Adaptation, Learning, and Optimization 16

Fuchen Sun
Kar-Ann Toh
Manuel Grana Romay
Kezhi Mao *Editors*

Extreme Learning Machines 2013: Algorithms and Applications

Springer

Adaptation, Learning, and Optimization

Volume 16

Series editors

Meng-Hiot Lim, Nanyang Technological University, Singapore
email: emhlim@ntu.edu.sg

Yew-Soon Ong, Nanyang Technological University, Singapore
email: asysong@ntu.edu.sg

For further volumes:
http://www.springer.com/series/8335

About this Series

The role of adaptation, learning and optimization are becoming increasingly essential and intertwined. The capability of a system to adapt either through modification of its physiological structure or via some revalidation process of internal mechanisms that directly dictate the response or behavior is crucial in many real world applications. Optimization lies at the heart of most machine learning approaches while learning and optimization are two primary means to effect adaptation in various forms. They usually involve computational processes incorporated within the system that trigger parametric updating and knowledge or model enhancement, giving rise to progressive improvement. This book series serves as a channel to consolidate work related to topics linked to adaptation, learning and optimization in systems and structures. Topics covered under this series include:

- complex adaptive systems including evolutionary computation, memetic computing, swarm intelligence, neural networks, fuzzy systems, tabu search, simulated annealing, etc.
- machine learning, data mining & mathematical programming
- hybridization of techniques that span across artificial intelligence and computational intelligence for synergistic alliance of strategies for problem-solving.
- aspects of adaptation in robotics
- agent-based computing
- autonomic/pervasive computing
- dynamic optimization/learning in noisy and uncertain environment
- systemic alliance of stochastic and conventional search techniques
- all aspects of adaptations in man-machine systems.

This book series bridges the dichotomy of modern and conventional mathematical and heuristic/meta-heuristics approaches to bring about effective adaptation, learning and optimization. It propels the maxim that the old and the new can come together and be combined synergistically to scale new heights in problem-solving. To reach such a level, numerous research issues will emerge and researchers will find the book series a convenient medium to track the progresses made.

Fuchen Sun · Kar-Ann Toh
Manuel Grana Romay · Kezhi Mao
Editors

Extreme Learning Machines 2013: Algorithms and Applications

Springer

Editors
Fuchen Sun
Department of Computer Science and Technology
Tsinghua University
Beijing
People's Republic of China

Manuel Grana Romay
Department of Computer Science and Artificial Intelligence
Universidad Del Pais Vasco
San Sebastian
Spain

Kar-Ann Toh
School of Electrical and Electronic Engineering
Yonsei University
Seoul
Republic of Korea (South Korea)

Kezhi Mao
School of Electrical and Electronic Engineering
Nanyang Technological University
Singapore
Singapore

ISSN 1867-4534 ISSN 1867-4542 (electronic)
ISBN 978-3-319-04740-9 ISBN 978-3-319-04741-6 (eBook)
DOI 10.1007/978-3-319-04741-6
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014933566

Printed on acid-free paper

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

Contents

Stochastic Sensitivity Analysis Using Extreme Learning Machine

David Becerra-Alonso, Mariano Carbonero-Ruz, Alfonso Carlos Martínez-Estudillo and Francisco José Marténez-Estudillo

Abstract The Extreme Learning Machine classifier is used to perform the perturbative method known as Sensitivity Analysis. The method returns a measure of class sensitivity per attribute. The results show a strong consistency for classifiers with different random input weights. In order to present the results obtained in an intuitive way, two forms of representation are proposed and contrasted against each other. The relevance of both attributes and classes is discussed. Class stability and the ease with which a pattern can be correctly classified are inferred from the results. The method can be used with any classifier that can be replicated with different random seeds.

Keywords Extreme learning machine · Sensitivity analysis · ELM feature space · ELM solutions space · Classification · Stochastic classifiers

1 Introduction

Sensitivity Analysis (SA) is a common tool to rank attributes in a dataset in terms how much they affect a classifier's output. Assuming an optimal classifier, attributes that turn out to be highly sensitive are interpreted as being particularly relevant for the correct classification of the dataset. Low sensitivity attributes are often considered irrelevant or regarded as noise. This opens the possibility of discarding them for the sake of a better classification. But besides an interest in an improved classification, SA is a technique that returns a rank of attributes. When expert information about a dataset is available, researchers can comment on the consistency of certain attributes being high or low in the scale of sensitivity, and what it says about the relationship between those attributes and the output that is being classified.

D. Becerra-Alonso (✉) · M. Carbonero-Ruz · A. C. Martínez-Estudillo · F. J. Martínez-Estudillo
Department of Management and Quantitative Methods, AYRNA Research Group,
Universidad Loyola Andalucía, Escritor CastillaAguayo 4, Córdoba, Spain
e-mail: davidba25@hotmail.com

F. Sun et al. (eds.), *Extreme Learning Machines 2013: Algorithms and Applications*, Adaptation, Learning, and Optimization 16, DOI: 10.1007/978-3-319-04741-6_1,

In this context, the difference between a deterministic and a stochastic classifier is straightforward. Provided a good enough heuristics, a deterministic method will return only one ranking for the sensitivity of each one of the attributes. With such a limited amount of information it cannot be known if the attributes are correctly ranked, or if the ranking is due to a limited or suboptimal performance of the deterministic classifier. This resembles the long standing principle that applies to accuracy when classifying a dataset (both deterministic and stochastic): it cannot be known if a best classifier has reached its topmost performance due to the very nature of the dataset, or if yet another heuristics could achieve some extra accuracy. Stochastic methods are no better here, since returning an array of accuracies instead of just one (like in the deterministic case) and then choosing the best classifier is not better than simply giving a simple good deterministic classification. Once a better accuracy is achieved, the question remains: is the classifier at its best? Is there a better way around it?

On the other hand, when it comes to SA, more can be said about stochastic classifiers. In SA, the method returns a ranked array, not a single value such as accuracy. While a deterministic method will return just a simple rank of attributes, a stochastic method will return as many as needed. This allows us to claim a probabilistic approach for the attributes ranked by a stochastic method. After a long enough number of classifications and their corresponding SAs, an attribute with higher sensitivity will most probably be placed at the top of the sensitivity rank, while any attribute clearly irrelevant to the classification will eventually drop to the bottom of the list, allowing for a more authoritative claim about its relationship with the output being classified.

Section 2.1 briefly explains SA for any generalized classifier, and how sensitivity is measured for each one of the attributes. Section 2.2 covers the problem of dataset and class representability when performing SA. Section 2.3 presents the method proposed and its advantages. Finally, Sect. 3 introduces two ways of interpreting sensitivity. The article ends with conclusions about the methodology.

2 Sensitivity Analysis

2.1 General Approach

For any given methodology, SA measures how the output is affected by perturbed instances of the method's input [1]. Any input/output method can be tested in this way, but SA is particularly appealing for black box methods, where the inner complexity hides the relative relevance of the data introduced. The relationship between a sensitive input attribute and its relevance amongst the other attributes in dataset seems intuitive, but remains unproven.

In the specific context of classifiers, SA is a perturbative method for any classifier dealing with charted datasets [2, 3]. The following generic procedure shows the most common features of sensitivity analysis for classification [4, 5]:

(1) Let us consider the training set given by N patterns $D = \{(\mathbf{x}_i, t_i) : \mathbf{x}_i \in \mathbf{R}^n, t_i \in \mathbf{R}, i = 1, 2, \ldots, N\}$. A classifier with as many outputs as class-labels in D is trained for the dataset. The highest output determines the class assigned to a certain pattern. A validation used on the trained classifier shows a good generalization, and the classifier is accepted as valid for SA.

(2) The average of all patterns by attribute $\bar{\mathbf{x}} = \frac{1}{N}\sum_i \mathbf{x}_i$ results in an "average pattern" $\bar{\mathbf{x}} = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_j, \ldots, \bar{x}_M\}$. The "maximum pattern" $\mathbf{x}^{max} = \{x_1^{max}, x_2^{max}, \ldots, x_j^{max}, \ldots, x_M^{max}\}$ is defined as the vector containing the maximum values of the dataset for each attribute. The "the minimum" pattern is obtained in an analogous way $\mathbf{x}^{min} = \{x_1^{min}, x_2^{min}, \ldots, x_j^{min}, \ldots, x_M^{min}\}$.

(3) A perturbed pattern is defined as an average pattern where one of the attributes has been swapped either with its corresponding attribute in the maximum or minimum pattern. Thus, for attribute j, we have $\bar{\mathbf{x}}_j^{max} = \{\bar{x}_1, \bar{x}_2, \ldots, x_j^{max}, \ldots, \bar{x}_M\}$ and $\bar{\mathbf{x}}_j^{min} = \{\bar{x}_1, \bar{x}_2, \ldots, x_j^{min}, \ldots, \bar{x}_M\}$.

(4) These M pairs of perturbed patterns are then processed by the validated classifier. The y_{jk} outputs per class k returned are then recorded for each pair of maximum and minimum perturbed patterns, giving us the set $\{x_j^{max}, x_j^{min}, y_{jk}^{max}, y_{jk}^{min}\}$. Sensitivity for class k with respect to attribute j can be defined as: $S_{jk} = \frac{y_{jk}^{max} - y_{jk}^{min}}{x_j^{max} - x_j^{min}}$. The sign in S_{jk} indicates the arrow of proportionality between the input and the output of the classifier. The absolute value of S_{jk} can be considered a measurement of the sensitivity of attribute j with respect to class k. Thus, if Q represents the total amount of class-labels present in the dataset, attributes can be ranked according to this sensitivity as $S_j = \frac{1}{Q}\sum_k S_{jk}$.

2.2 Average Patterns' Representability

An average pattern like the one previously defined implies the assumption that the region around it in the attributes space is representative of the whole sample. If so, perturbations could return a representative measure of the sensitivity of the attributes in the dataset. However, certain topologies of the dataset in the attributes space can return an average pattern that is not even in the proximity of any other actual pattern of the dataset. Thus, it's representability can be put to question. Even if the average pattern finds itself in the proximity of other patterns, it can land on a region dominated by one particular class. The SA performed would probably become more accurate for that class than it would for the others. A possible improvement, would be to propose an average pattern per class. However, once again, topologies for each class in the attributes space might make their corresponding average pattern land in a non-representative region. Yet another improvement would be to choose the median pattern instead of the average, but once again, class topologies in the attributes space will be critical.

In other words, the procedure described in Sect. 2.1 is more fit for regressors than for classifiers. Under the right conditions, and the right dataset, it can suffice for sensitivity analysis. Once the weights of a classifier are determined, and the classifier is trained, the relative relevance that these weights assign to the different input attributes might be measurable in most or all of the attributes space. Only then, the above proposed method would perform correctly.

2.3 Sensitivity Analysis for ELM

The aim of the present work is to provide with improvements to this method in order to return a SA according to what is relevant when classifying patterns in a dataset, regardless of the topology of the attributes space. Three improvements are being proposed:

- The best representability obtainable from a dataset is the one provided by all its patterns. Yet performing SA to all patterns can be too costly when using large datasets. On the other end there is the possibility of performing SA only to the average or median patterns. This is not as costly but raises questions about the representability of such patterns. The compromise here proposed is to only study the SA of those samples of a certain class, in a validation subset, that have been correctly classified by the already assumed to be good classifier. The sensitivity per attribute found for each one of the patterns will be averaged with that of the rest of the correctly classified patterns of that class, in order to provide with a measure of how sensitive each attribute is for that class.
- Sensitivity can be measured as a ratio between output and input. However, in classifiers, the relevance comes from measuring not just the perturbed output differences, but from measuring the perturbation that takes one pattern out of its class, according to the trained classifier. The boundaries where the classifier assigns a new (and incorrect) class to a pattern indicate more accurately the size of that class in the output space, and with it, a measure of the sensitivity of the input. Any small perturbation in an attribute that makes the classifier reassign the class of that pattern, indicates a high sensitivity of that attribute for that class. This measurement becomes consistent when averaged amongst all patterns in the class.
- Deterministic one-run methods will return a single attribute ranking, as indicated in the introduction. Using the Single Hidden Layer Feedforward (SLFN) version of ELM [6, 7], every new classifier, with its random input weights and its corresponding output weights, can be trained, and SA can then be performed. Thus, every classifier will return sensitivity matrix made of SA measurements for every attribute and every class. These can in turn be averaged into a sensitivity matrix for all classifiers. If most or all SA performed for each classifier are consistent, certain classes will most frequently appear as highly sensitive to the perturbation of certain attributes. The fact that ELM, with its random input weights, gives such a consistent SA, makes a strong case for the reliability of ELM as a classifier in general, and for SA in particular.

These changes come together in the following procedure:

(1) Let us consider the training set given by N patterns $D = \{(\mathbf{x}_i, t_i) : \mathbf{x}_i \in \mathbf{R}^n, t_i \in \mathbf{R}, i = 1, 2, \ldots, N\}$. A number L of ELMs are trained for the dataset. A validation sample is used on the trained ELMs. A percentage of ELMs with the highest validation accuracies is chosen and considered suitable for SA.
(2) For each ELM selected, a new dataset is made with only those validation patterns that have been correctly classified. This dataset is then divided into subsets for each class.
(3) For each attribute x_j in each pattern $\mathbf{x} = \{x, x_2, \ldots, x_j, \ldots, x_M\}$ that belongs to the subset corresponding to class k, that has been correctly classified by the q-th classifier, SA is measured as follows:
(4) x_j is increased in small intervals within $(x_j, x_j^{max} + 0.05(x_j^{max} - x_j^{min}))$. Each increase creates a pattern $\mathbf{x}^{pert} = \{x_1, x_2, \ldots, x_j^{pert}, \ldots, x_M\}$ that is then tested on the q-th classifier. This process is repeated until the classifier returns a class other than k. The distance covered until that point is defined as Δx_j^+.
(5) x_j is decreased in small intervals within $(x_j^{min} - 0.05(x_j^{max} - x_j^{min}), x_j)$. Each decrease creates a pattern $\mathbf{x}^{pert} = \{x_1, x_2, \ldots, x_j^{pert}, \ldots, x_M\}$ that is then tested on the q-th classifier. This process is repeated until the classifier returns a class other than k. The distance covered until that point is defined as Δx_j^-.
(6) Sensitivity for attribute j in pattern i, that is part of class-subset k, when studying SA for classifier q is: $S_{jkqi} = 1/(\min(\Delta x_j^+, \Delta x_j^-))$. If the intervals in steps 4 and 5 are covered without class change (hence, no Δx_j^+ or Δx_j^- are recorded), then $S_{jkqi} = 0$.
(7) The sensitivity of all the patterns within a class subset are averaged according to $S_{jkq} = \frac{1}{R_{kq}} \sum_i S_{jkqi}$, where R_{kq} is the number of correctly classified patterns on each classifier, for each class.
(8) The sensitivity of all classifiers is averaged according to $S_{jk} = \frac{1}{Q} \sum_q S_{jkq}$ where Q is the number ELMs that were considered as suitable for SA in step 1. This S_{jk} is the sensitivity matrix above mentioned. It represents the sensitivity per attribute and class of the correctly classified patterns in the validation subset, and assuming a good representability, the sensitivity of the entire dataset.
(9) Attribute and class based sensitivity vectors can then be defined by averaging the sensitivity matrix according to $S_j = \frac{1}{M} \sum_q S_{jk}$ and $S_k = \frac{1}{K} \sum_q S_{jk}$ respectively. K is the total number of classes in the dataset.

3 Results

3.1 Datasets Used, Dataset Partition and Method Parameters

Well known UCI repository datasets [8] are used to calculate results for the present model. Table 1 shows the main characteristics of the datasets used. Each dataset is partitioned for a hold-out of 75 % for training and 25 % for validation, keeping class

Table 1 UCI dataset general features

Dataset	# Patterns	# Attributes	# Classes	# Patterns per class
Haberman	306	3	2	225-81
Newthyroid	215	5	3	150-35-30
Pima	768	8	2	500-268
Vehicle	946	18	4	212-199-218-217

Table 2 Haberman

Sensitivity matrix	Attr.1	Attr.2	Attr.3	Class vec.	Rank
Class 1	0.0587	0.0446	0.1873	0.0968	2nd
Class 2	0.3053	0.2477	0.5067	0.3532	1st
Attribute vec.	0.1820	0.1461	0.3470		
Rank	2nd	3rd	1st		

representability in both subsets. The best $Q = 300$ out of $L = 3000$ classifiers will be considered as suitable for SA. All ELMs performed will have 20 neurons in the hidden layer, thus avoiding overfitting in all cases.

3.2 Sensitivity Matrices, Class-Sensitivity Vectors, Attribute-Sensitivity Vectors

Filters and wrappers generally offer a rank for the attributes as an output. SA for ELM offers that rank, along with a rank per class. For each dataset, the sensitivity matrices in this section are presented with their class and attribute vectors, that provide with a rank for class and attribute sensitivity. This allows for a better understanding of classification outcomes that were otherwise hard to interpret. The following are instances of this advantage:

- Many classifiers tend to favor the correct classification of classes with the highest number of patterns, when working with imbalanced datasets. However, the sensitivity matrices for Haberman and Pima (Tables 2 and 4), show another possible reason for such a result. For both datasets, class 2 is not just the minority class, and thus more prone to be ignored by a classifier. Class 2 is also the most sensitive. In other words, it takes a much smaller perturbation to meet the border where a classifier re-interprets a class 2 pattern into a class 1. On the other hand, the relatively low sensitivity of class 1 indicates a greater chance for patterns to be assigned to this class. It is only coincidental that class 1 also happens to be the majority class.
- The results for Newthyroid (Table 3) show a similar scenario: class 2, one of the two minority classes, is highly sensitive. In this case, since the two minority

Table 3 Newthyroid

Sensitivity matrix	Attr.1	Attr.2	Attr.3	Attr.4	Attr.5	Class vec.	Rank
Class 1	0.0159	0.1395	0.0622	0.0742	0.0915	0.0767	3rd
Class 2	0.3038	0.9503	1.8846	0.3813	0.3494	0.7739	1st
Class 3	0.0230	0.0890	0.0470	0.1363	0.1207	0.0832	2nd
Attribute vector	0.1142	0.3929	0.6646	0.1972	0.1872		
Rank	5th	2nd	1st	3rd	4th		

Table 4 Pima

Sensitivity matrix	Attr.1	Attr.2	Attr.3	Attr.4	Attr.5	Attr.6	Attr.7	Attr.8	Class vec.	Rank
Class 1	0.0569	0.0483	0.0862	0.0553	0.0434	0.0587	0.0416	0.0609	0.0564	2nd
Class 2	0.2275	0.1655	0.3238	0.2085	0.1656	0.2413	0.2798	0.2534	0.2332	1st
Attribute vector	0.1422	0.1069	0.2050	0.1319	0.1045	0.1500	0.1607	0.1571		
Rank	5th	7th	1st	6th	8th	4th	2nd	3rd		

classes (2 and 3) have similar population sizes, it can be expected to have better classification results for class 3, for the same reasons above mentioned.

- Classes with a highest averaged sensitivity don't imply the highest sensitivity class per attribute. Vehicle (Table 5) shows this: although class 3 is the most sensitive, sensitivities for attributes 1, 2, 9, 15 and 17 are not the highest for this class. Different attributes are fit for a better classification of different classes. Orthogonality in the rows of the sensitivity matrix implies perfect classification.

3.3 Attribute Rank Frequency Plots

Another way to easily spot relevant or irrelevant attributes is to use attribute rank frequency plots. Every attribute selection method assigns a relevance-related value to all attributes. From such values, an attribute ranking can be made. SA with ELM provides with as many ranks as the number of classifiers chosen as apt for SA. In Figs. 1 through 4, each attribute of the dataset is represented by a color. Each column represents the sensitivity rank in increasing order. Each classifier will assign a different attribute color to each one of the columns. After the $Q = 300$ classifiers have assigned their ranked sensitivity colors, some representations show how certain attribute colors dominate the highest or lowest rank positions. The following are interpretations extracted from these figures:

- Both classes in Haberman (Fig. 1) show a high sensitivity to attribute 3. This corresponds to the result obtained in Table 2. Most validated ELM classifiers

Table 5 Vehicle

Sensitivity matrix	Att.1	Att.2	Att.3	Att.4	Att.5	Att.6	Att.7	Att.8	Att.9	.....		
Class 1	0.0633	0.0389	0.0299	0.0403	0.0454	0.0261	0.0486	0.0328	0.0201	.....		
Class 2	0.2219	0.1917	0.1080	0.0735	0.0851	0.1201	0.0689	0.1181	0.2607	.....		
Class 3	0.1779	0.1786	0.1316	0.1134	0.0981	0.1268	0.1058	0.2005	0.1659	.....		
Class 4	0.0975	0.0354	0.0623	0.0750	0.0670	0.0453	0.0837	0.0572	0.0275	.....		
Attribute vector	0.1401	0.1112	0.0829	0.0755	0.0739	0.0796	0.0767	0.1021	0.1185	.....		
Rank	1st	5th	13th	17th	18th	14th	16th	6th	3rd	.....		
.....	Att.10	Att.11	Att.12	Att.13	Att.14	Att.15	Att.16	Att.17	Att.18		Class vec.	Rank
.....	0.0511	0.0431	0.0590	0.0403	0.0336	0.0292	0.0437	0.0320	0.0365		0.0397	4th
.....	0.0787	0.0936	0.1005	0.0965	0.0890	0.1702	0.2636	0.1367	0.1250		0.1334	2nd
.....	0.1353	0.0949	0.1116	0.2515	0.1410	0.1422	0.1300	0.1156	0.1678		0.1438	1st
.....	0.0772	0.0773	0.0982	0.0597	0.0717	0.0662	0.0746	0.0671	0.0541		0.0665	3rd
.....	0.0856	0.0772	0.0923	0.1120	0.0838	0.1020	0.1280	0.0878	0.0958			
.....	11th	15th	9th	4th	12th	7th	2nd	10th	8th			

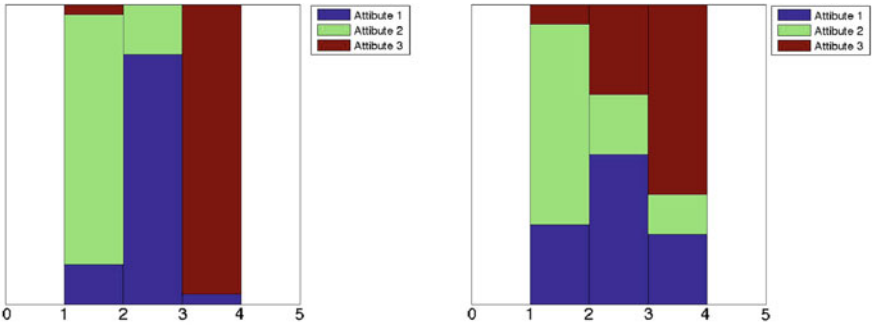

Fig. 1 Haberman for classes 1 and 2

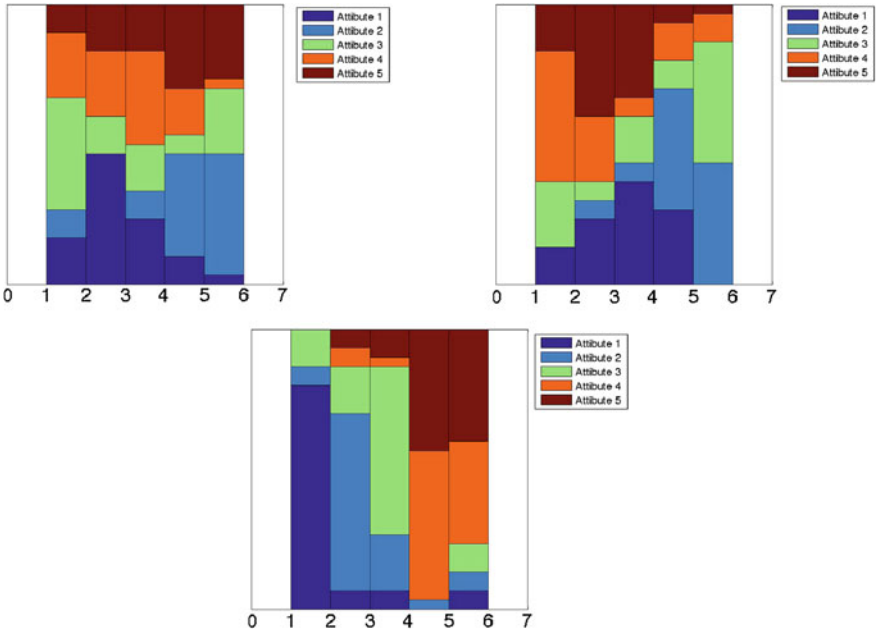

Fig. 2 Newthyroid for classes 1, 2 and 3

consider attribute 3 to be the most sensitive when classifying both classes. The lowest sensitivity of attribute 2 is more apparent when classifying class 1 patterns.

