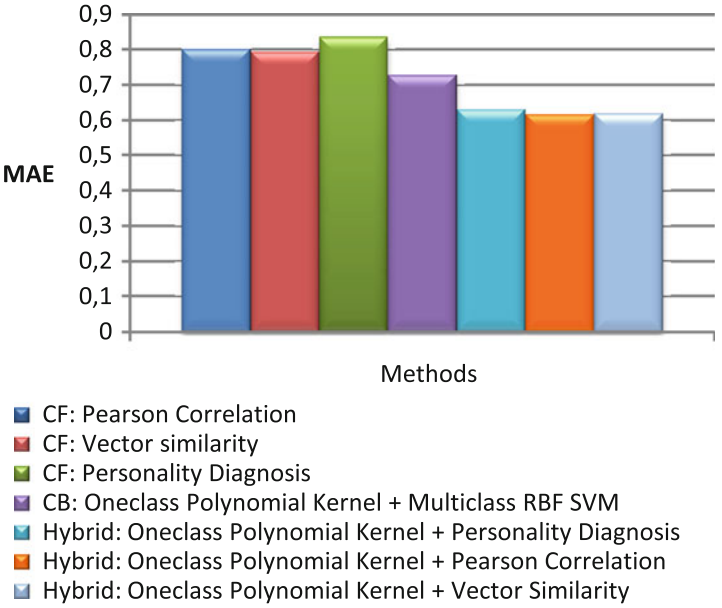


Fig. 7.4 MAE (mean for all users)

consequence of the incorporation of the one-class classification component at the first level of the cascade recommendation scheme.

The recommendation approaches that rely exclusively on CF estimate the rating value that a particular user would assign to an unseen item on the basis of the ratings that the other users have provided for the given item. In other words, the pure CF approaches do not take into account the subjective preferences of an individual user, as they are biased towards the items that are most preferred by the other users. The major drawback of the standard CF approaches is that they disorientate the user by operating exclusively on a basis formed by the preferences of the other users, ignoring the particular preferences an individual user might have.

On the other hand, the pure content-based recommendation approaches fail to exploit neighborhood information for a particular user. They operate exclusively on classifiers which are trained to be user-specific, ignoring any beneficial information related to users with similar preferences. A natural solution to the problems related to the CF and content-based recommendation approaches would be the formation of a hybrid RS. Such a system would incorporate the classification power of the content-based recommenders and the ability of standard CF approaches to estimate user ratings on the basis of similar users' profiles.

The Cascade Hybrid Recommendation approach presented in here mimics the social process in which someone has selected items according to his/her preferences and seeks other people's opinions about these, in order to make a better selection. In other words, the one-class classification component, at the first level, provides specialized recommendations by filtering out those items that a particular user would characterize as non-desirable. This is achieved through the user-specific training process of the one-class classifiers which are explicitly trained on user-defined positive classes of patterns. On the other hand, the second level of recommendation exploits the neighborhood of preferences formed by users with similar opinions. The recommendation superiority exhibited by the Cascade Hybrid Recommender is based on the more efficient utilization of its CF component. This is achieved by constraining its operation only on the subset of patterns that are already recognized as desirable. Therefore, this approach resolves the problem of user disorientation by asking for the opinions of other users only for the items that a particular user assigns to the positive class of patterns.

7.3 One-Class SVM—Fraction: Analysis

The purpose of this set of experiments is to reveal the contribution of the second (multi-class) classification level in the overall recommendation ability of the Cascade Content-based Recommender. Equations 6.41 and 6.42 provide the minimum and maximum values for the average MAE over all users, given the classification performance of the first (one-class) classification level. Having in mind that these lower and upper bounds on the average MAE concern the overall performance of the cascade recommender at both levels, they reflect the impact of the second (multi-class)

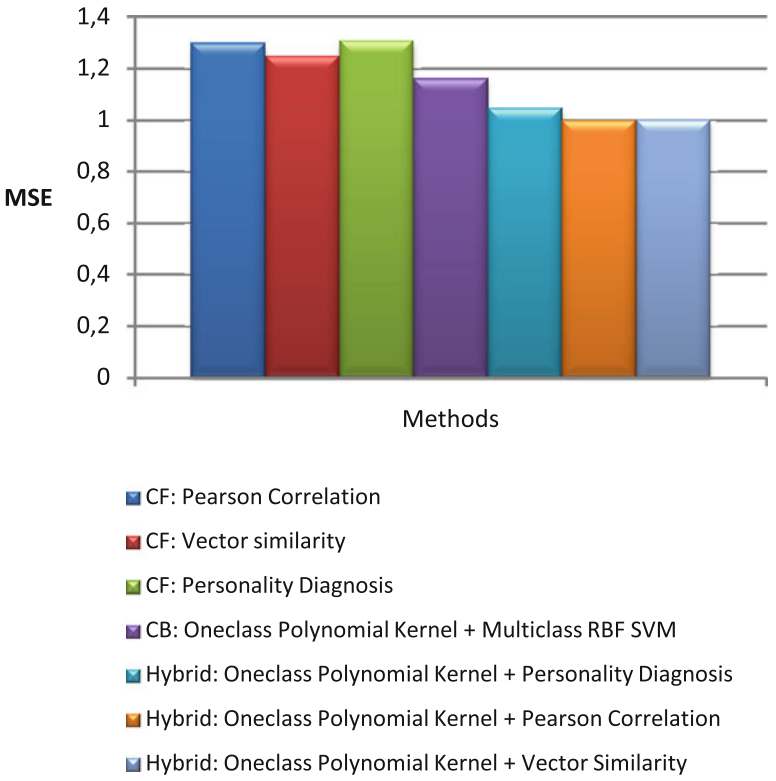


Fig. 7.5 MSE (mean for all users)

classification component. The lower bound on the average MAE corresponds to the best case scenario in which the second (multi-class) classification level performs inerrably. On the other hand, the upper bound on the average MAE corresponds to the worst case scenario, in which the second (multi-class) classification level fails completely. In this context, if we measure the actual value of the average MAE over all users, we can assess the influence of the second classification level on the overall recommendation accuracy of our system. Thus, if the actual value of the average MAE is close to the lower bound, this implies that the second classification level operated close to the highest possible performance level. On the other hand, if the actual value of the average MAE is closer to its upper bound, this implies that the second classification level did not contribute significantly to the overall performance of our recommender (Fig. 7.5).

Figure 7.7 shows the actual average MAE relative to its corresponding lower and upper bound curves. Each curve is generated by parameterizing the one-class SVM classifier with respect to the fraction of the positive data that should be rejected during the training process.

The relative performance of one-class SVM-based classifier was measured in terms of precision, recall, F1-measure and MAE, which are defined in the following.