- In Newthyroid (Fig. 2) both attributes 4 and 5 are more sensitive when classifying class 3 patterns. The same occurs for attributes 2 and 3 when classifying class 2 patterns, and for attributes 2 and 5 when classifying class 1 pattern. Again, this is all coherent with the results in Table 3.
- Pima (Fig. 3) shows attribute 3 to be the most sensitive for the classification of both classes, especially class 1. This corresponds to what was found in Table 4.

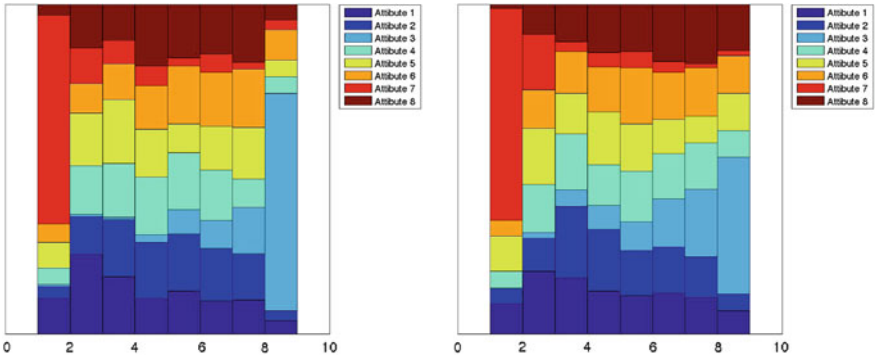

Fig. 3 Pima for classes 1 and 2

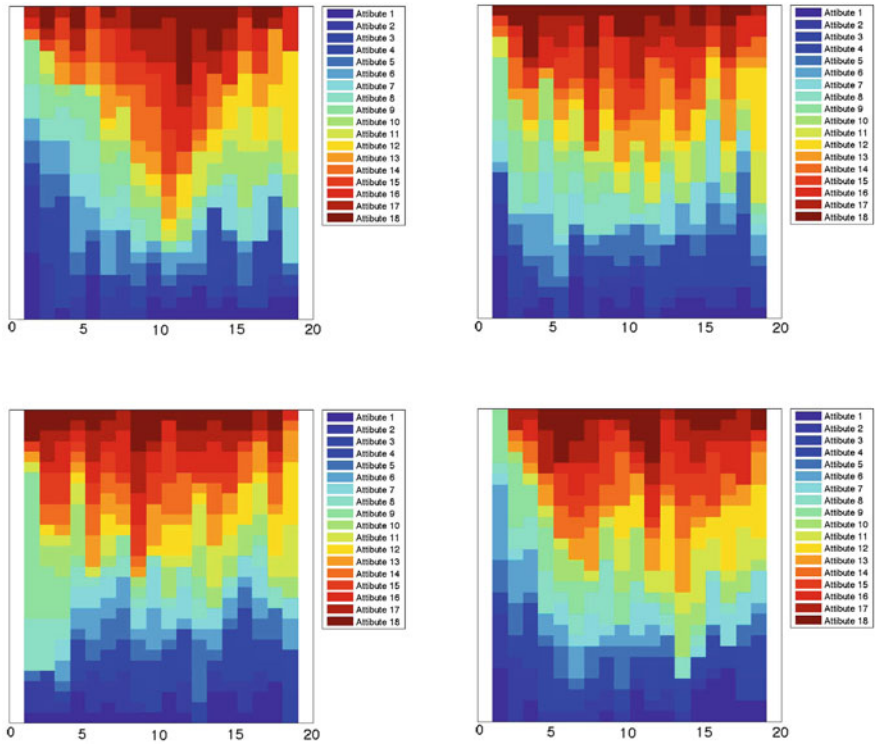

Fig. 4 Vehicle for classes 1, 2, 3 and 4

However, while Fig. 3 shows attribute 7 to be the least sensitive for both classes, attribute 7 holds second place in the averaged sensitivity attribute vector of Table 4. It is in cases like these where both the sensitivity matrix and this representation are necessary in order to interpret the results. Attribute 7 is ranked as low by most

classifiers, but has a relatively high averaged sensitivity. The only way to hint at this problem without the attribute rank frequency plots is to notice the dispersion for different classes for each attribute. In this case, the ratio between the sensitivity of attribute 7 for class 2 and attribute 7 for class 1 is the biggest of all, making the overall sensitivity measure for attribute 7 less reliable.

- The interpretation of more than a handful of attributes can be more complex, as we can see in Table 5. However, attribute rank frequency plots can quickly make certain attributes stand out. Figure 4 shows how attributes 8 and 9 are generally low sensitive to classification of class 3 of the Vehicle dataset. Other attributes are more difficult to interpret in these representations, but the possibility of detecting high or low attributes in the sensitivity rank can be particularly useful.

4 Conclusions

This work has presented a novel methodology for the SA analysis of ELM classifiers. Some refinements have been proposed for the traditional SA methodology, that seems to be more suitable for regressors. The advantage of creating stochastic classifiers with different random seeds of input weights allows for a multitude of classifiers to approximate sensitivity measures. This is something that deterministic classifiers (without such random seed) cannot do. A large enough number of validated classifiers can in principle provide with a more reliable measure of sensitivity.

Two different ways of representing the results per class and attribute have been proposed. Each one of them emphasizes a different way of ranking sensitivities according to their absolute (sensitivity matrix) or relative (attribute rank frequency plots) values. Both measures are generally consistent with each other, but sometimes present differences that can be used to assess the reliability of the sensitivities obtained.

Any classifier with some form of random seed, like the input weights in ELM, can be used to perform Stochastic SA, where the multiplicity of classifiers indicate a reliable sensitivity trend. ELM, being a speedy classification method, is particularly convenient for this task. The consistency in the results presented also indicate the inherent consistency of different validated ELMs as classifiers.

This work was supported in part by the TIN2011-22794 project of the Spanish Ministerial Commision of Science and Technology (MICYT), FEDER funds and the P11-TIC-7508 project of the “Junta de Andalucía” (Spain).

References

1. A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, S. Tarantola, *Global Sensitivity Analysis: The Primer* (Wiley-Interscience, Hoboken, 2008)
2. S. Hashem, Sensitivity analysis for feedforward artificial neural networks with differentiable activation functions, in *International Joint Conference on Neural Networks (IJCNN’92)*, vol. 1 (1992), pp. 419–424

3. P.J.G. Lisboa, A.R. Mehridehnavi, P.A. Martin, The interpretation of supervised neural networks, in *Workshop on Neural Network Applications and Tools* (1993), pp. 11–17
4. A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, S. Tarantola, Variance based sensitivity analysis of model output. design and estimator for the total sensitivity index. Comput. Phys. Commun. **181**(2), 259–270 (2010)
5. A. Palmer, J.J. Montaño, A. Calafat, Predicción del consumo de éxtasis a partir de redes neuronales artificiales. Adicciones **12**(1), 29–41 (2000)
6. G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in *Proceedings 2004 IEEE International Joint Conference on Neural Networks* (2004), pp. 985–990
7. G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications. Neurocomputing **70**(1–3), 489–501 (2006)
8. C.L. Blake, C.J. Merz, UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html (1998)

Efficient Data Representation Combining with ELM and GNMF

Zhiyong Zeng, YunLiang Jiang, Yong Liu and Weicong Liu

Abstract Nonnegative Matrix Factorization (NMF) is a powerful data representation method, which has been applied in many applications such as dimension reduction, data clustering etc. As the process of NMF needs huge computation cost, especially when the dimensional of data is large. Thus a ELM feature mapping based NMF is proposed [1], which combined Extreme Learning Machine (ELM) feature mapping with NMF (EFM NMF), can reduce the computational of NMF. However, the random parameter generating based ELM feature mapping is nonlinear. And this will lower the representation ability of the subspace generated by NMF without sufficiently constrains. In order to solve this problem, this chapter propose a novel method named Extreme Learning Machine feature mapping based graph regulated NMF (EFM GNMF), which combines ELM feature mapping with Graph Regularized Nonnegative Matrix Factorization (GNMF). Experiments on the COIL20 image library, the CMU PIE face database and TDT2 corpus show the efficiency of the proposed method.

Keywords Extreme learning machine · ELM feature mapping · Nonnegative matrix factorization · Graph regularized nonnegative matrix factorization · EFM NMF · EFM GNMF · Data representation

Z. Zeng (✉)
Hangzhou Dianzi University, Hangzhou 310018, China
e-mail: zzy07053437@163.com

Y. Jiang
Huzhou Teachers College, Huzhou 313000, China

Y. Liu (✉) · W. Liu
Zhejiang University, Hangzhou 310027, China
e-mail: yongliu@iipc.zju.edu.cn

F. Sun et al. (eds.), *Extreme Learning Machines 2013: Algorithms and Applications*, Adaptation, Learning, and Optimization 16, DOI: 10.1007/978-3-319-04741-6_2,

1 Introduction

Nonnegative matrix factorization (NMF) techniques have been frequently applied in data representation and document clustering. Given an input data matrix X, each column of which represents a sample, NMF aims to find two factor matrices U and V using low-rank approximation such that $X \approx UV$. Each column of U represents a base vector, and each column of V describes how these base vectors are combined fractionally to form the corresponding sample in X [2, 3].

Compared to other methods, such as principal component analysis (PCA) and independent component analysis (ICA), the nonnegative constraints lead to a parts-based representation because they allow only additive, not subtractive, combinations. Such a representation encodes the data using few active components, which makes the basis easy to interpret. NMF has been shown to be superior to SVD in face recognition [4] and document clustering [5]. It is optimal for learning the parts of objects.

However, NMF cost huge computing when it disposes high-dimensional data such as image data. ELM feature mapping [6, 7] as an explicit feature mapping techniques was proposed. It is more convenient than kernel function and can get more satisfactory results for classification and regression [8, 9]. NMF is a linear model, using nonlinear feature mapping techniques, it will be able to deal with nonlinear correlation in data. Then, ELM based methods is not sensitive to the number of hidden layer nodes, provided that a large enough number is selected [1]. So, using ELM feature mapping to improve the efficiency of NMF is feasible. Nevertheless, ELM feature mapping NMF (EFM NMF) can not keep generalization performance as NMF. Only the non-negative constraints in NMF unlike other subspace methods (e.g. Locality Preserving Projections (LPP) method [10]), may not be sufficiently understand the hidden structure of the space which transform from the original data. A wide variety of subspace constraints can be formulated into a certain form such as PCA and LPP to enforce general subspace constraints into NMF. Graph Regularized Nonnegative Matrix Factorization (GNMF [11], which discovers the intrinsic geometrical and discriminating structure of the data space by implant a geometrical regularization, is more powerful than the ordinary NMF approach. In order to obtain efficiency and keep generalization representation performance simultaneously, we proposed method named EFM GNMF which combined ELM feature mapping with GNMF.

The rest of the chapter is organized as follows: Sect. 2 gives a brief review of the ELM, ELM Feature mapping, NMF and GNMF. The EFM NMF and EFM GNMF are presented in Sect. 3. The experimental result will be shown in Sect. 4. Finally, in Sect. 5, we conclude the chapter.

2 A Review of Related Work

In this section, a short review of the original ELM algorithm, ELM Feature mapping, NMF and GNMF are given.

2.1 ELM

For N arbitrary distinct samples (x_i, t_i), where $x_i = [x_{i1}, x_{i2}, \ldots, x_{iD}]^T \in R^D$ and $t_i = [t_{i1}, t_{i2}, \ldots, t_{iK}]^T \in R^K$, standard SLFNs with L hidden nodes and activation function h(x) are mathematically modeled as:

$$\sum_{i=1}^{L} \beta_i h_i \left(x_j\right) = \sum_{i=1}^{L} \beta_i h_i \left(w_i \cdot x_j + b_i\right) = o_j \tag{1}$$

where j = 1, 2, ..., N. Here $w_i = [w_{i1}, w_{i2}, \ldots, w_{iD}]^T$ is the weight vector connecting the ith hidden node and the input nodes, $\beta_i = [\beta_{i1}, \ldots, \beta_{iK}]^T$ is the weight vector connecting the ith hidden node and the output nodes, and b_i is the threshold of the ith hidden node. The standard SLFNs with L hidden nodes with activation function h (x) can be compactly written as [12–15]:

$$\mathrm{H}\beta = \mathrm{T} \tag{2}$$

where

$$\mathrm{H} = \begin{bmatrix} h_1 (w_1 \cdot x_1 + b_1) & \ldots & h_L (w_L \cdot x_1 + b_L) \\ \vdots & \vdots & \vdots \\ h_1 (w_1 \cdot x_N + b_1) & \ldots & h_L (w_L \cdot x_N + b_L) \end{bmatrix} \tag{3}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix} \text{ and } \mathrm{T} = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix} \tag{4}$$

Different from the conventional gradient-based solution of SLFNs, ELM simply solves the function by

$$\beta = H^+ T \tag{5}$$

H^+ is the Moore-Penrose generalized inverse of matrix H.

2.2 ELM Feature Mapping

As show in Sect. 2.1 above, $h(x)$ as the ELM feature mapping, maps the sample x_1 from the D-dimensional input space to the L-dimensional hidden-layer feature space which is called ELM feature space. The ELM feature mapping process is shown in Fig. 1.

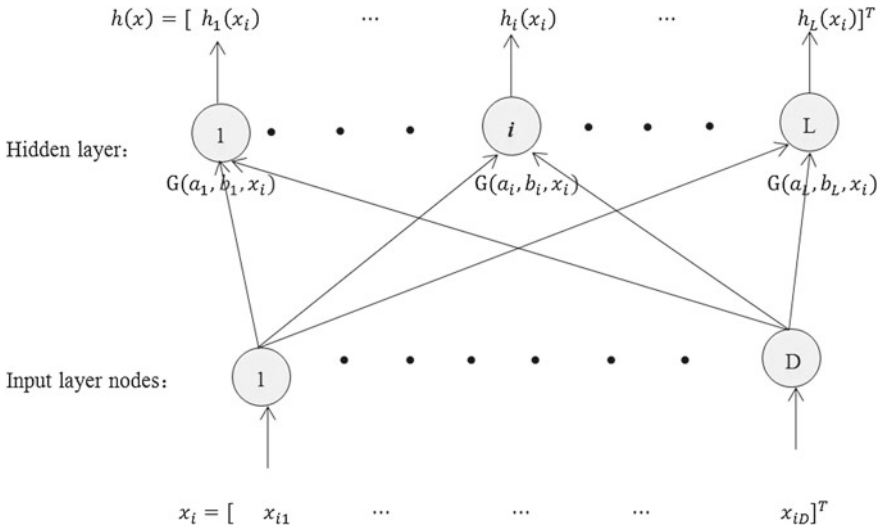

Fig. 1 ELM feature mapping process (cited from [1])

The ELM feature mapping can be formally described as:

$$h\,(x_i) = [h_1\,(x_i)\ , \ldots , h_L\,(x_i)]^T = [\mathrm{G}(a_1,b_1,x_i)\ , \ldots , \mathrm{G}(a_L,b_L,x_i)]^T \quad (6)$$

where $\mathrm{G}(a_i,\ b_i,\ x_i)$ is the output of the i-th hidden node. The parameters which need not to be tuned, $(a_i,\ b_i)_{i=1}^{L}$, can be randomly generated according to any continuous probability distribution. It is that ELM feature mapping is very convenient. Huang in [6, 7] has proved that almost all almost all nonlinear piecewise continuous functions can be used as the hidden-node output functions directly [1].

2.3 GNMF

NMF [16–18] is a matrix factorization algorithm that focuses on the analysis of data matrices whose elements are nonnegative. Consider a data matrix $\mathrm{X} = [x_1, \ldots, x_D] \in \mathrm{R}^{D\times M}$ each column of X is a sample vector which consists of D features. Generally, NMF can be presented as the following optimization problem:

$$C\ (\mathrm{X} \approx \mathrm{UV}),\ \text{s.t.}\ \ \mathrm{U},\mathrm{V} \geq 0 \quad (7)$$

NMF aims to find two non-negative matrices $\mathrm{U} = [u_{ij}] \in \mathrm{R}^{D\times K}$ and $\mathrm{V} = [v_{ij}] \in R^{K\times M}$ whose product can well approximate the original matrix X. $C(\cdot)$ denotes the cost function.

NMF performs the learning in the Euclidean space which cover the intrinsic geometrical and discriminating. To find a compact representation which uncovers the hidden semantics and simultaneously respects the intrinsic geometric structure, Cai et al. [11] proposed construct an affinity graph to encode the information and seek a matrix factorization to respects the graph structure in GNMF.

$$O_{GNMF} = \|X - UV\|_F^2 + \lambda tr\left(V^T LV\right) \quad st.U \geq 0, V \geq 0 \tag{8}$$

where L is graph Laplacian. The adjacent graph, which each vertex corresponding to a sample and the weight between vertex $\vec{x}_i$ and vertex $\vec{x}_j$, is defined as [19]

$$S_{ij} = \begin{cases} 1, & if\ \vec{x}_i \in N_k\left(\vec{x}_j\right)\ or\ \vec{x}_j \in N_k\left(\vec{x}_i\right) \\ 0, & otherwise \end{cases} \tag{9}$$

where $N_k\left(\vec{x}_i\right)$ signifies the set of k nearest neighbors of $\vec{x}_i$. Then L is written as L = T − W, where T is a diagonal matrix whose diagonal entries are column sums of S, i.e., $T_{ii} = \sum_i W_{ij}$.

3 EFM GNMF

In this section, we will present our EFM GNMF. EFM NMF will improve computational efficiency by reducing the feature number. But ELM feature mapping, which using random parameter, is a nonlinear feature mapping technique. This will lower the ability of representation of the subspace generating from NMF without sufficiently constrains. In order to solve this problem, this chapter propose a novel method EFM GNMF, combined ELM feature mapping with Graph Regularized Nonnegative Matrix Factorization (GNMF). Graph constrain guarantee that using ELM feature space in NMF can also has the local manifold feature. The proposed algorithm puts as follows:

(1) *Setting the number of hidden-layer nodes* $L < D$ *and threshold* $\varepsilon > 0$.
(2) *Calculate the weight matrix W on the nearst neighbor graph of the original data.*
(3) *Using ELM feature mapping* $h(x) = [h_1(x), \ldots, h_i(x), \ldots, h_L(x)]^T$ *transform the original data into ELM feature space. The original data with D-dimensional will transform into L-dimensional*

$$\mathrm{X} = [x_1, \ldots, x_D] \epsilon \mathrm{R}^{D \times M} \rightarrow \mathrm{H} = [h_1, \mathrm{ldots}, h_L] \epsilon \mathrm{R}^{L \times M}$$

(4) *Initialize* $U \epsilon R^{L \times K}$, $V \epsilon R^{K \times M}$ *and the regularization* λ *with nonnegative values.*
(5) *Using W as the weight matrix on the nearst neighbor graph of the ELM feature space data H*
(6) *Iterate for each i, j, and i until convergence* $(err < \varepsilon)$ *or reached the maxiamal iterations* [10]

$$(a)\ U_{ij}^{t+1} \leftarrow U_{ij}^t \frac{(HV^T)_{ij}}{(\mathrm{U}^t VV^T)_{ij}}$$

$$(b)\ V_{ij}^{t+1} \leftarrow V_{ij}^t \frac{(H^T U + \lambda WV^T)_{ij}}{((U^T UV^t)^T + \lambda DV^T)_{ij}}$$

$$(c)\ \mathrm{err} \leftarrow \max\left\{\frac{\left\|U^{t+1} - U^t\right\|}{\sqrt{LK}}, \frac{\left\|V^{t+1} - V^t\right\|}{\sqrt{KM}}\right\}$$

Table 1 Statistics of the three data sets

Datasets	Size (N)	Dimensionality (M)	# of classes (K)
COIL20	1440	1024	20
PIE	2856	1024	68
TDT2	9394	36771	30

4 Experiments Results

In this section, three of the mostly used datasets COIL20 image library, the CMU PIE face database and TDT2 corpus are used to prove the efficiency of the proposed algorithm. The important statistics of these data sets are summarized below (see Table 1). To make the results valid, every algorithm run 20 times on each data set and obtains the average result. This chapter chooses the sigmoid function as the ELM feature mapping activation function for it is most used. To obtain the efficiency and performance of these algorithms with different numbers of hidden nodes, we adopt the integrated data sets. K-mean is used as the cluster to test the generalization performance. The clustering result is evaluated by comparing the obtained label of each sample with the label provided by the data set. Two metrics, the accuracy (AC) and the normalized mutual information metric (NMI) are used to measure the clustering performance [11]. Please see [20] for the detailed definitions of these two metrics. All the algorithms are carried out in MATLAB 2011 environment running in a Core 2, 2.5 GHZ CPU.

4.1 Compared Algorithms

To demonstrate how the efficiency of NMF can be improve by our method, we compare the computing time of four algorithms (NMF, GNMFEFM NMF, EFM GNMF). The hidden nodes number is set as 1, 2, 4, 6, . . . , 18 within 18; 20, 30, . . ., 100 from 20 to 100; 125, 150, . . . , 600 from 125 to 600; 650, 700, . . . , 1000 from 600 to 1000. Comparing the clustering performance of these methods is also revealed (The values of clustering performance change little when nodes number surpass 100, that is, only the result of the hidden nodes number from 1to 100 is shown). The max iterations in NMF, GNMF, EFM NMF and EFM GNMF are 100.

Figure 2 show the time comparing results on the COIL20, PIE, and TDT2 data sets respectively. Over all, we can see that ELM feature mapping methods (EFM NMF, EFM GNMFF) is faster than NMF and GNMF when hidden nodes number is low. With the hidden nodes number increased, the computation time is monotone increasing. When the number is high, the computation time of EFM NMF or EFM NMF will exceed NMF and GNMF. Comparing the computation time of EFM NMF with EFM GNMFF, we can see that EFM NMF is faster than EFM GNMF. However, by increasing the hidden nodes number, the time difference between EFM NMF

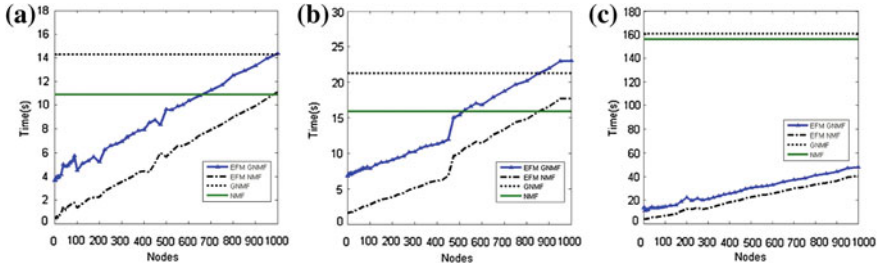

Fig. 2 Computation time on **a** COIL20 **b** PIE **c** TDT2

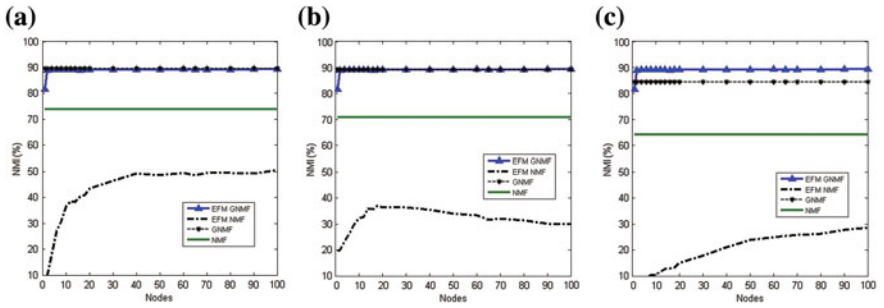

Fig. 3 NMI measure clustering performance. **a** COIL20 **b** PIE **c** TDT2

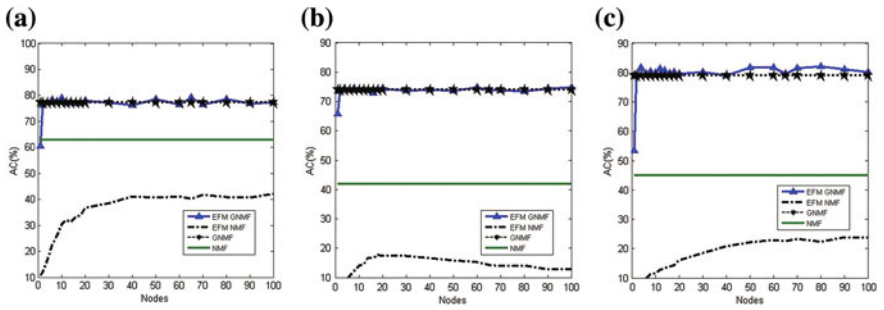

Fig. 4 AC measure clustering performance. **a** COIL20 **b** PIE **c** TDT2

and EFM GNMFF close to a constant. That is because GNMF need to compute the weight matrix W.

Figures 3 and 4 show clustering performance comparing results on data sets respectively. Obviously, EFM NMF can not get the approximate clustering performance as NMF. Nevertheless, EFM GNMF can reach approximate clustering performance as GNMF, provided that a large enough hidden nodes number is selected.

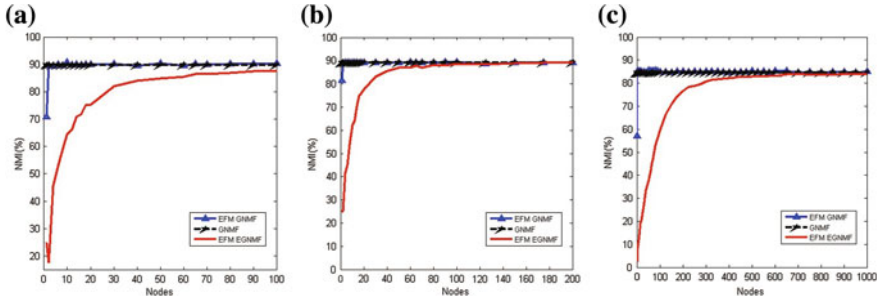

Fig. 5 NMI measure clustering performance. Comparing EFM GNMF with EFM EGNMF **a** COIL20 **b** PIE **c** TDT2

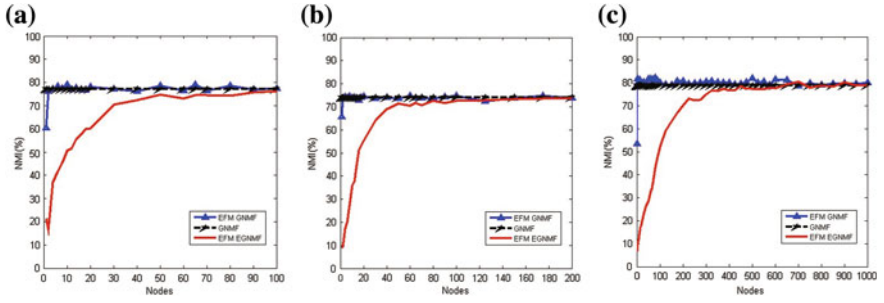

Fig. 6 AC measure clustering performance. Comparing EFM GNMF with EFM EGNMF **a** COIL20 **b** PIE **c** TDT2

4.2 Original Graph Versus ELM Feature Space Graph

We denote the method that uses ELM feature space neighbor graph to replace the original space neighbor graph in the EFM GNMF as ELM feature mapping with ELM space graph NMF (EFM EGNMF).

As show in Figs. 5 and 6, EFM EGNMF can also reach similar clustering performance as GNMF. However, comparing with EFM GNMF, EFM EGNMF need more hidden nodes number to reach similar clustering performance as GNMF. So, ELM feature mapping may be can simulate the local manifold of the original data.

4.3 The Geometric Structure of ELM Feature Space

Prompt by Sect. 4.2, we speculate that ELM feature mapping can keep approximated geometric of original data when transforming the original data space into ELM feature space with a large number of hidden nodes. In order to discover whether

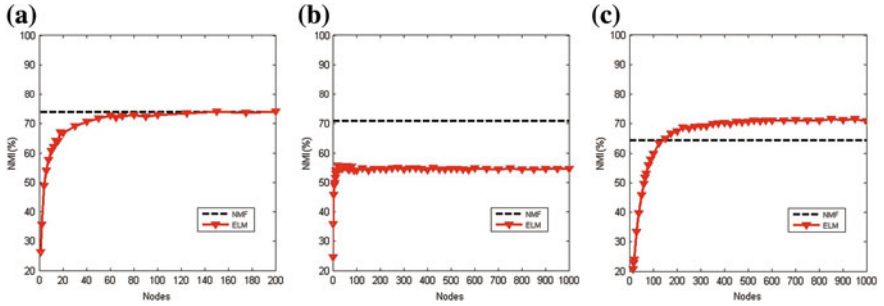

Fig. 7 NMI measure clustering performance. Comparing ELM with NMF **a** COIL20 **b** PIE **c** TDT2

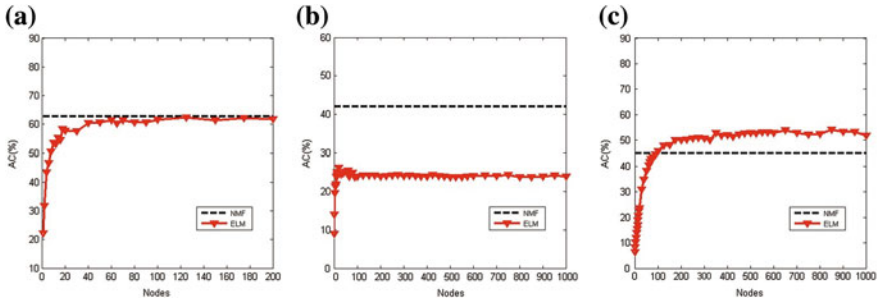

Fig. 8 AC measure clustering performance. Comparing ELM with NMF **a** COIL20 **b** PIE **c** TDT2

ELM can keep approximated geometric structure of original data, we compare the cluster performance of ELM with NMF under different hidden nodes number.

As showed in Figs. 7a, c and 8a, c, after transform into ELM feature space, the data can reach similar clustering performance as NMF, provided the hidden nodes number is enough. Figure 8b Even the hidden nodes number is huge, the data has an approximate constant gap with NMF in clustering performance. We can find that the number of samples for each class is 72 in COIL20 data set, 42 in PIE data set, 313 in TDT2 data set. So, for ELM feature mapping, it may be that having more samples for each class can get better performance. ELM feature mapping can keep approximated geometric of original data not only need enough hidden nodes, but also need enough samples for each class. This need more experiments to confirm.

4.4 Combining ELM and NMF with Other Constrains

In this chapter, neighbor graph based constrain has been proved powerful. Then, NMF can combine with a wide variety of subspace constraints that can be formulated into a certain form such as PCA and LPP. ELM feature mapping combined with general subspace constrained NMF(GSC NMF) can be the future work.

5 Conclusions

This chapter proposes a new method named EFM GNMF, which applies ELM feature mapping and graph constrains to solve computational problem in NMF without lose generalization performance. Experiments show that when dispose with high-dimensional data, the efficiency of EFM GNMF is better than directly using NMF or GNMF. Also, EFM GNMF is compared with GNMF in clustering performance. Unlike EFM NMF get efficiency without keep generalization performance, EFM GNMF can reach similar result as GNMF. Moreover, the difference of using the neighbor graph of the original data space with ELM feature space is raised. ELM feature mapping can keep approximated geometric structure hidden in the original data.

Acknowledgments We want to thank Dr. Huang Guangbin from NTU and Dr. Jin Xin from Chinese Academy of Sciences. They provide us with some codes and details of Extreme Learning Machine. This work was supported by the National Natural Science Foundation Project of China (61173123) and the Natural Science Foundation Project of Zhejiang Province (LR13F030003).

References

1. Q. He, X. Jin, C. Du, F. Zhuang, and Z. Shi, Clustering in extreme learning machine feature space, Neurocomputing (2013)
2. Z. Li, X. Wu, H. Peng, Nonnegative matrix factorization on orthogonal subspace. Pattern Recogn. Lett. **31**(9), 905–911 (2010)
3. S. B, S. L, Nonnegative matrix factorization clustering on multiple manifolds, in *AAAI* (2010)
4. S. Li, X. Hou, H. Zhang, Q. Cheng, Learning spatially localized, parts-based representation, Computer Vision and Pattern Recognition, 2001. CVPR 2001. in *Proceedings of the 2001 IEEE Computer Society Conference*, vol. 1, pp. 207–212, 2001
5. W. Xu, X. Liu, Y. Gong, Document clustering based on non-negative matrix factorization, in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 267–273, 2003
6. G.-B. Huang, L. Chen, Convex ncremental extreme learning machine. Neurocomputing **70**, 3056–3062 (2007)
7. G.-B. Huang, L. Chen, A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. IEEE Trans. Neural Networks **71**, 3460–3468 (2008)
8. Q. Liu, Q. He, and Z. Shi, Extreme support vector machine classifer, in *Knowledge Discovery and Data Mining* (Springer, Berlin, 2008), pp. 222–233
9. G. Huang, X. Ding, H. Zhou, Optimization method based extreme learning machine for classifcation. Neurocomputing **74**(1), 155–163 (2010)
10. N. X, Locality preserving projections. Neural Inf. proc. syst. **16**, 153 (2004)
11. D. Cai, X. He, J. Han, Graph regularized nonnegative matrix factorization for data representation. IEEE Trans. Pattern Anal. Mach. Intell. **33**(8), 1548–1560 (2011)
12. G.B. Huang, O.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications. Neurocomputing **70**(1), 489–501 (2006)
13. G.B. Huang, D.H. Wang, Y. Lan, Extreme learning machines: a survey. Int. J. Mach. Learn. Cybern. **2**(1), 107–122 (2011)
14. G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans. Neural Networks **17**(4), 879–892 (2006)

15. G. Zhou, A. Cichocki, S. Xie, Fast nonnegative matrix/tensor factorization based on low-rank approximation. IEEE Trans. Signal Process. **60**(6), 2928–2940 (2012)
16. P.M. Rossini, A.T. Barker, A. Berardelli, Non-invasive electrical and magnetic stimulation of the brain and spinal cord and roots: basic principles and procedures for routine clinical application. Electroencephalogr. Clin. Neurophysiol. Suppl. **91**(2), 79–92 (1994)
17. S. Nikitidis, A. Tefas, N. Nikolaidis, Subclass discriminant nonnegative matrix factorization for facial image analysis. Pattern Recognit. **45**(12), 4080–4091 (2012)
18. W. Y and Z. Y., "Non-negative matrix factorization: a comprehensive review", Pattern Recognition, vol. 1, no. 1, 2011.
19. Z. Luo, N. Guan, D. Tao, Non-negative patch alignment framework. IEEE Trans. Neural Networks **22**(8), 1218–1230 (2011)
20. D. Cai, X. He, J. Han, Document clustering using locality preserving indexing. IEEE Trans. Knowl. Data Eng. **17**(12), 1624–1637 (2005)

Extreme Support Vector Regression

Wentao Zhu, Jun Miao and Laiyun Qing

Abstract Extreme Support Vector Machine (ESVM), a variant of ELM, is a nonlinear SVM algorithm based on regularized least squares optimization. In this chapter, a regression algorithm, Extreme Support Vector Regression (ESVR), is proposed based on ESVM. Experiments show that, ESVR has a better generalization ability than the traditional ELM. Furthermore, ESVM can reach comparable accuracy as SVR and LS-SVR, but has much faster learning speed.

Keywords Extreme learning machine · Support vector regression · Extreme support vector machine · Extreme support vector regression · Regression

1 Introduction

Extreme Learning Machine (ELM) is a great successful algorithm for both classification and regression. It has the good generalization ability at an extremely fast learning speed [1]. Moreover, ELM can overcome some challenging issues that other machine learning algorithms face [1]. Some desirable advantages can be found in ELM such as, extremely fast learning speed, less human intervene and great computational scalability. The essence of ELM is that the hidden layer parameters need not be tuned iteratively and the output weights can be simply calculated by least square optimization [2, 3]. Extreme Learning Machine (ELM) has attracted a great number of researchers and engineers [4–8] recently.

W. Zhu · J. Miao
Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China

L. Qing (✉)
School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China
e-mail: lyqing@ucas.ac.cn

F. Sun et al. (eds.), *Extreme Learning Machines 2013: Algorithms and Applications*, Adaptation, Learning, and Optimization 16, DOI: 10.1007/978-3-319-04741-6_3,

Extreme Support Vector Machine (ESVM), a kind of single hidden layer feed forward network, has the same extremely fast learning speed, but it has a better generalization ability than ELM [9] on classification tasks. ESVM, a special form of Regularization Network (RN) derived from Support Vector Machine (SVM), has the same advantages as ELM such as, that hidden layer parameter can be randomly generated [9]. Due to these ideal properties, many researches have been conducted on ESVM [10–13]. In fact, ESVM is a variant of ELM. However, ESVM in [9] cannot be applied to regression tasks.

In this chapter, Extreme Support Vector Regression (ESVR) algorithm was proposed for regression. Our ESVR algorithm is based on the ESVM model and the essential of ELM for regression is utilized. Some comparison experiments show that the ESVR algorithm has quite good generalization ability and the learning speed of ESVR is quite large.

This chapter is organized as follows. ELM and ESVM are briefly reviewed in Sect. 2. The linear ESVR, nonlinear ESVR are proposed in Sect. 3. Performances of ESVR compared with ELM, SVR and LS-SVR are verified in Sect. 4.

2 Extreme Support Vector Machine

We here briefly introduce the basic concept of ELM and Extreme Support Vector Machine (ESVM). ELM can reach not only the smallest training errors, but also the best generalization ability [14]. ESVM is based on regularization least squares in the feature space. The performance of ESVM is better than ELM on classification tasks [9].

2.1 Extreme Learning Machine

ELM is a single hidden layer forward network (SLFNs). The parameters of the hidden layer can be randomly generated, and need not be iteratively tuned [2, 3]. The least square optimization process tackles the output weight vector [2, 3]. Therefore, the learning speed of ELM is extremely fast. Moreover, ELM has the unified algorithm to tackle classification and regression problems.

For N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i) \in (\mathbf{R}^d \times \mathbf{R}^m)$, where $\mathbf{x}_i$ is the extracted feature vector, and $\mathbf{t}_i$ is the target output. For the SLFNs, the mathematical model with L hidden nodes is

$$\sum_{i=1}^{L} \beta_i g_i(\mathbf{x}_j) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j) = \widehat{\mathbf{t}}_j, j = 1, ..., N, \tag{1}$$

where $\widehat{\mathbf{t}}_j$ is the output of the SLFNs, and $G(\mathbf{a}_i, b_i, \mathbf{x}_j)$ is the hidden layer feature mapping. According to [3], the hidden layer parameters $(\mathbf{a}_i, b_i)$ can be randomly generated.

The goal of ELM is to approximate the expected targets by the above predicted targets. That is,

$$\|\mathbf{H}\widehat{\beta} - \mathbf{t}\| = \min_{\beta} \|\mathbf{H}\beta - \mathbf{t}\|, \tag{2}$$

where

$$\mathbf{H} = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_N) \end{bmatrix}_{N \times L}, \beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m}, \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m}.$$

Therefore, the least square method can be used to solve the above optimization problem. That is to say, the output weight β can be obtained by the following equation.

$$\widehat{\beta} = \mathbf{H}^{\dagger}\mathbf{T}, \tag{3}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of matrix $\mathbf{H}$ [15].

From the above discussion, ELM can be implemented by the following steps. First, randomly generate hidden node parameters $(\mathbf{a}_i, b_i), i = 1, ..., L$, where L is the parameter of ELM denoting the number of hidden nodes. Second, calculate the hidden layer mapped feature matrix $\mathbf{H}$ as the above equation. Third, calculate the output weight by the least square optimization.

2.2 Extreme Support Vector Machine

Instead of using kernels to represent data features by SVM, ESVM explicitly utilizes SLFNs to map the input data points into a feature space [9]. ESVM is a variant of ELM [16]. The essential of ESVM is a kind of regularization network. Similar to ELM, ESVM has a number of advantages, such as, fast learning speed, good generalization ability and fewer human intervene.

The model of ESVM can be obtained by replacing the inequality constraint in the traditional SVM with the equality constraint [9].

$$\begin{aligned} &\min_{(\mathbf{w},r,\mathbf{y}) \in \mathbf{R}^{\tilde{n}+1+m}} \frac{\nu}{2}\|\mathbf{y}\|^2 + \frac{1}{2}\left\| \begin{bmatrix} \mathbf{w} \\ r \end{bmatrix} \right\|^2 \\ &s.t. \quad D(\Phi(A)\mathbf{w} - r\mathbf{e}) + \mathbf{y} = \mathbf{e} \end{aligned} \tag{4}$$

In the above equation, $\Phi(\mathbf{x}) : \mathbf{R}^n \rightarrow \mathbf{R}^{\tilde{n}}$ is the feature mapping function in the hidden layer of SLFNs. $\mathbf{y}$ is the slack variable of the model. ν is the tradeoff parameter between allowable errors and the minimization of weights, and $\mathbf{e}$ is a vector of size $m \times 1$ which is filled with 1s, where m is the number of the samples. D is the diagonal matrix of the element of 1 or -1 denoting the labels. A is the sample data matrix.

After deduction, the solution of the model is simply equivalent to calculating the following expression according to [9]:

$$\begin{bmatrix} \mathbf{w} \\ r \end{bmatrix} = \left(\frac{\mathbf{I}}{\nu} + {E_\Phi}^T E_\Phi \right)^{-1} {E_\Phi}^T D\mathbf{e}, \tag{5}$$

where $E_\Phi = [\Phi(A), -\mathbf{e}] \in \mathbf{R}^{m \times (\tilde{n}+1)}$.

ESVM can reach better generalization ability than ELM almost in all classification tasks [9]. Due to the simple solution, ESVM can learn at an extremely fast speed. Additionally, the activation functions can be explicitly constructed. However, diagonal label matrix D must be constructed in the above ESVM model and D must be with the element of 1 or -1 in the above deduction, which means that the ESVM model cannot be applied to multi-class classification or regression tasks directly.

3 Extreme Support Vector Regression

In this section, we will extend ESVM from classification tasks to regression tasks. The linear and nonlinear extreme support vector regression will be proposed.

3.1 The Linear Extreme Support Vector Regression

Our model is derived from the formulation of ESVM. Similar to ESVM, ESVR also replaces the inequality constraint of the ϵ-SV regression with the equality constraint [17]. But different from ESVM, the diagonal target output matrix need not be constructed. The model of ESVR is constructed as follows.

$$\begin{aligned} \min_{(\mathbf{w},r,\mathbf{y}) \in \mathbf{R}^{\tilde{n}+1+m}} & \frac{\nu}{2}\|\mathbf{y}\|^2 + \frac{1}{2}(\mathbf{w}^T\mathbf{w} + r^2) \\ s.t. \quad & A\mathbf{w} - r\mathbf{e} - \mathbf{T} = \mathbf{y} \end{aligned}, \tag{6}$$

where $\mathbf{T}$ is the expected target output of the sample data matrix A.

We will provide the solution of the above ESVR model. If $\mathbf{w}$, r have been obtained, the test process is to calculate $\mathbf{x}^T\mathbf{w} - r$ to get the output target of the sample. Nonlinear ESVR also will be supplied by introducing a nonlinear feature mapping function in the following section.

3.2 The Nonlinear Extreme Support Vector Regression

Nonlinear ESVR can be obtained by simply replace the original data matrix A by the transformed matrix $\Phi(A)$.

$$\begin{aligned} \min_{(\mathbf{w},r,\mathbf{y})\in\mathbf{R}^{\tilde{n}+1+m}} & \frac{\nu}{2}\|\mathbf{y}\|^2 + \frac{1}{2}\left\|\begin{bmatrix}\mathbf{w}\\ r\end{bmatrix}\right\|^2 \\ s.t. \quad & \Phi(A)\mathbf{w} - r\mathbf{e} - \mathbf{T} = \mathbf{y} \end{aligned} \tag{7}$$

After deduction, analytical solution can be obtained.
If $m < \tilde{n} + 1$, we can obtain a simple analytical solution of $\mathbf{w}$ and r.

$$\begin{bmatrix}\mathbf{w}\\ r\end{bmatrix} = {E_\Phi}^T\mathbf{s} = {E_\Phi}^T\left(\frac{\mathbf{I}}{\nu} + E_\Phi {E_\Phi}^T\right)^{-1}\mathbf{T} \tag{8}$$

If $m > \tilde{n} + 1$,

$$\begin{bmatrix}\mathbf{w}\\ r\end{bmatrix} = {E_\Phi}^T\mathbf{s} = \left(\frac{\mathbf{I}}{\nu} + {E_\Phi}^T E_\Phi\right)^{-1}{E_\Phi}^T\mathbf{T}, \tag{9}$$

where $E_\Phi = [\Phi(A), -\mathbf{e}] \in \mathbf{R}^{m\times(\tilde{n}+1)}$.

From the above discussion, the algorithm of ESVR can be explicitly concluded as follows. First, randomly generate hidden layer parameters and choose an activation function. $\Phi(A)$ can be obtained. Second, construct the matrix $E_\Phi = [\Phi(A), -\mathbf{e}]$. Third, choose some positive parameters ν to calculate $\begin{bmatrix}\mathbf{w}\\ r\end{bmatrix}$ by expression (8) or (9). When a new instance $\mathbf{x}$ comes, we can use $\Phi(\mathbf{x})^T\mathbf{w} - r$ to predict it.

3.3 The Essence of ESVR

Inspired by support vector theory in SVM, ESVR is an proximal algorithm of SVR. Intuitively, we replace the inequality constraints in ϵ-SV regression with equality constraints. The following equation is the ϵ-SV regression constraints formula [17, 18].

$$\begin{aligned} T_i - \langle w, x_i\rangle + r &\le \epsilon + y_i \\ \langle w, x_i\rangle - r - T_i &\le \epsilon + y_i^\star \\ y_i, y_i^\star &\ge 0 \end{aligned} \tag{10}$$

Actually, the replacement is a proximal method and proximal decision plane is obtained in the ESVR.

After deduction, the analytical solution of ESVR is quite similar to that of ELM. Compared to the algorithm of ELM, ESVR is similar to regularized ELM besides a biased term. However, the generalization performance of ESVR is better than that of ELM, SVR and LS-SVR. The technique used in ESVR is quite important for overcoming ill-pose problems and singular problems that traditional ELM may encounter [19]. Furthermore, ESVR has the desirable features as that of ELM such as, fast learning speed, fewer human intervene. From the computation view, ESVR is a variant of ELM. Such random parameters are utilized in the ESVR. ESVR has the similar form of that of regularized ELM.

4 Performance Verification

In this section, the performance of ESVR is compared with ELM, SVR and LS-SVR on some benchmark regression problems data sets.

4.1 Experimental Conditions

All the simulations for ESVR, ELM, SVR and LS-SVR for regression algorithms were carried out in MATLAB R2010a environment running in a Xeon E7520, 1.87GHZ CPU. The codes used for ELM, SVR and LS-SVR were downloaded from [1], [2], and [3] respectively.

In order to extensively verify the performance of ESVR, ELM, SVR and LS-SVR, twelve data sets of different sizes and dimensions were downloaded from UC Irvine Machine Learning Repository [4] or StatLib library [5] for simulation. These data sets can be divided into three categories according to different sizes and feature dimensions. Baskball, Strike, Cloud, and Autoprice are of small size and low dimensions. Pyrim, Housing, Body fat, and Cleveland are of small size and medium dimensions. Balloon, Quake, Space-ga, and Abalone are of large size and low dimensions. Table 1 lists some features of the regression data sets in our simulation.

In the experiments, three fold cross validation was conducted to select parameters. The best parameters ν of ESVR, the cost factor C and kernel parameter γ of SVR, LS-SVR were obtained from the candidate sequence $2^{-25}, 2^{-24}, \ldots, 2^{23}, 2^{24}, 2^{25}$. The number of hidden layer nodes $\tilde{n}$ in ESVR was obtained from [10, 300] with step 10. The average performance of testing Root Mean Square Errors (RMSE) was conducted as the evaluation metric to select the best parameters. And all the data

[1] http://www.ntu.edu.sg/eee/icis/cv/egbhuang.html

[2] http://asi.insarouen.fr/enseignants/arakotom/toolbox/index.html

[3] http://www.esat.kuleuven.be/sista/lssvmlab/

[4] http://archieve.ics.uci.edu/ml/

[5] http://lib.stat.cmu.edu/

Table 1 Specification of regression problems

Datasets	# Attributes	# Training data	# Testing data
Baskball	4	64	32
Cloud	9	72	36
Autoprice	9	106	53
Strike	6	416	209
Pyrim	27	49	25
Body fat	14	168	84
Cleveland	13	202	102
Housing	13	337	169
Balloon	2	1334	667
Quake	3	1452	726
Space-ga	6	2071	1036
Abalone	8	2784	1393

sets were normalized into $[-1, 1]$ before the regression process. The kernel function used in the experiments was the RBF function. The activation function of ESVR was sigmoidal function.

4.2 Performance Comparison on Benchmark Datasets

Comparisons of generalization performance between ESVR and ELM on the above twelve different benchmark regression data sets were firstly carried out. Nonlinear models with sigmoidal additive feature map function were used for comparison. Ten round experiments of the same parameters were conducted to obtain an average performance evaluation in each fold due to randomly selecting parameters in the hidden layer. Figure 1 is the testing RMSE of ESVR and ELM with different numbers of hidden nodes on six of the twelve real world data sets.

Figure 1 shows the testing RMSE of ESVR is lower than that of ELM. We can observe that the performance of ELM is varied greatly with the number of hidden nodes as well. Moreover, the standard deviation of ELM is much larger than that of ESVR. The result of the experiment reveals that the generalization of ESVR is better than that of ELM. Furthermore, ESVR is more stable than ELM from Fig. 1, because the slack variable added can make our model more stable in the ESVR.