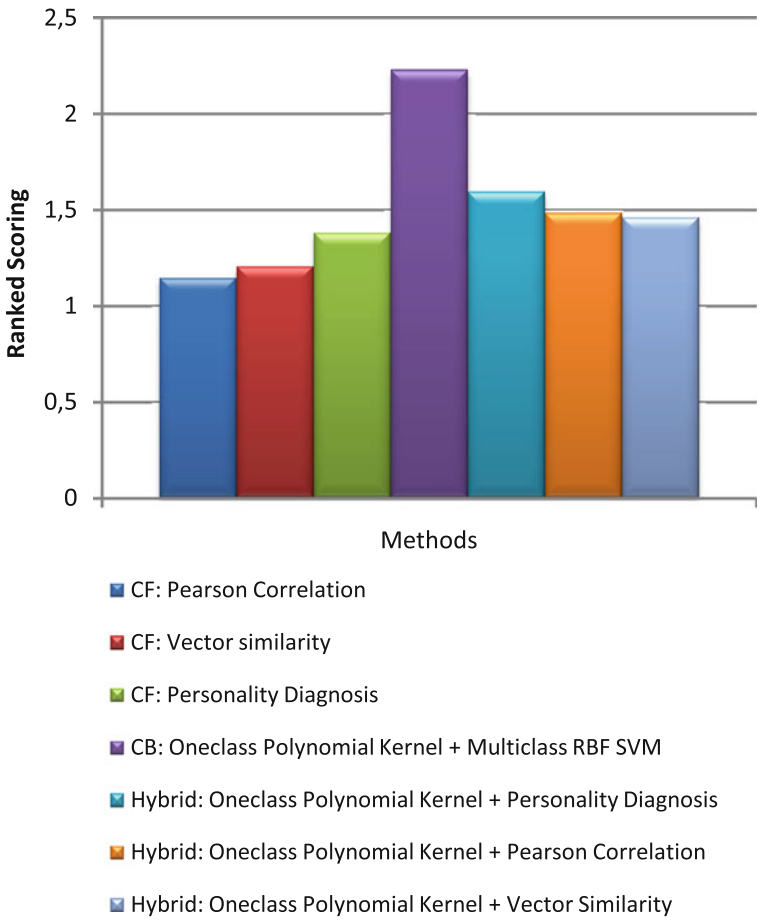


Fig. 7.6 Ranked Scoring (mean for all users)

The *precision* is defined as an average over all users and folds in relation to the average values for the true positives and the false positives:

$$\overline{Precision} = \frac{\overline{TP}}{\overline{TP} + \overline{FP}}. \tag{7.1}$$

On the other hand, the *recall* is defined as the average over all users and folds in relation to the average values for the true positives and the false negatives:

$$\overline{Recall} = \frac{\overline{TP}}{\overline{TP} + \overline{FN}}. \tag{7.2}$$