The second experiment was conducted to compare the performances of ESVR, SVR and LS-SVR. In this experiment, performances of ESVR algorithm were validated compared with SVR and LS-SVR. The same kernel function (RBF function) was used for SVR and LS-SVR. The activation function of ESVR was sigmoidal function. Through three fold cross validation, the best parameters (C, γ) or $(\nu, \tilde{n})$ were obtained. Table 2 records parameters of different models on different data sets.

Table 3 is the performance results of ESVR, SVR and LS-SVR. Training time and testing RMSE were recoded as the learning speed and generalization ability of the model separately. The best results for different data sets were emphasized into bold face.

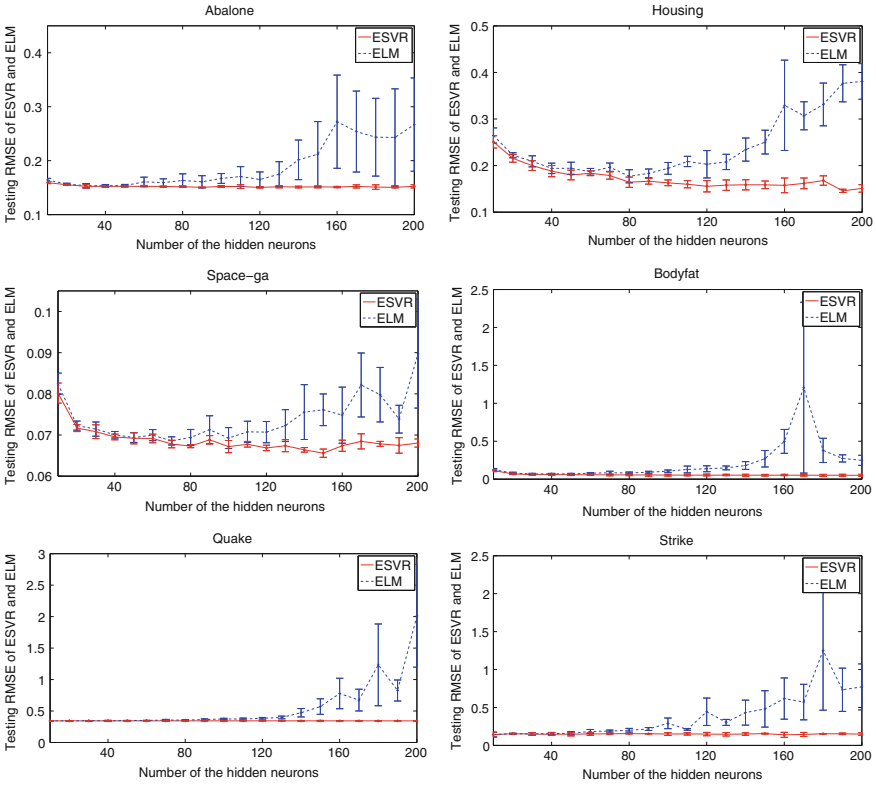

Fig. 1 Testing RMSE of ESVR and ELM

Table 3 shows that the testing RMSE of ESVR is the lowest in most of the data sets. The training time of ESVR is much less than that of SVR and LS-SVR especially in the large scale data instances. These results reveal that, ESVR has comparable generalization ability than that of SVR and LS-SVR. Furthermore, the average learning speed of ESVR can reach at least three times of that of LS-SVR, and at least ten times of that of SVR on the above real world benchmark data sets. The reason that ESVR is much faster is the same as that why ELM has an extremely fast learning speed. The solution of ESVR is an analytical equation. The learning process is simply to solve an least square expression.

5 Conclusions

This chapter studies the ESVM algorithm and proposes a new regression algorithm ESVR. Similar to ESVM, ESVR is a new nonlinear SVM algorithm based on regularized least squares and it is also a variant of ELM algorithm. ESVR not only can be

Table 2 Parameters of ESVR, SVR and LS-SVR

Algorithms	SVR		LS-SVR		ESVR	
	C	γ	C	γ	ν	$\tilde{n}$
Baskball	2^{10}	2^{2}	2^{0}	2^{2}	2^{4}	250
Cloud	2^{20}	2^{7}	2^{25}	2^{16}	2^{3}	170
Autoprice	2^{19}	2^{5}	2^{9}	2^{7}	2^{5}	290
Strike	2^{-3}	2^{-2}	2^{-1}	2^{-1}	2^{1}	250
Pyrim	2^{0}	2^{2}	2^{3}	2^{3}	2^{-1}	260
Body fat	2^{6}	2^{3}	2^{9}	2^{7}	2^{2}	300
Cleveland	2^{22}	2^{13}	2^{22}	2^{25}	2^{-5}	240
Housing	2^{6}	2^{1}	2^{6}	2^{3}	2^{7}	280
Balloon	2^{3}	2^{1}	2^{25}	2^{5}	2^{25}	260
Quake	2^{1}	2^{-12}	2^{-1}	2^{-15}	2^{0}	40
Space-ga	2^{3}	2^{-1}	2^{11}	2^{3}	2^{19}	300
Abalone	2^{-1}	2^{-1}	2^{2}	2^{2}	2^{9}	150

Table 3 Performance comparisons of SVR, LS-SVR and ELM

Algorithms	SVR		LS-SVR		ESVR	
	Testing RMSE	Training time (s)	Testing RMSE	Training time (s)	Testing RMSE	Training time (s)
Baskball	0.2567	0.1029	0.2568	0.0049	**0.2521**	0.0208
Cloud	0.1729	0.0774	0.1810	0.0065	**0.1582**	0.0115
Autoprice	0.1381	0.1328	**0.1359**	0.0072	0.1561	0.0365
Strike	**0.1443**	0.9707	0.1472	0.0541	0.1497	0.0641
Pyrim	**0.2151**	0.0336	0.2159	0.0051	0.2184	0.0240
Body fat	0.0514	0.0485	**0.0502**	0.0128	0.0506	0.0458
Cleveland	**0.4267**	0.2690	0.4333	0.0147	0.4279	0.0365
Housing	0.1469	0.7729	0.1458	0.0455	**0.1409**	0.0771
Balloon	0.0242	7.8253	0.0099	1.0798	**0.0098**	0.1932
Quake	0.3438	205.7426	**0.3425**	2.4292	0.3440	0.0146
Space-ga	**0.0654**	92.1705	0.0665	2.5293	0.0661	0.3677
Abalone	0.1519	250.4772	**0.1486**	9.6423	0.1510	0.1875

used to regression tasks, but also can be applied to classification tasks. Performances of ESVR are compared with that of ELM, SVR and LS-SVR. ESVR has a little better generation ability than ELM. Compared to SVR and LS-SVR, ESVR has a comparable generalization ability, but has the much faster learning speed.

Acknowledgments The authors would like to thank Mr. Zhiguo Ma and Mr. Fuqiang Chen for their valuable comments. This research is partially sponsored by National Basic Research Program of China (No. 2009CB320900), and Natural Science Foundation of China (Nos. 61070116, 61070149, 61001108, 61175115, and 61272320), Beijing Natural Science Foundation (No. 4102013), President Fund of Graduate University of Chinese Academy of Sciences (No.Y35101CY00), and Importation and Development of High-Caliber Talents Project of Beijing Municipal Institutions.

References

1. G.-B. Huang, D.H. Wang, Y. Lan, Extreme learning machines: a survey. Int. J. Mach. Learn. Cybernet. **2**(2), 107–122 (2011)
2. G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in *Proceedings. 2004 IEEE International Joint Conference on Neural Networks, 2004*, vol. 2, pp. 985–990, IEEE, 2004
3. G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications. Neurocomputing. **70**(1), 489–501 (2006)
4. W. Zong, H. Zhou, G.-B. Huang, Z. Lin, Face recognition based on kernelized extreme learning machine, in *Autonomous and Intelligent Systems* (Springer, Berlin, 2011) pp. 263–272
5. H.-J. Rong, Y.-S. Ong, A.-H. Tan, Z. Zhu, A fast pruned-extreme learning machine for classification problem. Neurocomputing. **72**(1), 359–366 (2008)
6. M. van Heeswijk, Y. Miche, T. Lindh-Knuutila, P.A. Hilbers, T. Honkela, E. Oja, A. Lendasse, Adaptive ensemble models of extreme learning machines for time series prediction, in *Artificial Neural Networks—ICANN 2009* (Springer, Berlin, 2009) pp. 305–314
7. G.-B. Huang, L. Chen, Convex incremental extreme learning machine. Neurocomputing. **70**(16), 3056–3062 (2007)
8. Q. He, T. Shang, F. Zhuang, Z. Shi, Parallel extreme learning machine for regression based on mapreduce. Neurocomputing. **102**, 52–58 (2013)
9. Q. Liu, Q. He, Z. Shi, Extreme support vector machine classifier, in *Advances in Knowledge Discovery and Data Mining* (Springer, Berlin, 2008) pp. 222–233
10. Q. He, C. Du, Q. Wang, F. Zhuang, Z. Shi, A parallel incremental extreme svm classifier. Neurocomputing. **74**(16), 2532–2540 (2011)
11. B. Frénay, M. Verleysen, Using SVMs with randomised feature spaces: an extreme learning approach, in *Proceedings of the 18th European symposium on Artificial Neural Networks (ESANN), Bruges, Belgium*, pp. 28–30 (2010)
12. A. Subasi, A decision support system for diagnosis of neuromuscular disorders using DWT and evolutionary support vector machines. Signal Image Video Process. **7**, 1–10 (2013)
13. P.-F. Pai, M.-F. Hsu, An enhanced support vector machines model for classification and rule generation, in *Computational Optimization, Methods and Algorithms* (Springer, Berlin 2011) pp. 241–258
14. G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification. IEEE Trans. Syst. Man Cybern. Part B Cybern. **42**(2), 513–529 (2012)
15. C.R. Rao, S.K. Mitra, *Generalized Inverse of a Matrix and Its Applications* (Wiley, New York, 1971)
16. G.-B. Huang, X. Ding, H. Zhou, Optimization method based extreme learning machine for classification. Neurocomputing. **74**(1), 155–163 (2010)
17. C. Cortes, V. Vapnik, Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
18. A.J. Smola, B. Schölkopf, A tutorial on support vector regression. Stat. Comput. **14**(3), 199–222 (2004)
19. N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans. Neural Networks. **17**(6), 1411–1423 (2006)

A Modular Prediction Mechanism Based on Sequential Extreme Learning Machine with Application to Real-Time Tidal Prediction

Jian-Chuan Yin, Guo-Shuai Li and Jiang-Qiang Hu

Abstract Neural networks have been proved to be efficient for online identification and prediction of complex nonlinear systems. However, for systems with time-varying dynamics which are common in practice, networks achieved by holistic learning scheme cannot reflect the time-varying local dynamics of system. In this study, a modular prediction scheme is proposed by combining the mechanism model with a neural network predictive model which is online acquired by a sequential learning extreme learning machine (ELM) based on a sliding data window (SDW). The SDW-based ELM (SDW-ELM) is online constructed by learning samples in the real-time updated SDW, is suitable for online identification and prediction of time-varying system dynamics. Tidal prediction is essential for marine safety and efficiency, but changes of tidal level is a typical time-varying system which varies not only with the revolutions of celestial bodies but with the environmental influences such as atmospheric pressure, wind, rainfall and ice. The harmonic analysis method is used to represent the influences of celestial bodies, while the SDW-ELM is used to represent the influences of meteorological factors and other unmodeled factors. Therefore, the proposed modular based on SDW-ELM is applied for real-time tidal level prediction based on measurement data in Port Hardy. Simulation results demonstrate the effectiveness and efficiency of the proposed algorithm and results are compared with that by online sequential ELM (OS-ELM) algorithm and SDW-ELM.

Keywords Extreme learning machine · Sliding data window · Time-varying dynamics · Modular prediction · Tidal prediction

J.-C. Yin (✉) · G.-S. Li · J.-Q. Hu
Navigation College, Dalian Maritime University, Linghai Rd. 1, Dalian 116026, China
e-mail: yinjianchuan@dlmu.edu.cn

G.-S. Li
e-mail: liguoshuai@dlmu.edu.cn

J.-Q. Hu
e-mail: hujiangqiang@126.com

F. Sun et al. (eds.), *Extreme Learning Machines 2013: Algorithms and Applications*, Adaptation, Learning, and Optimization 16, DOI: 10.1007/978-3-319-04741-6_4,

1 Introduction

Tidal prediction is an important issue in areas of marine safety, coastal construction design, tidal energy utilization, ocean natural calamities prevention and military affairs [1]. Precise tidal prediction is vital for the ship operation planning, such as stipulating ship schedule of navigating through shallow waters or under bridge. Under such conditions, accurate water depth under keel or clearance under bridge can be considered to ensure marine safety based on precise real-time tidal level predictions. The conventional harmonic analysis approach, in which the tide can be expressed as the superposition of several sinusoidal constituents, is the most commonly used tidal prediction approach and still the basis for long-term tidal prediction [2]. Location and revolution of celestial bodies such as moon and sun, as well as coastal topography like coastline shape and sea floor profile, are foundation for stipulating annual tidal table for seafarers. Tidal level is also influenced by meteorological factors such as atmospheric pressure, wind, rainfall and ice. In addition, other time-varying factors like river discharge also affect tidal level at estuary sites. However, these influences cannot be reflected in tidal table, which is the foundation for seafarers to make voyage planning [3]. Therefore, environmental influences would be ignored even if they may cause accidents such as grounding on rock. Furthermore, these meteorological factors are time-varying in nature, which is hard to be represented by strictly founded mathematical model. Therefore, there is a need to construct an adaptive model whose structure and parameters can adapt to the above-mentioned time-varying environmental changes.

Artificial intelligent (AI) techniques have shown their power in nonlinear computation, and have been widely applied in coastal and marine engineering [4]. Among various AI techniques, artificial neural network (ANN) can learn and represent complex mapping underlying measured data directly, and stores the knowledge within computational neurons and connecting weights [5]. Attributing to natures such as inherent nonlinearity, universal approximation capability and parallel information processing mechanism, ANN has been implemented successfully in prediction of different type of tides [3, 6]. Chang and Lin presents a neural network model of simulating tides at multi-points considering tide-generating forces [7]; Günaydın focuses on the prediction of monthly mean significant wave heights from meteorological data by using both artificial neural network (ANN) and regression methods [8]; Huang et al. presents a regional neural network for water level (RNNWL) prediction method; [9]; Lee proposed a back-propagation neural network mode which is efficiently for short-term and long-term tidal level predictions [2]; Lee also proposed neural network model for storm surge predictions which using four input factors, including the wind velocity, wind direction, pressure and harmonic analysis tidal level [10]; Liang et al. incorporate in the neural network the non-astronomical meteorological components for tidal level predictions under weathers such as typhoon and storm surge [3]; Rajasekaran et al. applied the functional networks (FN) and sequential learning neural network (SLNN) approaches for tidal predictions using short-term observation [11]; Tseng et al. developed a typhoon-surge forecasting model with a

back-propagation neural network in which the factors of typhoons characteristics, local meteorological conditions and typhoon surges at considered tidal station are both incorporated [12]; Yin et al. proposed a sequential learning algorithm to construct variable structure radial basis function network for real-time tidal prediction [13]. However, the structures of the commonly implemented neural networks are static, which cannot represent the time-varying system dynamics which varies with environmental changes mentioned above. Furthermore, networks with static dimension will inevitably result in phenomenon of over-fitting or under-fitting in processing data arriving sequentially, both would deteriorate the generalization capability of resulted network.

Sequential learning algorithms are designed for time-varying system dynamics by constructing ANN with variable structure [14]. The algorithms generate variable structure neural networks whose hidden neurons are added or pruned at each step upon learning samples sequentially, and the parameters are tuned accordingly. The sequential learning algorithm is originated from resource allocation network (RAN) algorithm [14], which learn samples sequentially and adding neurons accordingly at each step; Lu improves RAN by incorporating pruning strategy in the learning process, the resulted minimal RAN (MRAN) adding or pruning the neurons adaptively during learning [15]. Unlike conventional neural network theories and implementations, Huang proposes a extremely fast learning algorithm for feedforward neural network referred to as extreme learning machine (ELM) whose performance has been evaluated on a number of benchmark problems [16, 17]. Based on ELM, Liang et al. proposed the online sequential extreme learning machine (OS-ELM), improved ELM from batch learning to be able to handle data which arrives sequentially or chunk-by-chunk with varying chunk size [18]. OS-ELM has been applied in varies areas and simulation results indicate that it produces better generalization performance with lower training time, comparing with other sequential learning algorithms such as resource allocation network (RAN) and its extensions [15, 19, 20].

Sliding data window (SDW), which is frequently used for representing time-varying dynamics, has been adopted in various areas such as signal processing [21] and control [22]. In order to obtain accurate solutions in current situation and to avoid overflow in orthogonal decomposition of the oldest measurement, Luo and Billings forms a sliding data window to represent current system dynamics [23]. Akpan and Hassapiswe use a sliding stack window to store a short history of the training patterns, and implement this continuously updated stack for neural network training [22]. However, the model achieved by learning samples in sliding window can only represent local system dynamics and cannot reflect the global characteristics of system and may cause model instability. It is straightforward to take both advantages of global and local model by combining them together, the result modular structure can achieve better approximation and prediction accuracy with satisfying stability [24]. ELM has shown its computational power in various applications. Therefore, in this chapter, a modular neural network is proposed to construct a better inputCoutput mapping both locally and globally, by combining the mechanism model with a ELM which is online constructed based on a sliding data window.

To represent and predict the influence of various time-varying environmental factors on tidal changes, the proposed modular prediction approach is implemented for real-time tidal level prediction. The model is composed of the mechanism module which is performed by conventional harmonic tidal prediction method, and a neural network module which is realized by ELM based on SDW (SDW-ELM) which is real-time updated. The SDW-ELM adjusts network structure and connecting parameters by learning samples in the sliding window sequentially, and makes predictions simultaneously. The proposed neural prediction model was applied to real-time tidal level prediction at Port Hardy, a west coast of Canada, to validate its feasibility and effectiveness.

2 Sequential ELM Based on Sliding Data Window (SDW-ELM)

2.1 Sliding Data Window

Real-time construction of neural network by sequential learning is a research focus in recent years [25]. In a sequential learning scheme, if an algorithm learns only the latest received single sample at one step, the resulted network would be highly affected by the particular sample and may result in instability of network; while if there are too much samples, such as all the historically received samples, to be learned at one step, the resulted network cannot reflect the real-time changes in system dynamics, and the computational burden will be increased accordingly. To make a compromise, a sliding data window (SDW) is employed in the present study to represent the input-output mapping of system dynamics, and the algorithm learns samples in the window.

In the proposed learning scheme, samples are presented to the sliding window sequentially. The hidden neurons are determined by ELM algorithm. As the ELM only need to process the data in the sliding data window with fixed size, the computational burden is highly reduced comparing with OS-ELM algorithm. Once the hidden neurons are determined, the connecting weights between the hidden layer and output layer are adjusted accordingly. The most important feature of the resulting network is that the network can focus on the current dynamics of system which is represented by the sliding data window, thus the variable structure network can capture the real-time changes in system dynamics.

The SDW is implemented for representing the current dynamics of system. SDW is a first-in-first-out (FIFO) sequence: when the newly received sample slides in the window, the foremost one will be removed simultaneously. At time of t, the sliding data window is described as:

$$W_{SD} = [(x_{t-N+1}, y_{t-N+1}), \ldots, (x_t, y_t)], \tag{1}$$

where W_{SD} denotes sliding data window, N is the width of the sliding data window. x and y are inputs and outputs of the sliding data window with dimension of $R^{n\times N}$ and $R^{m\times N}$, with n and m are dimensions of input and output, respectively.

At time of $t+1$, the SDW is updated by substituting the t in (2) with $t+1$:

$$W_{SD} = [(x_{t-N+2}, y_{t-N+2}), \ldots, (x_{t+1}, y_{t+1})], \tag{2}$$

with the width of the window remain unchanged.

2.2 Single Hidden Layer Feedforward Neural Networks (SLFNs)

Assume we have N arbitrary distinct samples $(\mathbf{x}_k, \mathbf{t}_k) \in \mathbf{R}^n \times \mathbf{R}^m$, where $\mathbf{x}_k$ is an input vector and $\mathbf{t}_k$ is the corresponding desired output. A standard single hidden layer feedforward networks (SLFNs) with $\tilde{N}$ additive hidden nodes and activation function $G(\mathbf{x})$ can be represented by

$$f_{\tilde{N}}(\mathbf{x}_k) = \sum_{i=1}^{\tilde{N}} \boldsymbol{\omega}_i G(\mathbf{a}_i \cdot \mathbf{x}_k + b_i), \quad k = 1, \ldots, N. \tag{3}$$

where $\mathbf{a}_i = [a_{1i}, \cdots, a_{ni}]^T$ is the weight vector connecting the input layer to the ith hidden node, b_i is the bias of the ith hidden node, and $\mathbf{a}_i \cdot \mathbf{x}_k$ denotes the inner product of vectors $\mathbf{a}_i$ and $\mathbf{x}_k$ in $\mathbf{R}^n$. The activation functions $G(\mathbf{x})$ are sigmoids. For notational simplicity, here the scalar output case is considered. Extension to the multioutput case is straightforward. In fact, multioutput SLFNs can always be separated into a group of single output SLFNs. A schematic of the SLFNs with the scalar output is depicted in Fig. 1.

The ultimate purpose of SLFNs is to find out the values of ω_i, $\mathbf{a}_i$ and b_i such that $\sum_{k=1}^{N} \|f_{\tilde{N}}(\mathbf{x}_k) - t_k\| = 0$, or

$$f_{\tilde{N}}(\mathbf{x}_k) = \sum_{i=1}^{\tilde{N}} \omega_i G(\mathbf{a}_i \cdot \mathbf{x}_k + b_i) = t_k, \quad k = 1, \ldots, N. \tag{4}$$

Then, Eq. (4) can be written compactly as

$$\mathbf{H}\boldsymbol{\omega} = \mathbf{T}, \tag{5}$$

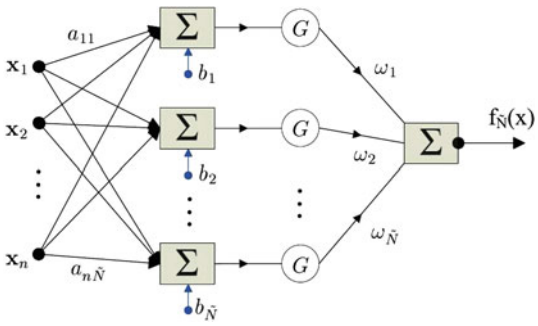

Fig. 1 The schematic of the SLFNs with the scalar output

where

$$\mathbf{H}(\mathbf{a}_1, \ldots, \mathbf{a}_{\tilde{N}}, b_1, \ldots, b_{\tilde{N}}, \mathbf{x}_1, \ldots, \mathbf{x}_N) = \begin{bmatrix} G(\mathbf{a}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & G(\mathbf{a}_{\tilde{N}} \cdot \mathbf{x}_1 + b_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ G(\mathbf{a}_1 \cdot \mathbf{x}_N + b_1) & \cdots & G(\mathbf{a}_{\tilde{N}} \cdot \mathbf{x}_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}, \quad (6)$$

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_{\tilde{N}} \end{bmatrix} \text{ and } \mathbf{T} = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}. \quad (7)$$

$\mathbf{H}$ is called the hidden layer output matrix of the network, the ith column of $\mathbf{H}$ is the ith hidden node's output vector with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ and the kth row of $\mathbf{H}$ is the output vector of the hidden layer with respect to input $\mathbf{x}_k$.

2.3 Extreme Learning Machine

The main idea of ELM is that for N arbitrary distinct samples $(\mathbf{x}_k, t_k)$ in order to obtain arbitrarily small non-zero training error, one may randomly generate $\tilde{N}(\leq N)$ hidden nodes (with random parameters $\mathbf{a}_i$ and b_i). Under this assumption, $\mathbf{H}$ is completely defined. Then, Eq. (5) becomes a linear system and the output weights $\boldsymbol{\omega}$ are estimated as

$$\hat{\boldsymbol{\omega}} = \mathbf{H}^{\dagger}\mathbf{T} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{T}, \quad (8)$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of the hidden layer output matrix $\mathbf{H}$. Calculation of the output weights is done in a single step here. Thus this avoids any lengthy training procedure to choose control parameters (learning rate and learning epochs, etc.). Universal approximation capability of ELM has been analyzed

in [16], which indicated that SLFNs with randomly generated additive or RBF nodes can universally approximate any continuous target function on any compact subspace of $\mathbf{R}_n$. Besides, in the implementations of ELM, the activation functions for additive nodes can be any bounded nonconstant piecewise continuous functions and the activation functions for RBF nodes can be any integrable piecewise continuous functions.

ELM Algorithm. Given a training set $\aleph = \{(\mathbf{x}_k, t_k)|\mathbf{x}_k \in \mathbf{R}^n, t_k \in R, k = 1, \ldots, N\}$, activation function G, and hidden node number $\tilde{N}$.

Step 1. Randomly assign hidden node parameters $(\mathbf{a}_i, b_i)$, $i = 1, \ldots, \tilde{N}$.

Step 2. Calculate the hidden layer output matrix $\mathbf{H}$.

Step 3. Calculate the output weight $\tilde{\omega} : \tilde{\omega} = \mathbf{H}^{\dagger}\mathbf{T}$.

2.4 Extreme Learning Machine Based on Sliding Data Window

The conventional ELM theory can be implemented within the learning of data in SDW similarly. That is, for N arbitrary distinct samples $(x_{t-N+1}, y_{t-N+1}), \ldots, (x_t, y_t)$ in the sliding window, in order to obtain arbitrarily small non-zero identification error, one may randomly generate $\tilde{N}$, $\tilde{N} \leq N$ hidden nodes with random parameters. Under this assumption, response matrix H is completely defined and H is the response matrix of hidden nodes with respect to the SDW as inputs. Then, $Hw = Y$ becomes a linear system and the output weights ω are estimated according to (8) where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of the hidden layer response matrix $\mathbf{H}$. It is noted that the calculation of the output weights is done in a single step here thus avoids any lengthy repetitive training procedure to choose control parameters such as learning rate and learning epochs, etc.. Universal approximation capability of ELM has been analyzed [16], which indicates that single layer feedforward networks (SLFNs) with randomly generated additive or radial basis function (RBF) nodes can universally approximate any continuous target function on any compact subspace of $\mathbf{R}^N$. Besides, in the implementations of ELM, the activation functions for additive nodes can be any bounded nonconstant piecewise continuous functions and the activation functions for RBF nodes can be any integrable piecewise continuous functions.

The learning process and prediction process are conducted at each step. In this study, the conventional autoregressive (AR) model is adopted. For arbitrary l-step-ahead prediction, at t-th step, the N data pairs for learning is:

$$y(t) = f(y(t-l), y(t-l-1), \ldots, y(t-l-N+1)) \tag{9}$$

where N is width of SDW, $y(t-l), y(t-l-1), \ldots, y(t-l-N+1)$ is input and $y(t)$ is output. Once the structure and parameters is determined, the achieved ELM is implemented for l-step-ahead prediction at each step:

$$y(t+l) = f(y(t), y(t-1), \ldots, y(t-N+1)) \tag{10}$$

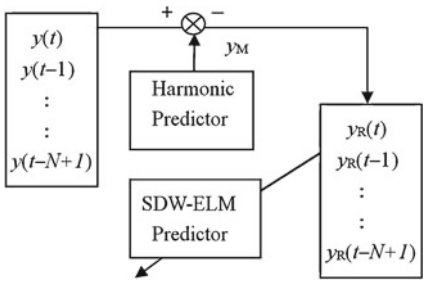

Fig. 2 The leaning process of the modular prediction model based on SDW-ELM

where N is width of SDW, $y(t), y(t-1), \ldots, y(t-N+1)$ is input and $y(t+l)$ is prediction output.

3 Modular Sequential ELM Based on Sliding Data Window

The parametric model and non-parametric model are suitable for representing the static and time-varying dynamics of system, respectively. A straightforward approach is to combine the mechanism module and neural network module into one ensemble system of presumably better quality and treat the combined output as the final forecast [26]. Modular neural networks can incorporate priori knowledge, thus possess merits such as robustness and parsimonious model structure.

In this study, conventional harmonic method is implemented to predict the periodical tidal changes which is driven by the revolution of celestial bodies; the ELM based on sliding data window is implemented to represent and predict the residual of the actual tidal level comparing to the harmonic model. The residual can be considered to be caused by time-varying meteorological or hydrological factors such as air pressure, wind, rainfall or river flow. Predictions generated by the two modules are combined together to form the final prediction model. That is, the two identification module is combined in series connection to form one ship motion forecast describing the current dynamics of tidal changes. The learning process configuration of the proposed combined modular is illustrated as Fig. 2.

The learning process is conducted in two steps. Firstly, the harmonic method is implemented to give tidal level predictions, and the prediction of y is denoted as y_M. y_R is the residual between $y(t)$ and y_M. y_R is composed of $y_R(t), \ldots, y_R(t-n_y)$, with n_y is the order of the autoregressive (AR) model. The residual information is considered as the effects of time-varying environmental changes. To train ELM, we set $y_R(t-1), \ldots, y_R(t-n_y)$ as input and $y_R(t)$ as output. After the ELM is constructed by learning data pairs of y_R in SDW, the achieved SDW-ELM module is combined with the harmonic method and form the modular prediction model. At each step, once the SDW-ELM is trained, $y_R(t), \ldots, y_R(t-n_y+1)$ is then set as input and the $y_R(t+1)$ would be the prediction of the influence of environment to the tidal levels.

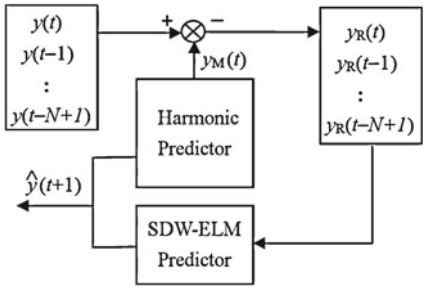

Fig. 3 The prediction process of the modular prediction model based on SDW-ELM

The result is combined with the result by the harmonic method and get the final tidal prediction result. The prediction process is illustrated in Fig. 3.

4 Online Tidal Level Prediction Performance

4.1 Tidal Level Prediction by Harmonic Method

As the conventionally used tidal level prediction method, harmonic method takes consider of the effects of celestial bodies and can give stable long-term predictions. Therefore, in this study, the harmonic method is employed in the modular mechanism. The result of harmonic prediction is a superposition of many constituents whose amplitudes and frequencies are determined by local analysis based on the long-term tidal measurements. Thus, the tidal level can be predicted as a time-dependant function:

$$h(t) = a_0 + \sum_{i=1}^{n} h_i \, cos(\omega_i t - \phi_i) + \varepsilon_i \tag{11}$$

where a_0 is the mean sea level (MSL); n is the number of constituents; h_i, ω_i and ϕ_i are the amplitude, frequency and phase of the corresponding constituent, respectively. And ε_i is the unmolded error and will be modeled by the RBFN constructed by SDW-ELM.

Measured tidal level data of Port Hardy, a west coast port of Canada, is implemented to evaluate the feasibility and effectiveness of the proposed SDW-ELM and modular prediction model based on SDW-ELM. Figure 4 shows the tidal data ranges from GMT0000 April 1 to GMT2300 April 30, 2013. All the measurement data of Port Hardy in this study are achieved from web site http://www.pac.dfo-mpo.gc.ca. The conventional harmonic tidal prediction result is also shown in Fig. 4.

It is shown in Fig. 4 that the tidal level values predicted by harmonic method diverse much from measured ones at some point, which means that there are

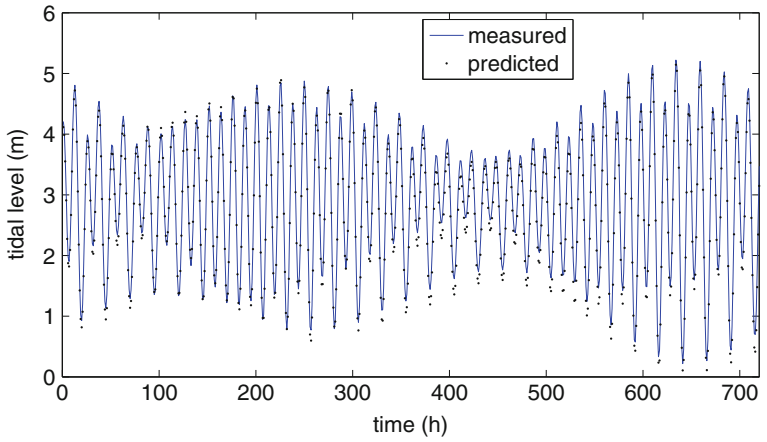

Fig. 4 The measured tidal level of Port Hardy and predicted results with harmonic method

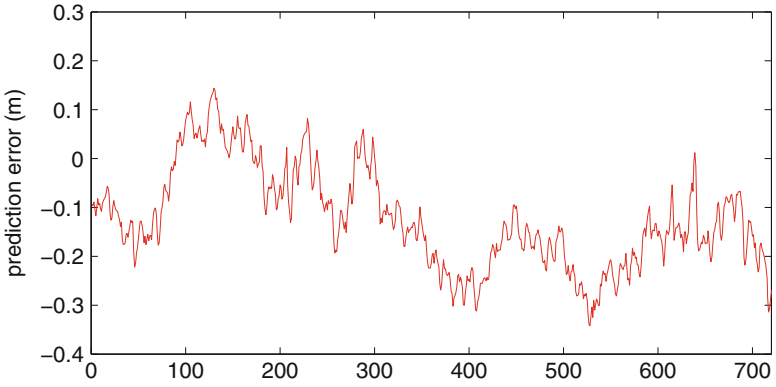

Fig. 5 Prediction error of tidal level using harmonic method

prediction error by harmonic method. The reason of diverse lies in many aspects, part of them is due the fact that the approach of harmonic method only take consider in factors of celestial bodies, whereas does not consider time-varying environmental factors such as wind, air pressure, ice, rainfall, etc. When the severe weather conditions occur such as storm surge or abrupt barometer pressure changes, the influence may cause a diverge to 1m from ordinary tidal level, which may cause harm to the coastal construction, marine transportation or people's lives in coastal areas.

The residual of the prediction of harmonic method is shown in Fig. 5.

It can be seen in Fig. 5 that the residual of conventional harmonic tidal prediction method display the characteristics of time-varying which may attribute to environmental influences. Since the environmental changes cannot be reflected by harmonic

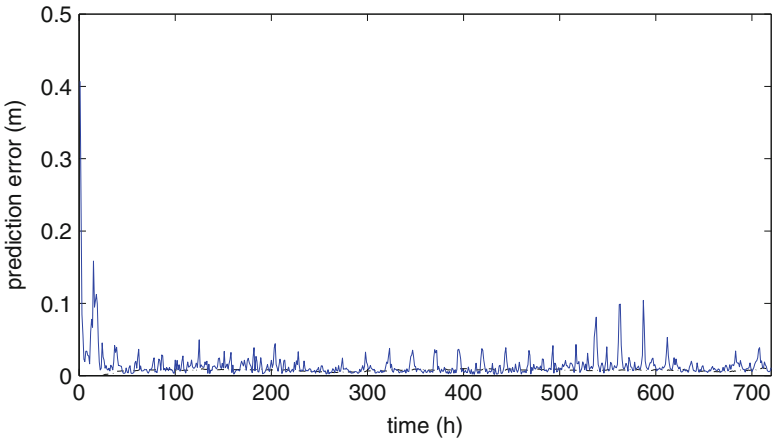

Fig. 6 1-step-ahead prediction error of tidal level using SDW-ELM

method, the proposed SDW-ELM and modular prediction based on SDW-ELM are implemented to represent the time-varying environmental influences.

4.2 Tidal Level Prediction by SDW-ELM

The SDW-ELM method is implemented for online tidal prediction based on the measured tidal level data of Port Hardy. The sample interval of the data is 1 h. To evaluate the performance of SDW-ELM on a long range, 720 steps of simulation is conducted. The parameters of SDW-ELM is: the width of SDW, N, is set 72, that is the tidal level data over 3 days; the number of hidden neurons assigned to the ELM is 24. 720 steps of simulation is conducted and altogether 50 times of simulation is conducted and the average prediction error of 1-step-ahead tidal level prediction is shown in Fig. 6.

The average running time for each step is 0.0024 s. The average identification root mean square error (RMSE) over 50 times is 0.0072 m and the average of predictive mean absolute error (MAE) is 0.0149 m. To evaluate the performance of the SDW-ELM, OS-ELM is also implemented based on the same measurement data and Simulation result is depicted in Fig. 7. For OS-ELM, the number of assigned hidden units is 24. The average running time for each step is 0.0045 s, the average identification RMSE and prediction MAE over 50 times is 0.0123 and 0.0152 m, respectively.

It can be seen from comparison that the processing speed of SDW-ELM is twice higher than that of OS-ELM method. This is because that the SDW-ELM only need to handling the data in the sliding window which much less than all the received samples in OS-ELM. The learning RMSE of OS-ELM is smaller than that of SDW-ELM but

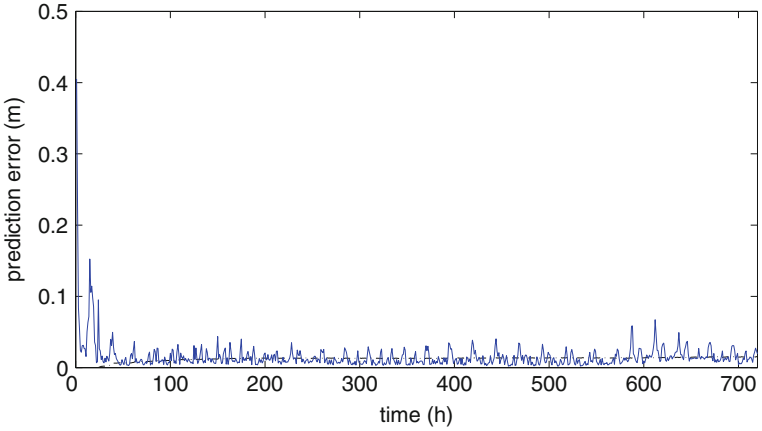

Fig. 7 1-step-ahead prediction error of tidal level using OS-ELM

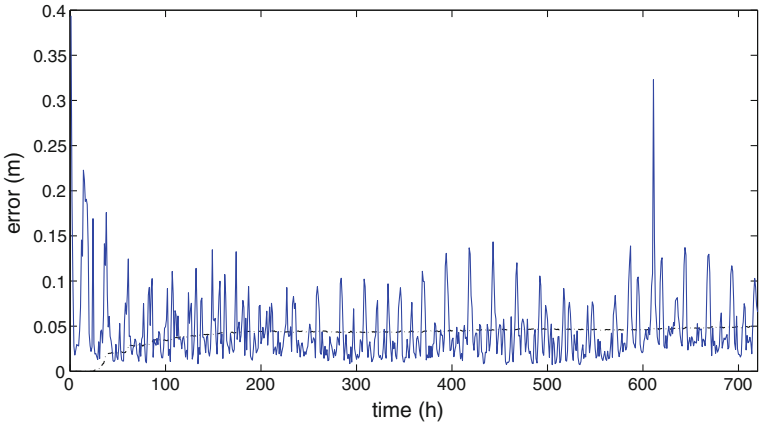

Fig. 8 2-step-ahead identification and prediction errors of tidal level using OS-ELM

the prediction accuracy of SDW-ELM is slightly higher than that of OS-ELM. That means that the performance of SDW-ELM overwhelm OS-ELM for 1-step-ahead tidal prediction for short term prediction.

Conventionally the prediction accuracy declines with the increase of prediction horizon. To evaluate the prediction performance for a longer time domain, the simulation results of 2-h-ahead prediction and 3-h-ahead prediction using the OS-ELM and SDW-ELM methods are shown in Figs. 8, 9, 10 and 11, respectively. In Figs. 8, 9, 10 and 11, both identification error and prediction error are depicted in the figures, with the black dotted line denote the identification error and the blue solid line denote the prediction error.

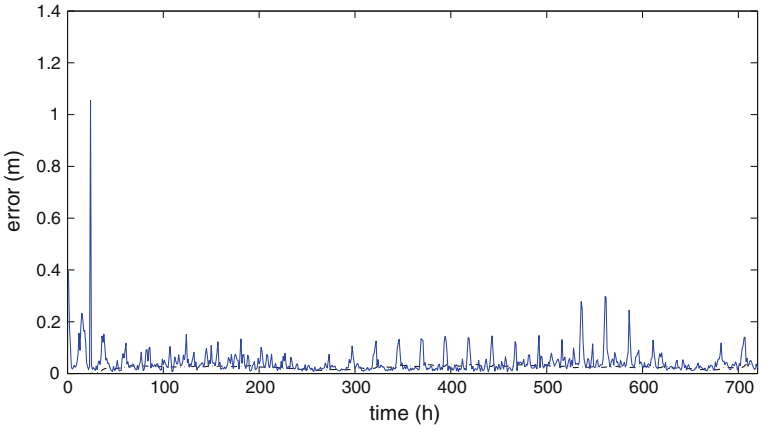

Fig. 9 2-step-ahead identification and prediction error of tidal level using SDW-ELM

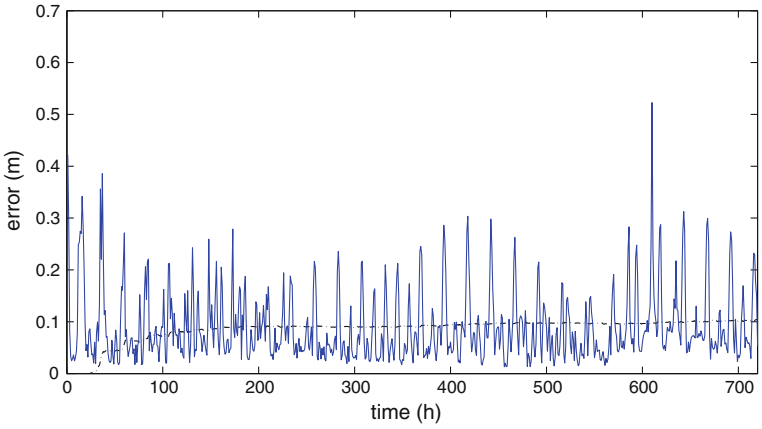

Fig. 10 3-step-ahead identification and prediction errors of tidal level using OS-ELM

It is shown from Figs. 8 and 9 that performance of SDW-ELM overwhelms that of the OS-ELM method, which also demonstrate the effectiveness of SDW in representing time-varying dynamics. The same conclusion can be drawn by comparison between Figs. 10 and 11. It is also find that the identification error and prediction error both increases with the increase of prediction horizon, which can be noted by comparing Fig. 8 with Fig. 10, and Fig. 9 with Fig. 11, respectively.

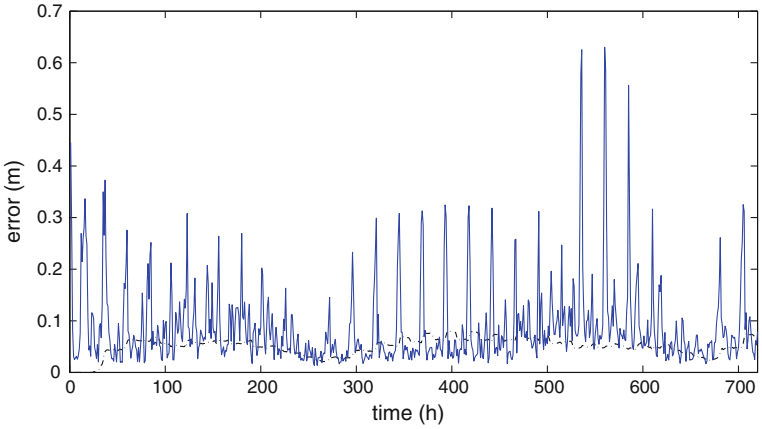

Fig. 11 3-step-ahead identification and prediction error of tidal level using SDW-ELM

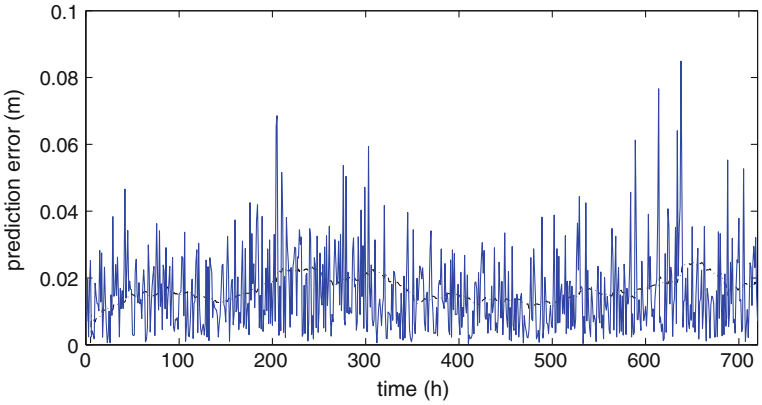

Fig. 12 1-step-ahead prediction error of tidal level using modular model based on SDW-ELM

4.3 Tidal Level Prediction by Modular Prediction based on SDW-ELM

However, as analyzed above, the SDW-ELM focus on the local dynamics whereas the OS-ELM can represent the global dynamics. With the prolonged prediction horizon, the performance of OS-ELM overwhelm that of SDW-ELM as the local dynamics which is reflected by SDW-ELM is a time-varying process, so the achieved SDW-ELM is suitable for short-term prediction and not suitable for long-term predictions.

Simulation result of average prediction and identification error over 50 times are shown in Fig. 12.

The average processing time of each step is 0.0012 s, the average identification RMSE is 0.0163 m and average prediction MAE are 0.0142 m, respectively.

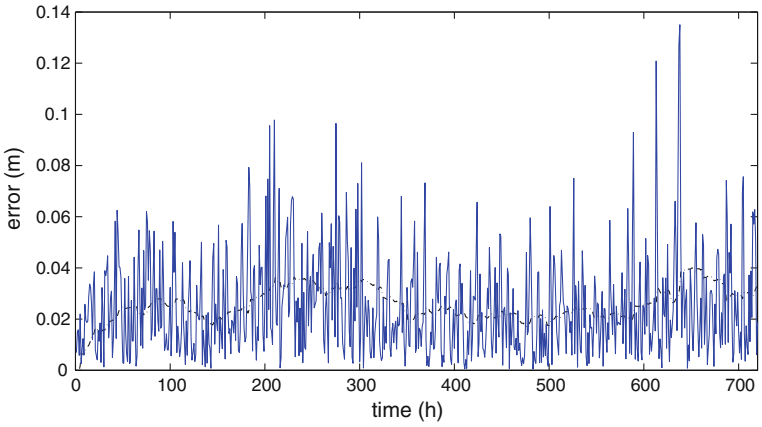

Fig. 13 2-step-ahead identification and prediction errors of proposed modular method

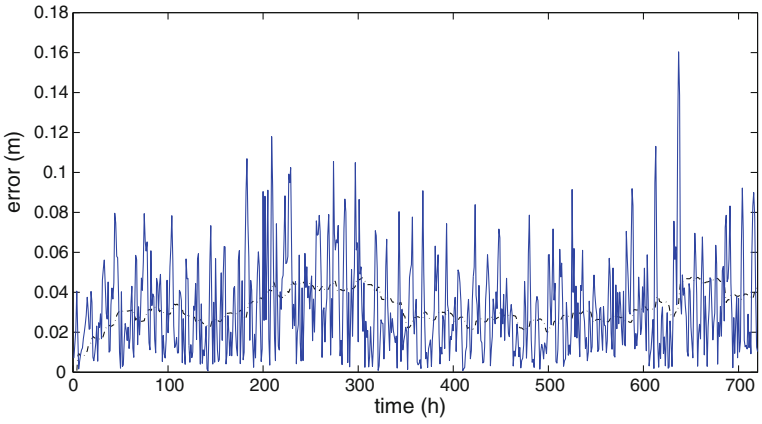

Fig. 14 3-step-ahead identification and prediction errors of proposed modular method

The simulation results for 2-step-ahead and 3-step-ahead predictions are simulated using proposed modular method and the results are depicted in Figs. 13 and 14.

For two-hours-ahead prediction, the average identification RMSE and prediction MAE becomes 0.0256 and 0.0248 m, which is much smaller than that of the SDW-ELM and OS-ELM. It is noticed from simulation results that the prediction accuracy is improved by combining the mechanism model and neural network model especially under conditions of long-term prediction. That is, the holistic dynamics caused by celestial body is represented by harmonic method, and the SDW-ELM only need to focus on the prediction residual of harmonic method, which is caused by time-varying meteorological, hydrological and other factors.

To depict the prediction performance of the OS-ELM and SDW-ELM in a longer time domain, simulations are conducted over the prediction horizon of 12 h and

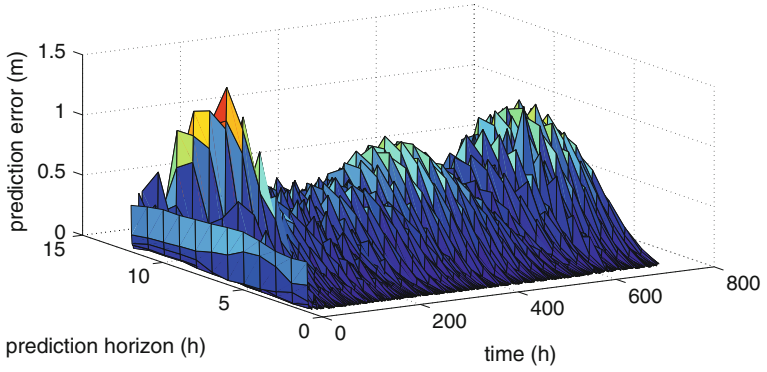

Fig. 15 Prediction error of tidal level over prediction horizon of 12 h using OS-ELM

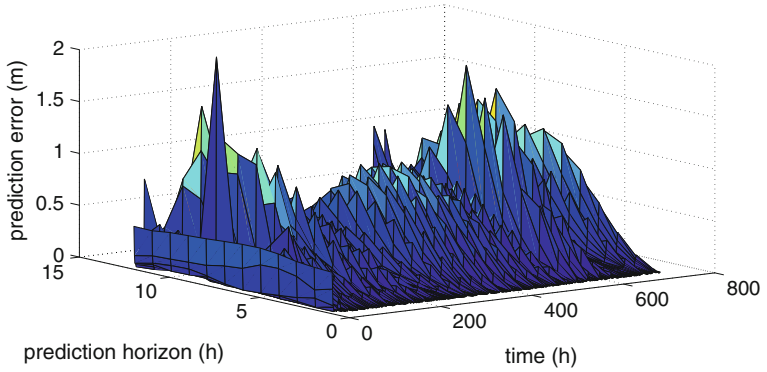

Fig. 16 Prediction error of tidal level over prediction horizon of 12 h using SDW-ELM

the results of 1-h-ahead to 12-h-ahead prediction are shown in Figs. 15 and 16, respectively.