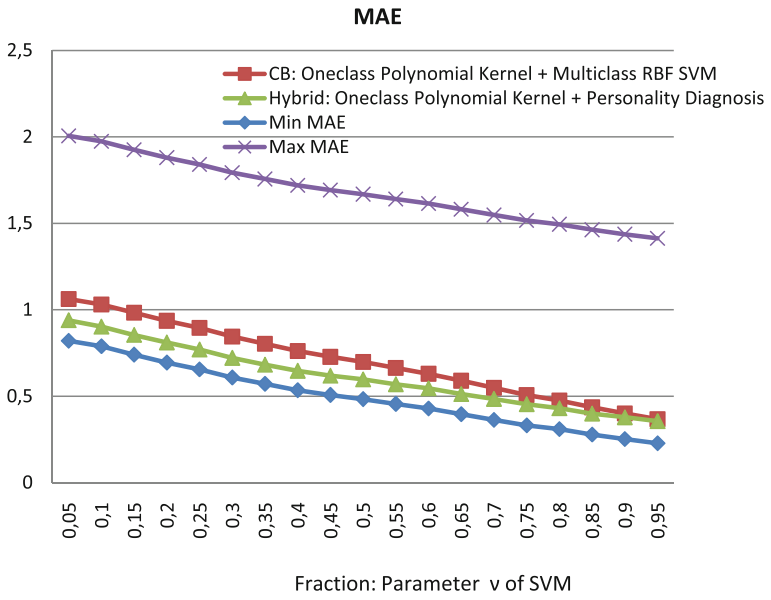


Fig. 7.7 MAE Boundaries for one-class SVM

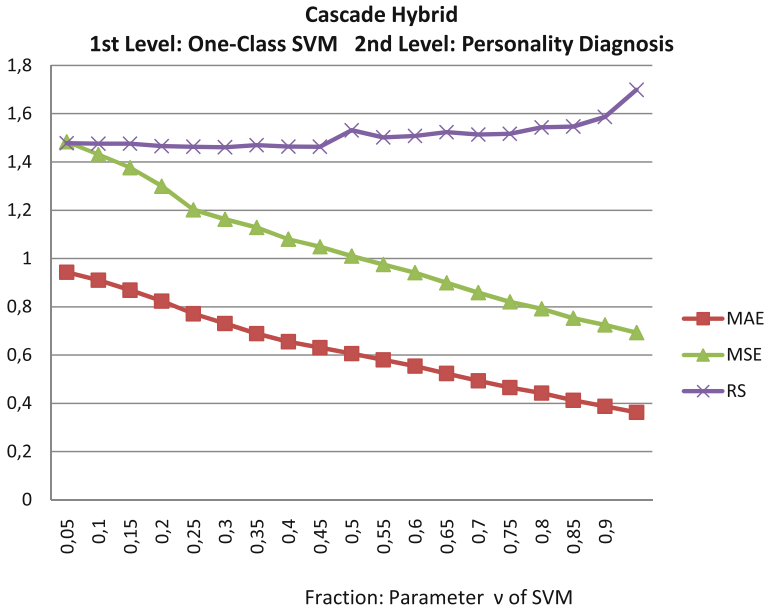


Fig. 7.8 Hybrid Recommender 2nd level personality diagnosis: Fraction analysis

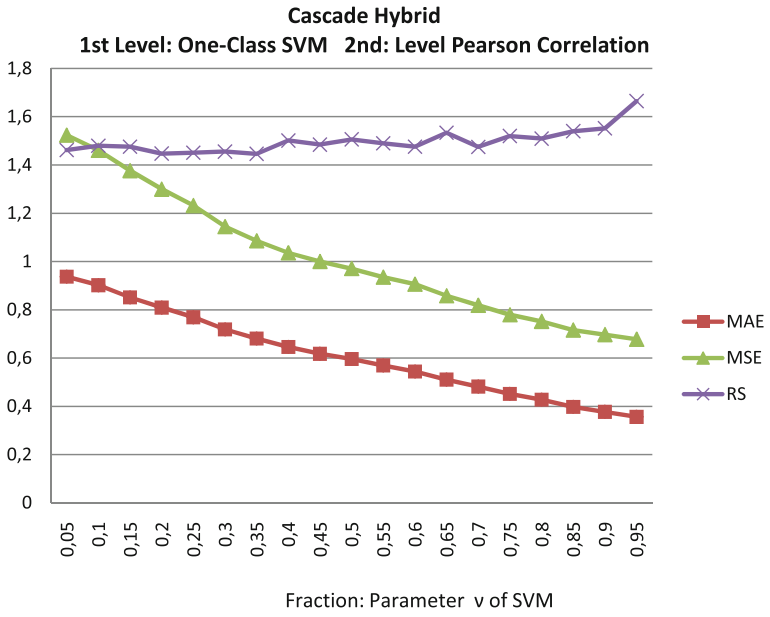


Fig. 7.9 Hybrid Recommender 2nd level Pearson correlation: Fraction analysis

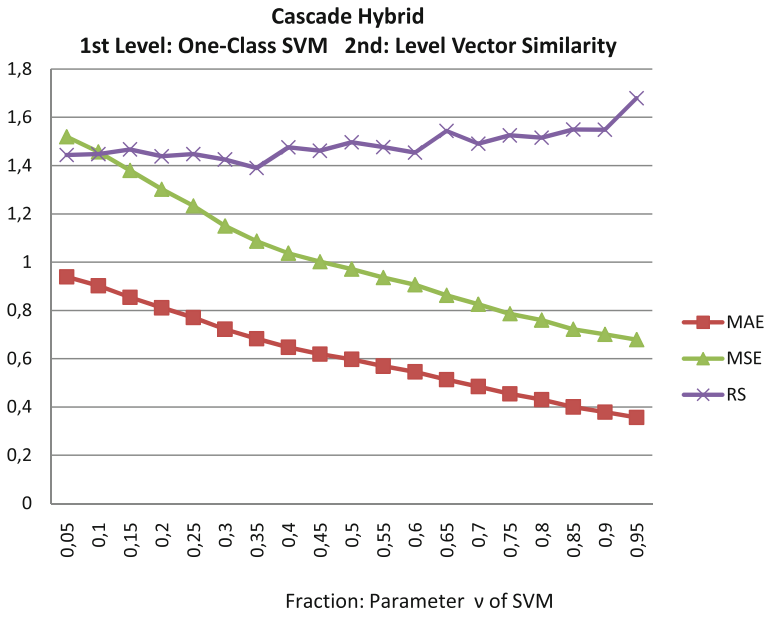


Fig. 7.10 Hybrid Recommender 2nd level vector similarity: Fraction analysis

Finally, the *F1-measure* is defined as the average value for the F1-measure over all users and folds.

$$\overline{F1} = \frac{2 \times \overline{Precision} \times \overline{Recall}}{\overline{Precision} + \overline{Recall}} \tag{7.3}$$

The precision quantifies the amount of *information* that is not lost, while the recall expresses the amount of *data* that is not lost. Higher precision and recall values indicate superior classification performance. The F1-measure is a combination of precision and recall which ranges within the [0, 1] interval. The minimum value (0) indicates the worst possible performance, while the maximum value (1) indicates the highest possible performance.