It can be noticed from Figs. 15 and 16 that the prediction with SDW-ELM varies violently with time and prediction horizon, which means it only represent the local system dynamics and will inevitably bring instability under circumstance of long-term prediction. The OS-ELM method performs more stable than SDW-ELM method, but it incorporate the information of all the received samples and cannot represent the time-varying system dynamics. The same problem exists in the conventional harmonic method. It means that all the above-mentioned methods are not suitable for precise long-time tidal prediction.

It is straight forward to combine the models which reflects local dynamics and global dynamic respectively. In this study, the harmonic tidal prediction model, the most popular mechanism model is implemented for representing the holistic characteristics of tidal changes, and the SDW-ELM is implemented for representing local dynamics by using the prediction residual by harmonic method. The sum of the

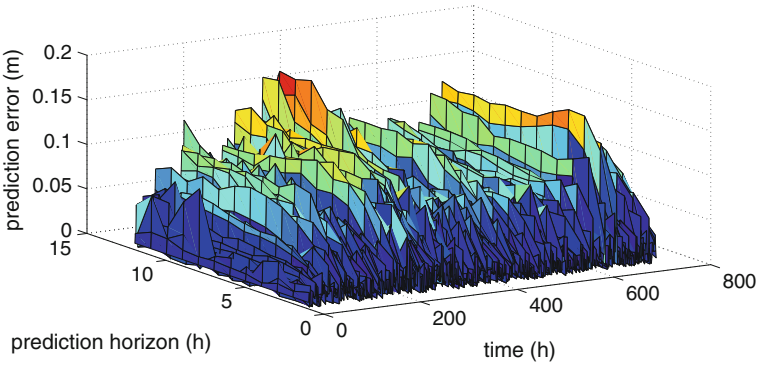

Fig. 17 Prediction error of tidal level over prediction horizon of 12 h using modular model based on SDW-ELM

Table 1 Tidal level prediction simulation results of Port Hardy

Prediction methods	SDW-ELM		OS-ELM		Modular prediction	
Simulation results	$RMSE_{Iden}$	MAE_{Pred}	$RMSE_{Iden}$	MAE_{Pred}	$RMSE_{Iden}$	MAE_{Pred}
1-h-ahead	0.0072	0.0149	0.0123	0.0152	0.0163	0.0142
2-h-ahead	0.0240	0.0436	0.0412	0.0452	0.0256	0.0248
3-h-ahead	0.0504	0.0870	0.0861	0.0899	0.0318	0.0322
6-h-ahead	0.1433	0.2340	0.2425	0.2370	0.0369	0.0403
12-h-ahead	0.0668	0.1250	0.1362	0.1404	0.0369	0.0427
24-h-ahead	0.0401	0.0818	0.1194	0.1320	0.0359	0.0442

prediction results by two prediction is the results of the modular prediction model. The tidal prediction simulation of 1-step-ahead prediction using the modular prediction model. The window width is 24 and the assigned number of hidden units is 4. Simulations of online tidal level prediction over prediction horizon of 12 h are conducted by using modular prediction model based on SDW-ELM and results are shown in Fig. 17.

It can be seen from Fig. 17 that the proposed modular prediction model demonstrates much higher prediction accuracy than the harmonic method, SDW-ELM model and OS-ELM. Even under condition of 12-h-ahead prediction, the maximum prediction error below 0.16 m which can satisfy the need of seafarers for stipulating voyage plans. The high prediction accuracy modular model is based on the strictly founded physical model of harmonic method and the SDW-ELM which can precisely represent the time-varying dynamics of tidal changes.

The changes of tide is a periodic process, and it may also influence the identification and prediction accuracies of many algorithms. The average identification RMSE ($RMSE_{Iden}$) and prediction MAE (MAE_{Pred}) of 720 steps of prediction over 50 times of simulation are listed in Table 1, under circumstances of 1, 2, 3, 6, 12 and 24 h ahead prediction, respectively.

It is shown in the table that under condition of short-term prediction (1, 2 and 3 steps ahead), all the method possesses satisfying identification and prediction

performance, but the OS-ELM and modular model possesses faster processing speed owing to the limited number of samples needed to be processed. However, it is interesting to notice that for methods of SDW-ELM and OS-ELM, there are larger identification and prediction errors under condition of 6-h-ahead prediction, whereas the performance is much improved under conditions of 12 and 24 h ahead predictions. It is the periodicity that causes this phenomenon. As the period of the tide is about 12 h and 24 min, the characters of current tide dynamics is like the dynamics of tide 12 and 24 h before. It can also be noticed from Table 1 that the modular method are affected little by the periodicity because the periodicity mainly caused by celestial bodies has been represented by mechanism harmonic prediction module, so the SDW-ELM module can concentrate on the identification and prediction of the time-varying effects caused by environmental disturbances such as meteorological and hydrological factors.

5 Conclusions

To fit the need in areas of navigation efficiency and safety, this chapter proposes an accurate modular tidal prediction model. The modular prediction model combines the strictly founded mechanism tidal prediction model of harmonic method with a sequential learning ELM whose hidden units and connecting parameters can be adjusted based on the learning of data in a sliding data window. The model takes both advantages of global prediction and local prediction, the prediction accuracy is highly improved. And the processing speed is faster than that of OS-ELM method cause it only need to process the limited number of data in the sliding data window. The tidal prediction simulation is conducted base on the measured data of Port Hardy and the results demonstrate the feasibility and effectiveness of the proposed modular method.

Acknowledgments This work is supported by the National Natural Science Foundation of China (Grant No: 51279106), the Fundamental Research Funds for the Central Universities of China (Grant Nos.:2012QN004 & 2012QN002) and Applied Basic Research Fund of the Chinese Ministry of Transport.

References

1. G.-H. Fang, W.-Z. Zheng, Z.-Y. Chen, *Analysis and Prediction of Tide and Tidal Current* (Ocean Press, Beijing, 1986)
2. T. Lee, Back-propagation neural network for long-term tidal predictions. Ocean Eng. **31**, 225–238 (2004)
3. S. Liang, M. Li, Z. Sun, Prediction models for tidal level including strong meteorologic effects using a neural network. Ocean Eng. **35**, 666–675 (2008)
4. I. Malekmohamadi, M. Bazargan-Lar, R. Kerachian et al., Evaluating the efficacy of SVMs, BNs, ANNs and ANFIS in wave height prediction. Ocean Eng. **38**, 487–497 (2011)

5. S. Haykin, *Neural Networks: A Comprehensive Foundation* (Prentice Hall, New Jersey, 1999)
6. T. Lee, D. Jeng, Application of artificial neural networks in tide-forecasting. Ocean Eng. **29**, 1003–1022 (2002)
7. H.-K. Chang, L.C. Lin, Multi-point tidal prediction using artificial neural network with tide-generating forces. Coastal Eng. **53**, 857–864 (2006)
8. K. Günaydın, The estimation of monthly mean significant wave heights by using artificial neural network and regression methods. Ocean Eng. **35**(14–15), 1406–1415 (2008)
9. W. Huang, C. MUrray, N. Kraus, J. Rosati, Development of a regional neural network for coastal water level predictions. Ocean Eng. **30**, 2275–2295 (2003)
10. T. Lee, Neural network prediction of a storm surge. Ocean Eng. **33**, 483–494 (2006)
11. S. Rajasekaran, K. Thiruvenkatasamy, T. Lee, Tidal level forecasting using functional and sequential learning neural networks. Appl. Math. Model. **30**, 85–103 (2006)
12. C. Tseng, C. Jan, J. Wang, C. Wang, Application of artificial neural networks in typhoon surge forecasting. Ocean Eng. **34**, 1757–1768 (2007)
13. J.-C. Yin, Z.-J. Zou, F. Xu, Sequential learning radial basis function network for real-time tidal level predictions. Ocean Eng. **57**, 49–55 (2013)
14. J. Platt, A resource allocating network for function interpolation. Neural Comput. **3**(2), 213–225 (1991)
15. Y.-W. Lu, N. Sundararajan, P. Saratchandran, A sequential learning scheme for function approximation using minimal radial basis function neural networks. Neural Comput. **9**, 461–478 (1997)
16. G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications. Neurocomputing **70**, 489–501 (2006)
17. G.-B. Huang, D.-H. Wang, Y. Lan, Extreme learning machines: a survey. Int. J. Mach. Learn. Cybern. **2**(2), 107–122 (2011)
18. N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans. Neural Netw. **17**(6), 1411–1423 (2006)
19. G.-B. Huang, P. Saratchandran, N. Sundararajan, An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks. IEEE Trans. Syst. Man Cybern. **34**(6), 2284–2292 (2004)
20. G.-B. Huang, P. Saratchandran, N. Sundararajan, A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. IEEE Trans. Neural Netw **16**(1), 57–67 (2005)
21. J.-M. Li, X.-F. Chen, Z.-J. He, Adaptive stochastic resonance method for impact signal detection based on sliding window. Mech. Syst. Signal Process. **36**, 240–255 (2013)
22. V.A. Akpan, G.D. Hassapis, Nonlinear model identification and adaptive model predictive control using neural networks. ISA Trans. **50**, 177–194 (2011)
23. W. Luo, S.A. Billings, Adaptive model selection and estimation for nonlinear systems using a sliding data window. Signal Process. **46**, 179–202 (1995)
24. N. Jiang, Z.-Y. Zhao, L.-Q. Ren, Design of structural modular neural networks with genetic algorithm. Adv. Eng. Softw. **34**, 17–24 (2003)
25. S. Suresh, K. Dong, H. Kim, A sequential learning algorithm for self-adaptive resource allocation network classifier. Neurocomputing **7**(3), 3012–3019 (2010)
26. L. Kuntcheva, *Combining Pattern Classifiers-Methods and Algorithms* (Wiley, New York, 2004)

An Improved Weight Optimization and Cholesky Decomposition Based Regularized Extreme Learning Machine for Gene Expression Data Classification

ShaSha Wei, HuiJuan Lu, Yi Lu and MingYi Wang

Abstract The gene expression data classification problem has been widely studied due to the development of DNA microarray technology. However, how to classify the complex gene expression data accurately still remains as a major problem. In this chapter, an improved Regularized Extreme Learning Machine (RELM) method is proposed for gene expression data classification. The new training algorithm, called COW-RELM, is based on weight optimization and Cholesky decomposition. In the proposed method, the input weights of RELM are optimized based on genetic algorithm in which the fitness function is defined as the reciprocal of error function. To accelerate the speed of the algorithm, the output weights matrix is optimized based on Cholesky decomposition. The experiments of COW-RELM algorithm have been conducted on the Breast, Leukemia, Colon, Heart and other gene expression data. The results are thus presented to show the excellent performance and effectiveness of the classification accuracy.

Keywords Regularized extreme learning machine · Weight optimization · Cholesky decomposition · Gene expression data

1 Introduction

With the wide application of Microarray technology, a growing number of gene expression data is used to study the gene functions, as well as the relationship between specific genes and certain disease [1]. Gene expression data is obtained through DNA

S. Wei · H. Lu (✉) · M. Wang
Hangzhou 310018, China
e-mail: hjlu@cjlu.edu.cn; huijuanlu29@gmail.com

Y. Lu
Computer Science Department, Prairie View A&M University, Prairie View, TX 77446, USA

F. Sun et al. (eds.), *Extreme Learning Machines 2013: Algorithms and Applications*, Adaptation, Learning, and Optimization 16, DOI: 10.1007/978-3-319-04741-6_5,

microarray hybridization test after pretreatment, which usually represents in the form of matrix [2].

Data classification [3] is used to divide genes into different groups according to the similarity or pattern of gene expression data. In 1999, Alon firstly classified the colon cancer data set into multi-groups using hierarchical clustering algorithm [4]. Gloub [5] constructed the classifier with nearest neighbor algorithm to predict the classification of leukemia. Khan and Narayanan obtained the classification model by the known samples through artificial neural network [6]. Furey and Lu realized the classification of gene expression data respectively by using support vector machine (SVM) and compressed sensing technology [7, 8].

In 2006, Huang proposed an original algorithm called extreme learning machine (ELM) in [9]. This method makes the selection of the weights of the hidden neurons very fast. Hence, compared to some classical methods, the overall computational time for model structure selection and actual training of the model is often saved a lot. Furthermore, the algorithm remains rather simple, which makes its implementation easy. However, there are inherent limitations in ELM. Studies show [10] that in most cases ELM has high performance, but hidden layer initial parameters (connection weights, the offset value, the number of nodes) of ELM have big impacts on classification accuracy. Huang proposed I-ELM which increases hidden layer node of ELM one by one to improve the convergence rate. In 2008, Huang proposed another algorithm EI-ELM which can produce a more compact network structure and learn faster [11, 12]. The above-mentioned ELMs do not take into account the structural risks that may lead to overfitting problems. In 2010, Deng [13] proposed a Regularized Extreme Learning Machine (RELM) which incorporates the structural risk minimization theory and the weighted least squares method into the ELM. RELM has better generalization performance that not only minimizes the training error, but also makes the edge distance maximized. Further, it has certain anti-jamming capability for outliers.

The input weights and hidden layer of RELM are randomly assigned and the changes of the hidden layer output matrix of RELM may be very large [14, 15]. This in turn results in large changes of the output weight matrix, which greatly increases both empirical risk and structural risk, and degenerates the robustness.

In this chapter, we studied the RELM with both input weights and output weights matrix for gene expression data classifications [13–15], and develop a new training algorithm, called COW-RELM. The RELM input weights are optimized based on Genetic Algorithm (GA). The fitness function of GA is defined as the reciprocal of error function. When the number of samples is relatively large, the training speed of the output weights solving progress is slow. Optimizing the output weights matrix with the Cholesky decomposition method can improve the training speed and reduce training time. The proposed algorithm has been applied to the Breast, Leukemia, Colon, Heart and other gene expression data, the experimental results show significant improvement in classification results.

2 RELM

According to statistical theory in [16], the actual risks include empirical and structural risks. RELM considers these two factors at the same time through the parameter γ to adjust the proportion. The mathematical model of RELM can be expressed as:

$$\min(\frac{1}{2}\|\beta\|^2 + \frac{\gamma}{2}\|\varepsilon\|^2) \tag{1}$$

$$\text{Subject to} \sum_{i=1}^{N} \beta_i g\left(a_i x_j + b_i\right) - t_j = \varepsilon_j \tag{2}$$

where ε is the matrix of errors between the reference feature vectors and the feature vectors generated by the hidden layer of RELM in [17]. γ is the proportion of two kinds of risk parameters. $\|\beta\|^2$ is used to smooth the cost function at the singular point of the correlation matrix of the feature vectors to avoid the ill-posed inverse of the data matrix and improve the robustness of RELM with respect to the noisy environment in [13, 14, 17].

The Eq. (3) which is converted to an unconditional extremum problem by Lagrange function, is a conditional extremum one. The following Lagrange function:

$$\begin{aligned}(\beta\varepsilon\alpha) &= \frac{\gamma}{2}\|\varepsilon\|^2 + \frac{1}{2}\|\beta\|^2 - \sum_{j=1}^{N} \alpha_j \left(g\left(a_i x_j + b_i\right) - t_j - \varepsilon_j\right) \\ &= \frac{\gamma}{2}\|\varepsilon\|^2 + \frac{1}{2}\|\beta\|^2 - \alpha\left(H\beta - T - \varepsilon\right)\end{aligned} \tag{3}$$

where $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_N], \alpha_j \in R^m (j = 1, 2, \ldots, N)$, and α represents the Lagrange weights. Let's solve the partial derivative of each variable in the Lagrange function and make partial derivative to zero:

$$\begin{cases} \frac{\partial}{\partial} \to \beta^T = \alpha H & (4.1) \\ \frac{\partial}{\partial \varepsilon} \to \gamma \varepsilon^T + \alpha = 0 & (4.2) \\ \frac{\partial}{\partial \alpha} \to H\beta - T - \varepsilon = 0 & (4.3) \end{cases} \tag{4}$$

Substituting Eq. (4.3) to (4.2):

$$\alpha = -\gamma(H\beta - T)^T \tag{5}$$

$$\beta = \left(I^{-1}\gamma + H^T H\right)^{-1} H^T T \tag{6}$$

I is the unit matrix. Since the formula contains only a $N \times N (N \ll N)$ matrix inverse operations, the calculation is fast.

3 The Input Weight Optimization of RELM Based on GA

For random allocation of input weights, a large output change matrix leads to larger output weight matrix. As a result, Empirical risk and structural risk are increased to degenerate the robustness of RELM. Instead of randomly selected weight as the goal stated in [17], this chapter defines the reciprocal of the error as the fitness value with GA: (1) the error between the reference feature vectors and the feature vectors generated by the hidden layer of the RELM classifier can be minimized and then (2) the error between the desired output pattern and the actual output pattern of the RELM classifier is minimized.

3.1 Basic of Genetic Algorithm

Genetic Algorithms (GA) was first introduced in 1965 by professor Holland [18] based on the Darwin's theory of evolution and population genetics of Mendel. Biological strategies of behavior adaptation and synthesis are used to enhance the probability of survival and propagation during their evolution and it has good advantages in finding the best global answer of optimization problem. There are four major parts required in the conventional genetic algorithm to solve a problem, namely the encoding mechanism, the fitness function, the variables for controlling and the genetic operators. With the reason that GA can't deal with the parameters of the problem space directly, its first step has to convert the solution parameters form of the optimization problem to the gene chain expression. In this chapter, we use binary number to encode the solution.

Without using external information in the optimization search, it only evaluates each chromosome's pros and cons depending on the value of fitness function. The bigger fineness value of an individual indicates the better fitness for that individual to survive for next generation.

3.2 The Weight Optimization Based on GA

According to the above analysis, the weight optimization based on GA between input layer and hidden layer works as follows:

Step 1: The initial population of weight is filled with individuals that are generally created at random. Each chromosome is encoded in binary numbers which correspond to the relationship of input layer and hidden layer.
Step 2: The fitness function is defined as the reciprocal of the error function. So the bigger of the fitness value has, the better fitness a solution has.
Step 3: If the termination criterion is met, the best solution is returned.

Step 4: From the current population, the least fitted individuals are omitted based on the previously computed fitness values.
Step 5: New individuals are generally created as offspring of parents after crossover and mutation operations.
Step 6: Actions starting from Step 2 are repeated until the termination criterion is satisfied.

$$F = 1/E \tag{7}$$

$$E = \frac{1}{2P} \sum_{P=1}^{P} \sum_{i=1}^{O} \left[Y_i^p(t) - Y_{di}^P(t) \right]^2 \tag{8}$$

where E is defined as the average square error function, the smaller difference of squares, the higher accuracy of training, thus, F is defined as the reciprocal of E which is the fitness function. The larger the fitness value, the better the training results, the more suitable weight obtained. P as the number of training samples; O for the number of output layer neurons, $Y_i^p(t)$ is the actual output of the ith neuron of the Pth sample, $Y_{di}^P(t)$ for the expected output. The bigger E is, the smaller F is.

4 Output Weights of RELM Based on Cholesky Decomposition

Computing the output weight β is the next task after optimizing the input weight in RELM. In [14], the authors proposed one method which involve matrix inversion. However, it requires intensive computation which reduces training efficiency of the RELM. In this chapter, we propose an approach to obtain RELM output weights based on Cholesky decomposition.

By the Eq. 4:

$$\left(r^{-1} I + H^T H \right) \beta = H^T T \tag{9}$$

Make $A = r^{-1} I + H^T H, b = H^T T$, the Eq. 9 is transformed to:

$$A\beta = b \tag{10}$$

Prove coefficient A is symmetric positive definite matrix:

Step 1:

$$A^T = \left(r^{-1} I + H^T H \right)^T = r^{-1} I + H^T H = A \tag{11}$$

Step 2:

$$x^T A x = x^T \left(r^{-1} I + H^T H \right) x = r^{-1} x^T x + (Hx)^T Hx \tag{12}$$

From what has been discussed above, A is a symmetric matrix when $x \neq 0, r^{-1}x^T x > 0, (Hx)^T Hx > 0$. Therefore, $x^T Ax > 0$. So A is symmetric positive definite matrix.

$$A = SS^T \tag{13}$$

S is a triangular matrix with a diagonal of positive element:

$$S_{ij} = \begin{cases} \sqrt{a_{ij} - \sum_{n=1}^{i-1} S_{in}^2} & i = j, \\ \left(a_{ij} - \sum_{n=1}^{j-1} S_{in} S_{jn}\right) \Big/ S_{jj} & i > j. \end{cases} \tag{14}$$

Substituting Eq. 13 to 12 and multiplying S^{-1} with both sides:

$$S^T \beta = F \tag{15}$$

$$\text{Among them } F = S^{-1} b \tag{16}$$

$$f_i = \begin{cases} b_i \big/ S_{ii} & i = 1, \\ \left(b_i - \sum_{n=1}^{i-1} S_{ni} f_n\right) \Big/ S_{ii} & i > 1. \end{cases} \tag{17}$$

Eventually:

$$\beta_i = \begin{cases} f_i \big/ S_{ii} & i = N, \\ \left(f_i - \sum_{n=1}^{N-i} S_{i+n} \beta_{i+n}\right) \Big/ S_{ii} & i < N. \end{cases} \tag{18}$$

In conclusion, the solution of β is based on Cholesky decomposition only using simple arithmetic in which the calculation is simple and fast.

In addition, while the hidden layer neurons of RELM from N into N+1, the neuron matrix becomes:

$$H_{N+1} = \left[H_N | h_{N+1} \right] \tag{19}$$

$$\text{Subject to } h_i = \left[g(a_i \cdot x_1 + b_1) \cdots g\left(a_i \cdot x_j + b_i\right) \right]^T$$

$$\text{Therefore, } A_{N+1} = H_{N+1}^T H_{N+1} + \frac{I_{N+1}}{\gamma} \tag{20}$$

The Eq. 14 shows that the $N(N+1)/2$ non-zero elements of result S_{N+1} of A_{N+1} after Cholesky decomposition are equal to S_N. Therefore, by only calculating $L+1$

Table 1 Experiment datasets

Dataset	Sample num	Gene num	Class distribution	
			Class name	Num
Breast	97	24481	Relapse	46
			Non-Relapse	51
Leukemia	72	7129	ALL	24
			MLL	20
			AML	28
Colon	62	2000	Negative	40
			Positive	22
Heart	270	3510	Negative	150
			Positive	120

non-zero elements from $S_{N+1,1}$ to $S_{N+1,N+1}$, we can obtain S_{N+1}:

$$b_{N+1} = H_{N+1}^T T = \begin{bmatrix} b_N \\ h_N^T T \end{bmatrix} \tag{21}$$

$$F_{N+1} = \left[F_N \Big/ f_{N+1} \right] \tag{22}$$

In conclusion, to get the F_{N+1}, we only need the f_{N+1}. It is not necessary to calculate $f_1 - f_n$. The information, which was stored during the process of computing β based on Cholesky decomposition, can be fully used so that the β_{N+1} is computed based upon β directly. It is more convenient and quick than the method introduced in Eq. 6.

5 Performance Verification

To demonstrate the performance of COW-RELM, four gene expression datasets are selected as showed in Table 1. Among them, the Breast, Colon and Heart are the two-class dataset and the Leukemia is the multi-class dataset.

Four different kinds of algorithms, BP [19], SVM [20], ELM [9] and RELM [13] are used to compare with COW-RELM. Parameter C in SVM is set to ten. In these experiments, we generate the mean value by repeating fifty times to avoid the unstable situation of the algorithms.

COW-RELM algorithm steps are as follows:

Step 1: Optimize the input weights instead of randomly selected of RELM by GA, compute A_N and b_N;
Step 2: Calculate S_N according to the Cholesky decomposition of A_N, using Eq. 17 to calculate F_N;
Step 3: Calculate β_N by S_N and F_N according to Eq. 18;

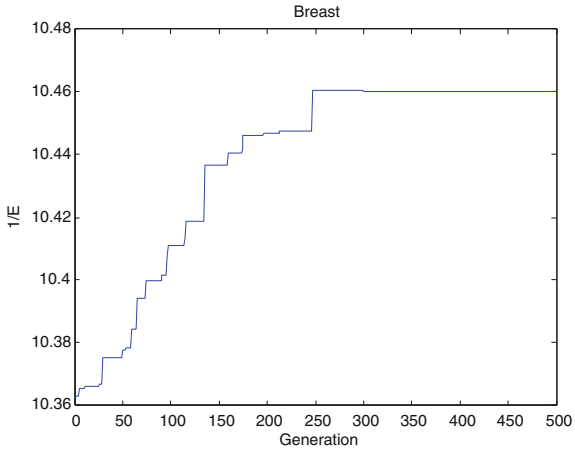

Fig. 1 Fitness curve of COW-RELM on the breast datasets

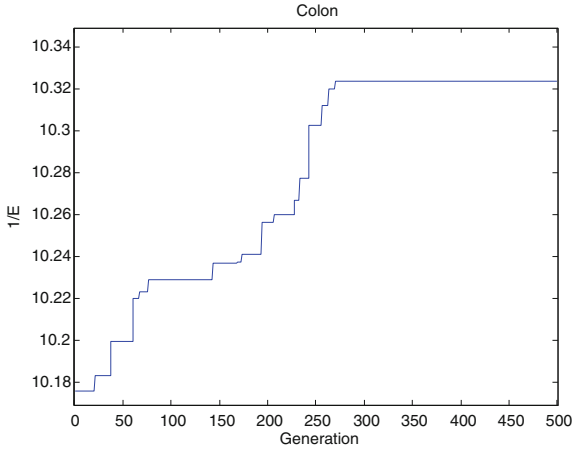

Fig. 2 Fitness curve of COW-RELM on the colon datasets

Step 4: Take mean value through multi-times testing.

In step 1, the sample for training with GA to find the proper input weights which replace the random original input layer of RELM.

Fitness carve of COW-RELM on the Breast, Leukemia, Colon and Heart datasets are shown in Figs. 1, 2, 3 and 4.

From Fig. 5, COW-RELM performs better than other algorithms in classification accuracy. In addition, with the determined weights, COW-RELM also outperforms other algorithms in time, and it is about 2 or 3 fold faster than ELM and RELM as showed in Table 2 and Fig. 6.

In order to reflect the generalization ability of COW-RELM, the Root Mean Square Error (RMSE) is used. As showed in Table 3, on the four datasets, the RMSE of COW-RELM is the smallest in most datasets.

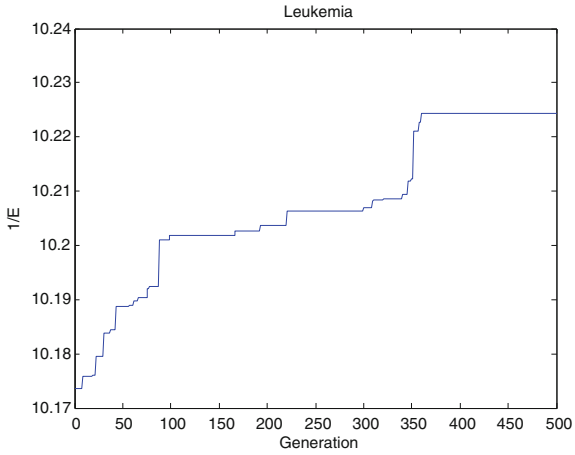

Fig. 3 Fitness curve of COW-RELM on the leukemia datasets

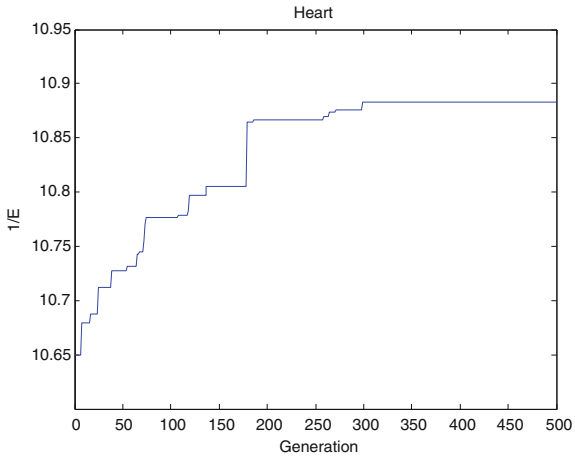

Fig. 4 Fitness curve of COW-RELM on the heart datasets

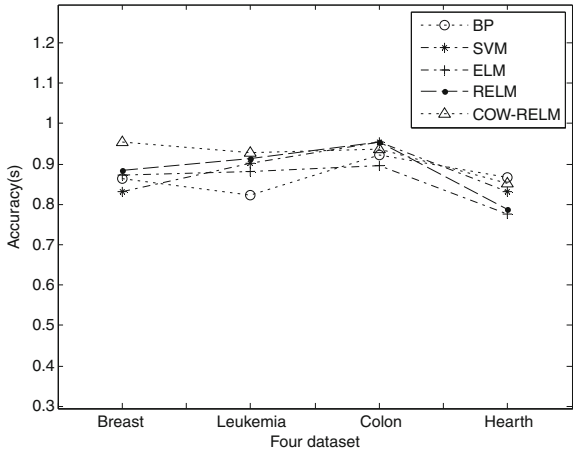

Fig. 5 Comparison of classification accuracy of BP, SVM, ELM, RELM and COW-RELM

Table 2 Comparison of training and testing time of BP, SVM, ELM, RELM and COW-RELM

Dataset	BP(s)		SVM(s)		ELM(s)		RELM(s)		COW-RELM (s)	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Breast	3.091	0.052	1.202	0.244	0.769	0.229	0.902	0.130	0.553	$<10e^{-4}$
Leukemia	2.085	0.012	0.882	0.212	0.450	0.051	0.687	0.048	0.340	$<10e^{-4}$
Colon	0.904	0.009	0.987	0.128	0.221	0.045	0.455	0.155	0.112	$<10e^{-4}$
Heart	0.108	$<10e^{-4}$	0.652	0.337	0.371	0.041	0.590	$<10e^{-4}$	0.112	$<10e^{-4}$

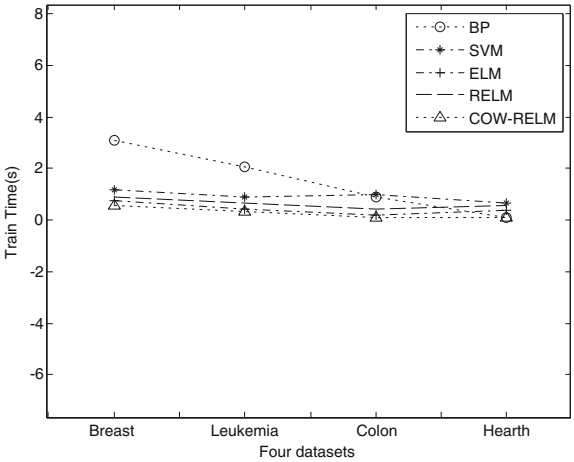

Fig. 6 Comparison of training and testing time of BP, SVM, ELM, RELM and COW-RELM

Table 3 Comparison of training RMSE and testing RMSE of BP, SVM, ELM, RELM and COW-RELM

Dataset	BP		SVM		ELM		RELM		COW-RELM	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Breast	0.0478	0.2643	0.0718	0.0728	0.0378	0.2643	0.2470	0.2679	0.2145	0.2011
Leukemia	0.0164	0.1829	0.0534	0.0540	0.0512	0.4829	0.1897	0.2002	0.1877	0.1801
Colon	0.0204	0.0337	0.0461	0.0420	0.0980	0.0937	0.0754	0.0994	0.0724	0.0753
Heart	0.0430	0.0446	0.0117	0.0101	0.0318	0.0346	0.0624	0.0660	0.0624	0.0360

6 Conclusions

In this chapter, we have developed an improved regularized extreme learning machine which is based on weights optimization and Cholesky decomposition, and then applied the algorithm for gene expression data classification. In order to obtain higher accuracy, the input weights of RELM are optimized based on Genetic Algorithm (GA). The output weights matrix is optimized by the Cholesky decomposition method to improve the training speed and thus reduce the training time. The new

algorithm has been applied on the Breast, Leukemia, Colon, Heart and other gene expression data. The experimental results show significant improvement in classification results.

Acknowledgments The authors would like to thank the National Natural Science Foundation of China (No. 61272315, No. 60842009, and No. 60905034), Zhejiang Provincial Natural Science Foundation (No. Y1110342, No. Y1080950) and the Pao Yu-Kong and Pao Zhao-Long Scholarship for Chinese Students Studying Abroad.

References

1. P.J. DeRisi, L.P. Brown et al., Use of a cDNA microarray to analyse gene expression patterns in human cancer. J Nat. Genet. **14**, 457–460 (1996)
2. J.P. Ye, L. Tao, X. Tao, R. Janardan, Using uncorrelated discriminant analysis for tissue classification with gene expression data. IEEE/ACM Trans. Comput. Biol. Bioinformat. **1**, 183 (2004)
3. C.H. Zheng, D.S. Huang, X.Z. Kong, X.M. Zhao, Gene expression data classification using consensus independent component analysis. J. Genomics Proteomics Bioinformat. **6**, 74–78 (2008)
4. U. Alon, N. Barkai, D.A. Notterman, K. Gish, Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. J. Proc. Natl Acad. Sci. **96**, 6745–6750 (1999)
5. J. Khan, M.L. Bittner, Y. Chen, DNA microarray technology: the anticipated impact on the study of human disease. Biochim. Biophys. Acta **1423**, 17–28 (1999)
6. Reverse engineering causal networks from multiple myeloma gene expression data, http://www.dcs.ex.ac.uk/~anarayan/publications/myeloma_paper1.pdf
7. T.S. Furey, N. Cristianini, N. Duffy, Support vector machine classification and validation of cancer tissue samples using microarray expression data. J Bioinformat. **16**, 906–914 (2000)
8. H.J. Lu, J.J. Lu, M.Y. Wang, Y. Lu, Classification of cancer gene expression data based on compressed sensing. J. China Univ. Metrol. **23**, 70–74 (2012). (in china)
9. G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications. J. Neurocomput. **70**, 489–501 (2006)
10. H.J. Lu, C.L. An, X.P. Ma, E.H. Zheng, X.B. Yang, Disagreement measure based ensemble of extreme learning machine for gene expression data classification. Chinese J. Comput. **36**, 341–348 (2013). (in china)
11. G.-B. Huang, L. Chen, C.K. Siew, Universal approximation using incremental feedforward networks with arbitrary input weights. J. Neural Netw. **17**, 879–892 (2006)
12. G.-B. Huang, L. Chen, Enhanced random search based incremental extreme learning machine. J. Neurocomput. **71**, 3060–3068 (2008)
13. W.Y. Deng, L. Chen, Regularized extreme learning machine. J. Data mining. pp. 385–389 (2009)
14. Z.H. Man, K. Lee, D.H. Wang, Z.W. Cao, S. Khoo, Robust single-hidden layer feedforward network-based pattern classifier. IEEE Trans. Neural Netw. Learn. Syst. **23**, 1974–1986 (2012)
15. G.B. Huang, D.H. Wang, Y. Lan, Extreme learning machines: a survey. J. Mach. Learn. Cybern **2**, 107–122 (2011)
16. S. Haykin, *Neural Networks: A Comprehensive Foundation* (Prentice Hall, New Jersey, 1999)
17. Z.H. Man, K. Lee, D.H. Wang, Z.W. Cao, S. Khoo, An optimal weight learning machine for handwritten digit image recognition. J. Sign. Proces. **93**, 1624–1638 (2013)

18. J.H. Holland, The psychology of vocational choice: a theory of personality types and model environments. J. Couns. Psychol. **12**, 25 (1965)
19. D.E. Rumelhart, J.L. McClelland, *Parallel Distributed Processing*, vol. 1, (MIT Press, Cambridge, 1986), pp. 125–187
20. C. Nelb, S.T. John, *An Introduction to Support Machines* (House of Electronics Industry, Hollywood, 2000)

A Stock Decision Support System Based on ELM

Chengzhang Zhu, Jianping Yin and Qian Li

Abstract People often tend to use a reliable way to predict the stock market in order to get a substantial return on investment. However, with plenty of uncertainty and noise, prediction is full of challenging and risk when it comes to stock markets. This chapter combines extreme learning machine (ELM) and the Oscillation box theory together to construct a stock decision support system, which can help people make decisions on stock trading through suggestion buy or sell stock. In experiments, 4 typical stock movements have been tested trading and 400 stocks in S&P500 are used to detect the performance of the system. Results show that our method is much better than buy-and-hold strategy.

Keywords Stock predict · ELM · Oscillation box theory

1 Introduction

The study of stock market, which helps people make lucrative investment decisions, is a focus of attention. However, owing to the fact that stock market indices are essentially dynamic, nonlinear, complicated, nonparametric, and chaotic, the stock time-series forecasting is regarded as one of the most challenging applications of time-series forecasting [1]. In recent years, a lot of work had been done and trying to analyse and predict stock prices or trends in the future [2]. Although nobody

C. Zhu (✉)
College of Computer, National University of Defense Technology, Changsha 410073, China
e-mail: kevin.zhu.china@gmail.com

J. Yin
State Key Laboratory of High Performance Computing, National University of Defense Technology, Changsha 410073, China

Q. Li
School of Economics, Minzu University of China, Beijing 100081, China

F. Sun et al. (eds.), *Extreme Learning Machines 2013: Algorithms and Applications*, Adaptation, Learning, and Optimization 16, DOI: 10.1007/978-3-319-04741-6_6,

can predict a stock market with high enough accuracy, one could be able to predict the overall trend in this market based on historical data. Therefore, people may obtain high profit by using some well trading strategies with the prediction results. A successful stock market prediction is characterized by achieving best results using minimum required input data and the least complex stock market model [3]. Since it is affected by many macro economic factors, the stock market cannot be well comprehensively described by traditional model [4]. As a comparatively accurate solution can be found in the complex, noisy environment due to artificial neural networks [5], lots of attentions have been devoted to applying different neural networks into stock prediction [6–9]. In addition, with the application of SVM in regression, some work introduced SVM method in stock market prediction and got outstanding result [10, 11]. All the methods mentioned above have made breakthrough achievements, which made the stock market prediction significantly accurate and robust however, there still exists an unresolved problem that is the speed will become very slow when a large number of historical data in the stock market need to be learned. It limits computer dynamic learning new data.

Recently, a new type of learning machine called extreme learning machine (ELM), which is a methodology for learning single-hidden layer feedforward neural networks (SLFN) and is proposed by Huang et al. [12–15], has been proved to be extremely fast and it can also provide excellent generalization performance. Different with the traditional neural network training algorithms such as back-propagation algorithm (BP), ELM does not need any other extra time to adjust the hidden weights and biases since it chooses them at random and then obtains the output layer weights and biases analytically. For this reason, we can introduce ELM to forecast the stock price trends in the future in order to get a better performance in a short time.

A powerful trading strategy is necessary for stock transactions. Nicolas proposed a box theory, which indicates the price of stock would generally oscillate in a certain range in a period of time named price box. The price will fall when it is close to the upper boundary of the price box and rise on the contrary. If the price breaks the upper boundary or the lower boundary of the oscillation box, it will enter another oscillation box in which the price will start a new upward or downward trend. So it will be the best time to buy or sell the stock [16]. It is fairly clear that the most important and difficult work is to accurately identifying the boundary of the box and confirm the price breakout it, since one can only predict it based on experience in daily life.

The box theory and extreme learning machine algorithm are combined in this chapter. We train extreme learning machine by history price data and utilize it to predict the highest and lowest stock price in the next period as the upper and lower boundary of the oscillation box. Meanwhile, we have developed an inspection rule to confirm whether the stock price breakout the boundary or not. Then we can formulate our trading strategy based on the box theory to make decisions. Experiments show that our approach has obvious performance advantages compared to hold-and-buy strategy in which an investor buys stocks and holds them for a long period of time, regardless of fluctuations in the market. The advantages of the systems are mainly reflected in two aspects. On one hand, the system has a great learning speed to learn

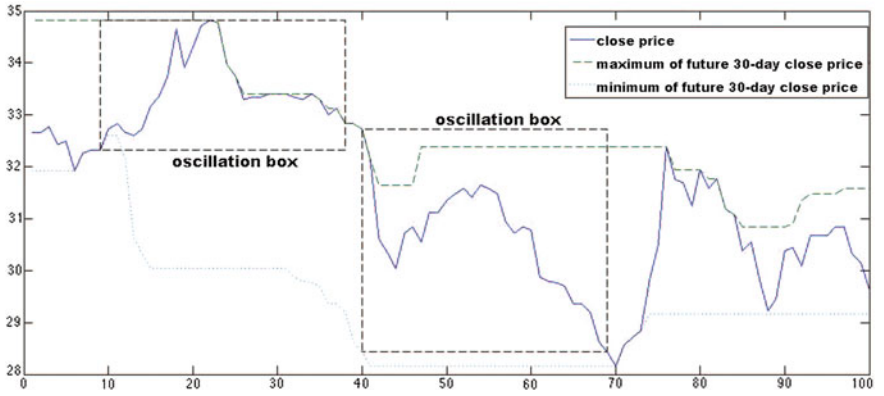

Fig. 1 The oscillation box samples

from a large number of historical stock price data and make decisions. On the other hand, the system can always get a better performance than it compared with other methods.

The remainder of this chapter is organized into four sections. Section 2 briefly reviews the oscillation box theory and extreme learning machine theory. Section 3 details our system trading strategy. Section 4 shows experiments and analysis. Finally, Sect. 5 contains the concluding remarks.

2 Related Work

2.1 Oscillation Box Theory

Nicolas proposed the Oscillation box theory. The basic idea of this theory is that the stock price is always has a certain shock range in a period of time, thus it has a maximum and a minimum price during this time. Imaging there are two ends of a box—the upper boundary and the lower boundary, thus Nicolas had it called oscillation box in his theory. When the stock price close to the lower boundary it has the rising trend and on the contrary close to the upper boundary. Furthermore, the price will go into another box to start a new shock in a range after it breaks through the boundary. The Oscillation box is showed in Fig. 1. Obviously, we can get a fruitful profit if we buy the stock when the price breaks the upper boundary and sell it as soon as it breaks the lower boundary. However, effective to detect the price when it breaks through the boundary, which is always based on experience, is quite challenging. In our system, we proposed a method to detect it automatically based on the ELM prediction.

2.2 Extreme Learning Machine

Extreme learning machine is a novel algorithm proposed by Huang et al. in [14]. The theory provides a new approach to training the single hidden layer feedforward networks, which makes the training completed within a very short time to achieve the effect of extreme learning. A SLFN consists of three layers, namely input layer, hidden layer and output layer. We can train the network by adjusting the connection weights and biases of layers.

Denote the numbers of nodes in input, hidden and output layers as n_1, n_2 and n_3, we can represented a SLFN by

$$\boldsymbol{t}_r = f_r(\boldsymbol{x}_j) = \sum_{i=1}^{n_2} \beta_{ir} G_i(\boldsymbol{a}_i, b_i, \boldsymbol{x}_j) \qquad (j = 1, 2, \ldots, n_1; r = 1, 2, \ldots, n_3). \quad (1)$$

where $\boldsymbol{t}_r = [t_{r1}, t_{r2}, \ldots, t_{rn}]^T$ is the output vector; $\boldsymbol{x}_j = [x_{j1}, x_{j2}, \ldots, x_{jn_1}]^T$ is the input vector; $\boldsymbol{a}_i = [a_{1i}, a_{2i}, \ldots, a_{n_1 i}]$ represents the connection weights between the input layer and ith node in the hidden layer; $\boldsymbol{\beta}_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{in_3}]^T$ represents the connection weights between the ith node in the hidden layer and the output layer; b_i means the ith hidden node bias; $G_i(\boldsymbol{a}_i, b_i, \boldsymbol{x}) = g(\boldsymbol{a}_i \cdot \boldsymbol{x} + b_i)$ is the value provided by the network for $\boldsymbol{x}_i$ in the hidden layer, where $g(\cdot)$ represents the activation function of the hidden layer. The $g(\cdot)$ can have a variety of options such as Sigmoid function, Sine function, Hard Limit function, Triangular basis function and Radial basis function.

The above Eq. (1) can be written compactly as

$$\boldsymbol{T} = \boldsymbol{G} \cdot \boldsymbol{\beta}. \quad (2)$$

where

$$\boldsymbol{G} = \begin{bmatrix} g(\boldsymbol{a}_1 \cdot \boldsymbol{x}_1 + b_1) & \ldots & g(\boldsymbol{a}_{n_2} \cdot \boldsymbol{x}_1 + b_{n_1}) \\ \vdots & \ddots & \vdots \\ g(\boldsymbol{a}_1 \cdot \boldsymbol{x}_n + b_1) & \ldots & g(\boldsymbol{a}_{n_2} \cdot \boldsymbol{x}_n + b_{n_1}) \end{bmatrix}_{n \times n_2}. \quad (3)$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \\ \vdots \\ \boldsymbol{\beta}_{n_2} \end{bmatrix}_{n_2 \times n_3}. \quad (4)$$

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1 \\ \boldsymbol{t}_2 \\ \vdots \\ \boldsymbol{t}_n \end{bmatrix}_{n \times n_3}. \quad (5)$$

The extreme learning machine is trying to minimize the empirical and structural error by adjusting the weights and biases. The objective can be written as

$$minE(\boldsymbol{a}_i, \boldsymbol{\beta}_i) = \sum_{r=1}^{n} ||\boldsymbol{t}_r - \boldsymbol{T}_r||. \tag{6}$$

where $\boldsymbol{T}_r$ represents the real target values. In ELM theory, $\boldsymbol{a}_i$ and b_i, which are the weights and biases of hidden layer, can be randomly assigned. We only need to focus on $\boldsymbol{\beta}_i$, which are the weights of output layer. The theory provides that solving the optimization problem Eq. (6) is equivalent to Eq. (2) for its least square solution $\boldsymbol{\beta}$. It will be easy to get the weights $\boldsymbol{\beta} = \boldsymbol{G}^{\dagger} \cdot \boldsymbol{T}$ based on the Moore Penrose generalized inverse matrix theory, where $\boldsymbol{G}^{\dagger}$ is the generalized inverse matrix of $\boldsymbol{G}$.

2.3 Gray Correlation Degree (GCD)

In [17], Deng proposed the gray correlation degree, which has been applied in many fields [18]. The method is using the geometric shape of sequence curves to present the relational degree between two data sequences. The closer the two curves are, the higher degree is it. If we have a feature $\boldsymbol{X} = [x_1, x_2, \ldots, x_n]$ and target $\boldsymbol{T} = [t_1, t_2, \ldots, t_n]$ we can calculate the feature GCD as follow:

$$r(t_i, x_i) = \frac{min|t_i - x_i| + \xi max|t_i - x_i|}{|t_i - x_i| + \xi max|t_i - x_i|}. \tag{7}$$

$$r(\boldsymbol{T}, \boldsymbol{X}) = \frac{1}{n} \sum_{i=1}^{n} r(t_i, x_i). \tag{8}$$

where $\xi \in (0, 1)$ is the discernibly coefficient which often set to 0.5. The $r(t_i, x_i)$ is the gray correlation degree of $\boldsymbol{T}$ and $\boldsymbol{X}$ at ith point. The $r(\boldsymbol{T}, \boldsymbol{X})$ is the gray correlation degree of $\boldsymbol{T}$ and $\boldsymbol{X}$.

3 Detail of the Decision Support System

The decision support system is in accordance with the following steps. First, the system calculates the related indicators from the historical data of the stock market and scales it. Then it obtains relationships of the scaled indicators and stock prices time-series, which could set as the input value weight of the ELM. Next step comes to training the ELM, using the weighted indicators sequence as input values and stock history prices time-series as target values. The third step is using the trained ELM to predict the stock price sequence for the next period of time in order to

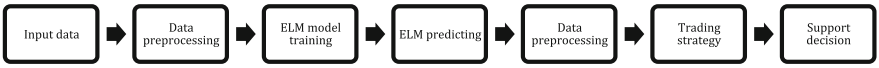

Fig. 2 Support system architecture

get the minimum and maximum values of stock prices, which can set as the lower and upper bounds of the oscillation box. Finally, the transaction is carried out in accordance with the trading strategy based on box theory. The architecture of the system is showed in Fig. 2.

3.1 Pre-processing History Indictors

The system uses closing price to present stock market price, which is the target value. We have selected some indicators as the ELM input feature values, which are OPEN, HIGH, LOW, CLOSE, VOL, AMOUNT, MA, ROC, RSI, FASTK, SLOWK, SLOWD. The computations can be found in [19]. In order to get the boundary in next few days through prediction the stock market time-series, the boundary in period few days can be used as features. In this system, we use the highest price and lowest price of the stock in the future n_1 days as the upper boundary Up_k and lower boundary Low_k.

$$Up_k = max(C_{i+1}, C_{i+2}, \ldots, C_{i+n_1})$$

$$Low_k = min(C_{i+1}, C_{i+2}, \ldots, C_{i+n_1})$$

were C_k represent the closing price in the kth day.