The MAE is a measure related to the overall classification performance of the Cascade Recommender. MAE values closer to zero indicate higher recommendation accuracy. It is very important to note that in the context of the highly unbalanced classification problem related to recommendation, the quality that dominates the level of the MAE is the number of the correctly classified negative patterns, i.e. the true negatives. Since the vast majority of patterns belong to the negative class, correctly identifying them reduces the overall classification error. Thus, a lower MAE value for the one-class SVM classifier indicates that this classifier performs better in filtering out non-desirable patterns. On the other hand, the F1-measure, that specifically relates to precision and recall according to Eq. 7.3, is dominated by the amount of positive patterns that are correctly classified (i.e., true positives), according to Eqs. 7.1 and 7.2. The F1-measure quantifies the amount of true (thus, useful) positive recommendations that the system provides to the user.

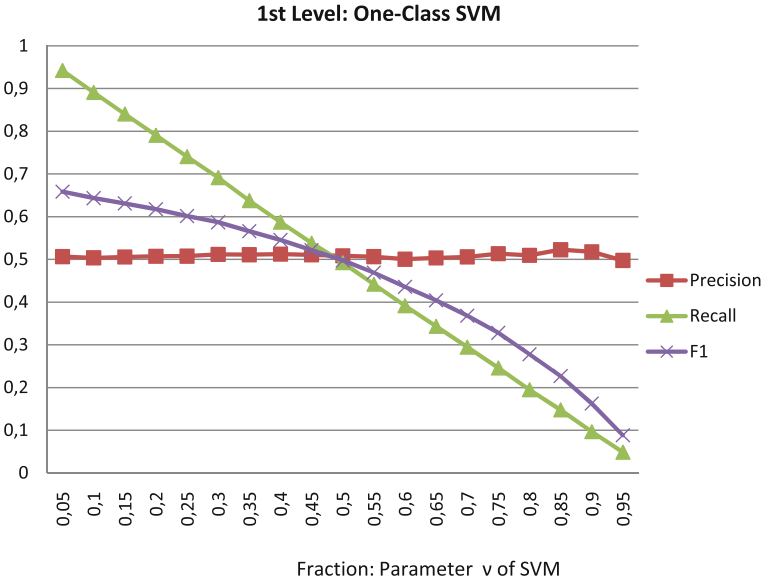


Fig. 7.11 One class SVM (precision, recall, F1)

The previous findings are characteristic of the behavior of the one-class classifiers with respect to the fraction of positive and negative patterns that they identify during their testing process. Our experiments indicate the following:

- The precision performance of the one-class SVM classifier involves increasing true negative rates as the fraction of positive patterns rejected during training approaches 95 %.
- On the other hand, the recall performance of the one-class SVM classifier involves increasing true positive rates as the fraction of positive patterns rejected during training approaches 5 %.

An efficient one-class classifier attempts to achieve one of two goals: (1) to minimize the fraction of false positives and (2) to minimize the fraction of false negatives. Thus, it is a matter of choice whether the recommendation process will focus on increasing the true positive rate or increasing the true negative rate. Increasing the true negative rate results in lower MAE levels, while increasing the true positive rate results in higher F1-measure levels. Specifically, the fact that the non-desirable patterns are significantly higher in number than the desirable ones, suggests that the quality of recommendation is crucially influenced by the number of the correctly identified negative patterns. In other words, constraining the amount of the false positive patterns that pass to the second level of the RS increases the reliability (quality) of the recommended items. The most appropriate measure to describe the quality of recommendation is given by the RSC, as the RSC illustrates the amount of true positive items that are placed at the top of the ranked list. This fact is clearly demonstrated in Fig. 7.6, where the RSC for the Cascade Content-based RS of the one-class SVM classifier outperforms the other recommendation approaches (Figs. 7.7, 7.8, 7.9, 7.10 and 7.11).

Chapter 8
Conclusions and Future Work

Abstract Recommender Systems (RS) attempt to provide information in a way that will be most appropriate and valuable to its users and prevent them from being overwhelmed by huge amounts of information that, in the absence of RS, they should browse or examine. In this book, we presented a number of innovative RS, which are summarized in this chapter. Conclusions are drawn and avenues of future research are identified.

8.1 Summary and Conclusions

Recent advances in electronic media and computer networks have allowed the creation of large and distributed repositories of information. However, the immediate availability of extensive resources for use by broad classes of computer users gives rise to new challenges in everyday life. These challenges arise from the fact that users cannot exploit available resources effectively when the amount of information requires prohibitively long user time spent on acquaintance with and comprehension of the information content. The risk of information overload of users imposes new requirements on the software systems that handle the information.

In this book, firstly, we explored the use of objective content-based features to model the individualized (subjective) perception of similarity between multimedia data. We present a content-based RS which constructs music similarity perception models of its users by associating different similarity measures to different users. The results of the evaluation of the system verified the relation between subsets of objective features and individualized (music) similarity perception and exhibits significant improvement in individualized perceived similarity in subsequent recommended items. The investigation of these relations between objective feature subsets and user perception offer an indirect explanation and justification for the items one selects. The users are clustered according to specific subsets of features that reflect different aspects of the music signal. This assignment of a user to a specific subset of features allows us to formulate indirect relations between his/her perception and corresponding item similarity (e.g. music similarity) that involves his/her preferences. Consequently, the selection of a specific feature subset can provide a justification-

A.S. Lampropoulos and G.A. Tsihrintzis, *Machine Learning Paradigms*,
Intelligent Systems Reference Library 92, DOI 10.1007/978-3-319-19135-5_8

reasoning of the various factors that influence the user's perception of similarity to his/her preferences.

Secondly, we addressed the recommendation process as a hybrid combination of one-class classification with CF. Specifically, we followed a cascade scheme in which the recommendation process is decomposed into two levels. In the first level, our approach attempts to identify for each user only the desirable items from the large amount of all possible items, taking into account only a small portion of his/her available preferences. Towards this goal we apply a one-class classification scheme, in the training stage of which only positives examples (desirable items for which users have express an opinion-rating value) are required. This is very important, as it is sensibly hard in terms of time and effort for users to explicitly express what they consider as non-desirable to them. In the second level, either a content-based or a CF approach is applied to assign a corresponding rating degree to these items. Our cascade scheme first builds a user profile by taking into consideration a small amount of his/her preferences and then selects possible desirable items according to these preferences which are refined and into a rating scale in the second level. In this way, the cascade hybrid RS avoids known problems of content-based or CF RS.

The fundamental idea behind our cascade hybrid recommendation approach was to mimic the social recommendation process in which someone has already identified some items according to his/her preferences and seeks the opinions of others about these items, so as to make the best selection of items that fall within his/her individual preferences. Experimental results reveal that our hybrid recommendation approach outperforms both a pure content-based approach or a pure CF technique. Experimental results from the comparison between the pure collaborative and the cascade content-based approaches demonstrate the efficiency of the first level. On the other hand, the comparison between the cascade content-based and the cascade hybrid approaches demonstrates the efficiency of the second level and justifies the use of the CF method in the second level.

8.2 Current and Future Work

In relation to the work reported in this book, we are currently investigating the possibility of incorporating similar ideas into the construction of RS that are able to recommend items not only to specific user, but also to groups of users. Such RS utilize a combination (fusion) of RS based on game theory.

Another direction of current and future work is along the exploration of machine learning approaches based on the transductive inference paradigm. Transductive SVM approaches that utilize only positive and unlabelled data form a new, unexplored direction for RS. Related research has the potential to lead to efficient solutions to the highly unbalanced nature of the classification problem of RS. As mentioned earlier in this book, it is common to be faced with situations in which positive and unlabelled examples are available but negative examples cannot be obtained without paying an additional cost. Therefore, the utilization of additional information that is contained

in unlabelled data can offer the RS new possibilities to learn the users preferences more efficiently and to provide better recommendations.

These and other research avenues are currently being explored and related results will be presented elsewhere in the future.