There are totally 14 indictors and 1 target in our system. If a indictor data sequences are $X = (x(1), x(2), ..., x(n))$, all data of the indictor will be normalized to $[-1, 1]$ by

$$x(i)_{normalize} = -1 + 2\frac{x(i) - min(X)}{max(X) - min(X)} \tag{9}$$

Otherwise, the prediction result will be denormalized by

$$p_{denormalize} = \frac{p(max(X) - min(X)) + max(X) + min(X)}{2}. \tag{10}$$

where p is the prediction result.

However, the influence degrees of these indicators on the prediction results are not equal. Obviously, enlarge the indicator, which has considerable impact on the result, can help get a more accurate prediction result. Therefore we use the gray relation analysis method to get relationships of the scaled indicators and stock market prices time-series and use it as the input weight w_i.

3.2 Stock Prices Prediction Based on ELM

This system needs to predict prices in the next n_1 days based on history data in n_2 previous days. We define the target vector as $\boldsymbol{T}_i = [C_{i+1}, C_{i+2}, \ldots, C_{i+n_1}]$, the feature vector as

$$\boldsymbol{F}_i = [O_k, H_k, L_k, C_k, VOL_k, MA_k, ROC_k, RSI_k, FastK_k, SlowK_k, SlowD_k, Up_k, Low_k]$$

where $k = (i-1, i-2, \ldots, i-n_2)$, all features are indictors mentioned in Sect. 3.1.

Since the input weights w_i has been acquired based on gray relation analysis method, we can descript the weight vector as $\boldsymbol{W}_i = [w_{O_k}, w_{H_k}, \ldots, w_{Low_k}]$, where $k = (1, 2, \ldots, n_2)$. The input vector $\boldsymbol{I}_i$ can calculate as follow:

$$\boldsymbol{I}_i = \boldsymbol{F}_i \circ \boldsymbol{W}_i. \tag{11}$$

where $\circ$ is Hadamard product, i presents ith day.

Due to the recent data do more contribution to learning stock market, we need to set up a window that contains recent data for ELM training. If we want to predict stock prices after ith-day, and the window size is set to n days. The input vectors can form a matrix as $\boldsymbol{I} = [\boldsymbol{I}_{i-n}, \boldsymbol{I}_{i-n+1}, \ldots, \boldsymbol{I}_{i-n_1}]$ while the target vectors can form a matrix as $\boldsymbol{T} = [\boldsymbol{T}_{i-n}, \boldsymbol{T}_{i-n+1}, \ldots, \boldsymbol{T}_{i-n_1}]$. After training ELM, which used $\boldsymbol{I}$ as input matrix and $\boldsymbol{T}$ as target matrix, we can predict stock price $\boldsymbol{T}_i$ using input vector $\boldsymbol{I}_i$. The upper boundary and lower boundary can set to maximize and minimize price of $\boldsymbol{T}_i$.

3.3 Trading Strategy Based on Box Theory

Our trading strategy is based on the oscillation box theory. After predicting the upper and lower boundary, which are described as Up_i and Low_i, in next n_1 days after ith-day, we need to set a standard to detect whether the price series crossing the border. Obviously, two conditions need to be met when the price series up through the box. The first thing is that the price is very close to the lower boundary of the new box, and the second thing is that the lower boundary of the new box is moved upward. Similarly, the price will close to the upper boundary of the new box and the upper boundary of the new box will move downward when it crosses the lower boundary. Thus our strategy can be defined as Algorithm 1.

Algorithm 1 Trading strategy.

if next trade == buy **then**
  if $\frac{|C_i - Low_i|}{C_i} \leqslant \sigma$ and Low_i is in uptrend **then**
    if $sellprice - C_i \geqslant \varphi$ **then**
      Buy,$buyprice = C_i$
      next trade=sell
    end if
  end if
else if $\frac{|C_i - Up_i|}{C_i} \leqslant \sigma$ and Up_i is in downtrend **then**
  if $C_i - buyprice \geqslant \phi$ **then**
    Sell,$sellprice = C_i$
    next trade = buy
  end if
  if $\frac{buyprice - C_i}{buyprice} \geqslant \theta$ **then**
    Sell,$sellprice = C_i$
    next trade=buy
  end if
end if

4 Experiments

We conducted some experiments to verify the system's feasibility and efficiency. In these experiments, several typical stock movements, such as bull market, bear market, fluctuant market and so on, are selected to carry out a comparative analysis. After that we tested 400 stocks in the S&P500, in order to detect the average performance of our system. Finally, the optimal sets of parameters are discussed and tested. All of the experiments are run in MATLAB environment.

4.1 Performance Evaluation

There are two performance indicators in our experiments. One of them is MSE (mean squared error), which is used to illustrate the accuracy of ELM regression. The MSE is defined as follow:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_i^*)^2. \tag{12}$$

where y_i is the actual output and y_i^* is the estimate.

The other is rate of profit which can defined as

$$\text{rate of profit} = (Y - Y_0)/Y_0 \times 100\,\%. \tag{13}$$

where Y is the money after trade and Y_0 is the initial money.

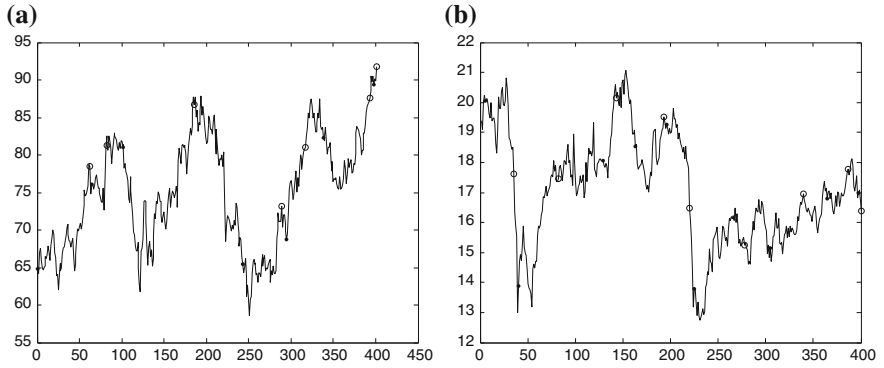

Fig. 3 **a** The fluctuant and bull market movement, where • means buy point and ○ means sell point. **b** The fluctuant and bear market movement

In our experiments, we suppose $10000 initial money and use all money or stock to trade at each operation. As the real trading, we set the transaction cost of each trading 0.5 %. Then we let the system trade on a stock for a period of time and get the average MSE and rate of profit in the final. In particular, we short-selling of all stocks hold at the last day of test trading.

4.2 Typical Stock Movement Trading

A movement of fluctuant and bull market is showed as Fig. 3a. The transaction rate σ is set to 0.01 and stop-loss rate θ is set 0.1 and φ, ϕ set to 0, 0.05, respectively. The window size is 120. In the experiment, our system profits to 93.20 % while the market gains about 41.69 %. The MSE of the ELM is $7.2957e^{-30}$. The data set is samples from March 13, 2002 to August 29, 2003.

A movement of fluctuant and bear market is showed as Fig. 3b. The transaction rate σ is set to 0.01 and stop-loss rate θ is set 0.1 and φ, ϕ set to 0, 0.05, respectively. The window size is 120. In the experiment, our system can profit to 38.61 % while the market losses about 15.48 %. The MSE of the ELM is $5.6965e^{-30}$. The data set is samples from October 26, 2000 to June 6, 2002.

Obviously, our system is significantly better than the buy-and-hold strategy in a fluctuant market.

A movement of overall bull market is showed as Fig. 4a. The transaction rate σ is set to 0.05 and stop-loss rate θ is set 0.1 and φ, ϕ set to 0, 0.05, respectively. The window size is 120. In the experiment, our system can profit to 137.17 % while the market gains about 94.38 %. The MSE of the ELM is $1.0645e^{-29}$. The data set is samples from March 17, 2004 to October 17, 2005.

A movement of overall bear market is showed as Fig. 4b. The transaction rate σ is set to 0.05 and stop-loss rate θ is set 0.1 and φ, ϕ set to 0, 0.05, respectively.

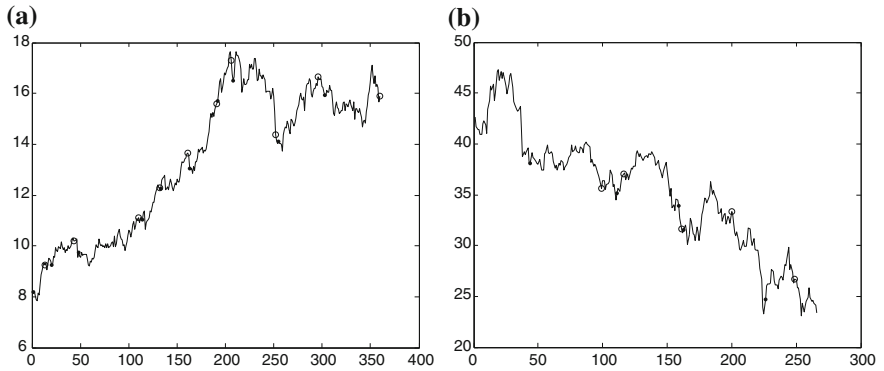

Fig. 4 **a** The overall bull market movement. **b** The overall bear market movement

The window size is 90. In the experiment, our system only losses 2.74 % while the market losses about 45.13 %. The MSE of the ELM is $2.3462e^{-30}$. The data set is samples from August 8, 2001 to September 9, 2002.

4.3 Trade on S&P500

We simulate trading of 400 stocks in 400 trading days, which are selected in S&P500 from Mar 18, 2004 to Oct 17, 2005, to examine the average performance of the system. There are 212 of these stocks movement are bull, the other are bear. In these trading test, the same parameters are used. The transaction rate σ is set to 0.01 and stop-loss rate θ is set to 0.05 and φ, ϕ set to 0, 0.05, respectively. While the window size is set to 120. The test results are showed in Table 1. Clearly, our system is in most cases superior to the buy-and-hold strategy, and can gains a much higher average profit.

4.4 The Optimal Sets of Parameters

Firstly, we consider the impact on the training window size. Obviously, the learning of history data in a period of time before can generate important guiding significance for the prediction of future case. However, if we take a long time to learn the prediction accuracy, it might be decreased because of much noise data. At the same time, if we take a short time to learn, it might lack of experience. On experience, a stock with bull movement always has less noise data over a longer period of time. Conversely, a stock with bear movement is often accompanied by more noise data. Therefore, if we set a large window size to a bull movement stock and set a small size to a bear movement stock, the system may give better results. We have already conducted experiment

Table 1 Average performance of trading 400 stocks in S&P500 for 400 days

Market pattern	Stock number	Less than buy-and-hold	Less (%)	Loss number	Loss (%)	Average profit (%)	Average profit of buy-and-hold (%)
Bull	212	47	21.17	0	0	40.16	27.32
Bear	188	1	0.53	25	13.30	14.18	−20.68
Total	400	48	12.00	25	6.25	**27.95**	**4.76**

Table 2 Average performance of trading 400 stock in S&P500 for 400 days with window size control

Market pattern	Stock number	Less than buy-and-hold	Less (%)	Loss number	Loss (%)	Average profit (%)	Average profit of buy-and-hold (%)
Bull	212	32	15.09	0	0	57.28	27.32
Bear	188	0	0	8	4.26	14.73	−20.68
Total	400	32	8.00	8	2.00	**37.28**	**4.76**

Table 3 Average performance of trading 50 stock by varying n_2

	15/15	15/30	15/45	15/60	15/75
Average profit (%)	30.25	35.32	36.64	**37.38**	34.32
Number of transaction	15.2	14.6	13.4	12.8	12.2

Table 4 Average performance of trading 50 stock by varying n_1

	5/20	8/32	10/40	15/60	20/80
Average profit (%)	28.21	32.48	35.64	**37.38**	36.32
Number of transaction	14.2	13.4	12.2	12.4	11.2

while the window size is set to 120 trading days (nearly 3 months). Table 1 shows the results. Now we have tested same stocks again with window size control while other parameter settings are not change. The results are shown in Table 2. Revenue in this experiment has been significantly increased.

Secondly, since we have used the previous n_2 days for feature value extraction and prediction, the extreme value stock may reach in next n_1 days, the n_1 and n_2 may be considered. We first fix n_1 as 15 and chance n_2 from 1 to 5 times of n_1 to find the optimal value of n_2. The result illustrates in Table 3, which shows that the most profit will be gained while n_2 is 4 times of n_1. Then we adjust n_1 from 5 to 20, and set n_2 as 4 times of n_1. At this time we can see the average yield arrive maximum when n_1 is set to 15 in Table 4.

The last but not the least, the σ also need to be considered. On one hand this parameter controls the speed of transactions, on the other hand it also eliminates some of the impact caused by the prediction error. For fluctuations or bull movement, we can set it to a larger value to get more transactions. In contrast, for the bear movement,

Table 5 Performance of trading MSFT by varying σ

σ	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
Profit (%)	2.41	7.20	**8.15**	4.53	5.75	4.84	4.97	4.73	5.18	6.07
Number of transaction	8	9	9	9	9	9	9	9	9	8

it can be set a small value to reduce transactions. However, the parameter must be set within a range, otherwise it will reduce the accuracy. We selected a stock to show the impact for system profit of σ. The value of σ adjust from 0.005 to 0.04 while window size is set to 90, stop-loss rate θ is set to 0.05 and φ, ϕ set to 0, 0.05, respectively. The data set is samples from September 7, 2004 to March 24, 2006. In Table 5 we can see when σ is 0.015 the profit will be highest.

5 Conclusion

We have proposed a stock decision support system in this chapter. In the experiment, we have shown that this system is capable of superior performance to give investors considerable returns, especially when it is in a fluctuant movement the system can bring more lucrative benefits. It is mainly based on two reasons. The first one is the fast learning ability and high precision of ELM. The second one is that trading with box theory is based on the highest and lowest values the stock could reach in a period of time, which reduces the impact of noise and uncertainty in the stock market on the prediction accuracy. Similarly, using gray relation degree method to obtain each factor weight, to a certain extent, helps the ELM get more precise results.

The whole system is based on the ELM's prediction value to produce the result. However, features, which are used to train ELM, cannot fully represent all the influencing factors in the stocks. For this reason, ELM can only reach a certain degree of prediction accuracy. Therefore, how to reasonably model on the stock market and select more representative and comprehensive features become the future work.

Acknowledgments This work was supported by the National Natural Science Foundation of China (Project No. 60970034, 61170287 and 61232016).

References

1. S. Chen, P.M. Grant, A clustering technique for digital communications channel equalization using radial basis function networks. IEEE Trans. Neural Netw. **4**, 570–578 (1993)
2. X. Zhang, H. Fuehers, P.A. Gloor, in *Predicting Asset Value Through Twitter Buzz*. Advances in Collective Intelligence 2011 (Springer, New York, 2012), pp. 23–34
3. S.A. George, P.V. Kimon, Forecasting stock market short-term trends using a neuro-fuzzy based methodology. Expert Syst. Appl. **36**(7), 10696–10707 (2009)

4. T.Z. Tan, C. Quek, N.G. See, Biological brain-inspired genetic complementary learning for stock market and bank failure prediction. Comput. Intell. **23**(2), 236–261 (2007)
5. G. Grudnitski, L. Osburn, Forecasting S&P and gold futures prices: an application of neural networks. J. Futur. Market **13**, 631–643 (1993)
6. A.S. Chen, M.T. Leung, Regression neural network for error correction in foreign exchange forecasting and trading. Comput. Oper. Res. **31**, 1049–1068 (2004)
7. Y.K. Kwon, B.R. Moon, A hybrid neurogenetic approach for stock forecasting. IEEE Trans. Neural Netw. **18**(3), 851–864 (2007)
8. H.J. Liu, H.M. Chen, Y.R. Hu, Financial characteristics and prediction on targets of M&A based on SOM-Hopfield neural network, in *IEEE International Conference on Industrial Engineering and Engineering Management 2007*, pp. 80–84 (2007)
9. X. Lin, Z. Yang, Y. Song, T. Washio, The application of echo state network in stock data mining. PAKDD **5012**, 932–937 (2008)
10. L.J. Cao, E.H. Francis Tay, support vector machine with adaptive parameters in financial time series forecasting. IEEE Trans. Neural Netw. **14**(6), 1506 –1518 (2003)
11. Y.K. Bao, Forecasting stock composite index by fuzzy support vector machines regression, in *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, Guangzhou, 18–21 Aug 2005
12. G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications. Neurocomputing **70**, 489–501 (2006)
13. G.B. Huang, D.H. Wang, Advances in extreme learning machines. Neurocomputing **74**(16), 2411–2412 (2011)
14. G.B. Huang, Q.Y. Zhu, C.K. Siew, Real-time learning capability of neural networks. IEEE Trans. Neural Netw. **17**(4), 863–878 (2006)
15. G.B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification. IEEE Trans. Syst. Man Cybern. **42**, 513–529 Part B (2011)
16. D. Nicolas, How I made two million dollars in the stock market. BN Publishing, Illinois, America (2007)
17. J.L. Deng, Control problems of gray systems. Syst. Control Lett. **1**(4), 288–294 (1982)
18. L.J. Zhang, Z.J. Li, Gene selection for classifying microarray data using grey relation analysis. Discov. Sci. **4265**, pp. 378–382 (2006)
19. P. Martin, *Technical Analysis Explained*, 4th edn. Paperback. McGraw-Hill Company. ISBN0071226699

Robust Face Detection Using Multi-Block Local Gradient Patterns and Extreme Learning Machine

Sihang Zhou and Jianping Yin

Abstract A novel multi-block local gradient patterns (MB-LGP) based face detection method was proposed in this article. The MB-LGP operators extract face features in the way similar to local gradient patterns (LGP) however, the gradient of pixels in LGP was replaced by the counterparts of square image areas in MB-LGP. We have proved that the MB-LGP has most of the advantages of LGP and moreover with a stronger discriminant power and better robustness against noise. In the classification part, the extreme learning machine was introduced in the last stage in the proposed cascade classifier in order to speed up training process and increase classification accuracy. As was shown in experiments using the CMU+MIT database the new method possesses high detection rate.

Keywords Face detection · Multi-block local gradient patterns (MB-LGP) · Extreme learning machine (ELM)

1 Introduction

Face detection is one of the basic yet sophisticated procedure in a computer vision or an object recognition system whose accuracy and robustness can easily be influenced by changes in illumination condition, occlusions, facial expression, scale, pose, orientation, etc.

This work was supported by the National Natural Science Foundation of China (Project no. 60970034, 61170287, 61232016).

S. Zhou (✉)
College of Computer, National University of Defense Technology,
Changsha 410073, China
e-mail: zsh306114653@gmail.com

J. Yin
State Key Laboratory of High Performance Computing, National University
of Defense Technology, Changsha, China

F. Sun et al. (eds.), *Extreme Learning Machines 2013: Algorithms and Applications*,
Adaptation, Learning, and Optimization 16, DOI: 10.1007/978-3-319-04741-6_7,

In order to handle these variations, a good feature is indispensable. Among all the exiting features, local binary patterns (LBP) [1] are one of most attractive alternatives. Because of its stability against monotonic illumination changes and its computational simplicity, local binary patterns have drawn increasing interest of a lot of scientists. As a result, a large number of improved and transformed versions have been proposed focusing on different aspects of this feature in the recent years. Among them Jin et al. [2] enhanced the discriminative capability by comparing all the pixels in the patch with the mean intensity; Tan and Triggs [3] improved the robustness of original LBP by introducing a version with 3-value code local ternary patterns; Liao and Chung [4] changed the strategy to choose neighborhoods which leaded a great improvement on the variety and amount of information of the previous features. After this, some improvement had been made to extend the LBP to 3-D volume [5, 6]. Recently, a new method Local Gradient Patterns [7], which uses the gradient values of the neighboring pixel to gain the binary codes, has improved the invariance to local intensity variations. In addition to selecting features with good discriminability and classification performance, choosing a good classifier is also essential to an accurate and fast face detection system. One of the most significant contribution in the field of feature selection and sample classification was made by Viola and Jones [8]. In their work, the first real-time face detector with high accuracy was designed mainly due to the implementation of integral image representation, the cascaded framework and the use of Adaboost algorithm. However, the simplicity of Haar-like features and decision stump function has long been a bottleneck of this method which was the focus of researchers in recent years. For instance, Jones and viola [9] and Xu et al. [10] developed the original simple stump function into more complicated tree structure, Xiao et al. [11], Friedman et al. [12] and Li et al. [13] were mainly focusing on the improvement of strong classifier learning strategy. Moreover, Zeng et al. [14] developed the lower level of parallelism through OpenMP and higher level parallelism through MPI to accelerate the computation of the algorithm. In this article, we designed a four stage cascaded face detector. Novel multi-block local gradient patterns with strong classification ability as well as great invariance to both local and global intensity variation were proposed as features in the detection system. At the same time, the latest Single-hidden-Layer Feedforward Neural Networks called Extreme Learning Machine (ELM) [15] was introduced to make the final decision of the cascaded classifier so as to improve the classification accuracy.

2 Method

2.1 Face Representation

Introduction of MB-LGP A distinctive representation for face patterns is the basis of an accurate face detection system. Many face representation methods have been proposed. One of the most effective face features LBP attracted scientists' further

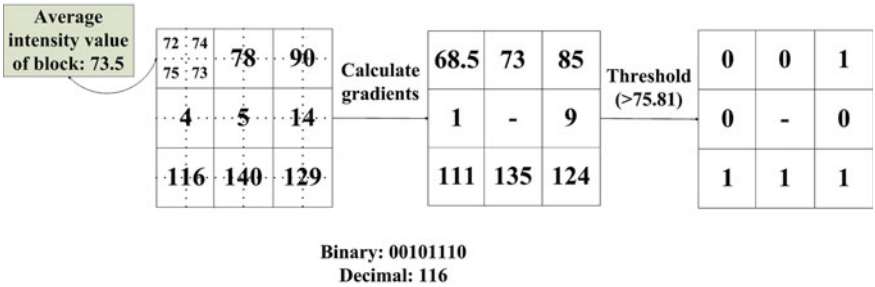

Fig. 1 Multi-block local gradient patterns for face representation

study mainly on reliance of its perfect displaying of texture in the local area with great robustness to illumination variations. Among all the extension of LBP, the latest local gradient pattern proposed by Jun, Kim [7] is an outstanding one. It was proved to be more stable to local intensity variation, less influenced by local color variation and more distinctive than LBP. However, the LGP features seem relatively sensitive to noise and location variation. In this article, we proposed a generalization of LGP–multi-block local gradient patterns (MB-LGP). The MB-LGP operator uses the gradient values of eight neighboring counterparts of a given square area in an image space, which are assigned by the absolute value of average intensity difference between the central square and its surroundings. After the gradient values is gained, compare the values with their average value, and assign 1 if the gradient value in the according square is larger than the mean value, and 0 otherwise. At last, concatenate the 0s and 1s in a clockwise direction (Fig. 1).

The procedure of extracting MB-LGP feature is similar with LGP's, besides that it is the gradient values of image blocks rather than those of pixels which were being compared. Define the average gradient value of a certain square area as $\overline{g}$ and the values of its neighboring areas as $\{g_1, g_2, \ldots, g_8\}$. Here g_n is obtained by calculating the absolute difference of average intensity value between the nth and the central image square. We define the average intensity of the nth image block as i_n, $(n = 1, 2, \ldots, 8)$ and i_c as the average intensity of the central image area. Then, g_n can be represented as: $g_n = |i_c - i_n|$ and $\overline{g} = \frac{1}{8}\sum_{n=1}^{8} g_n$. Then, the output value of the MB-LGP operator is:

$$MB - LGP = \sum_{i=1}^{8} s(g_i - \overline{g})2^i$$

where

$$s(x) = \begin{cases} 1 & x < 0, \\ 0 & otherwise. \end{cases}$$

In order to control the dimensionality of features, each block of the 3×3 operators is constrained to be square. In addition to that, the operator of MB-LGP can also be

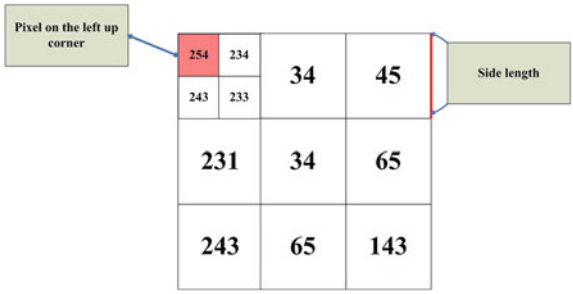

Fig. 2 Determination of a MB-LGP mask

extended to $MB - LGP_s$ with $s \times s$ pixels in each sub-block at different sizes and scales.

Consulting the work of Viola and Jones [9], a 24×24 square block were used to scan the images to seek for human faces, accordingly there 1436 different masks of MB-LGP operators. The number of masks is determined as follow: a mask can be determined when the position of the pixel on the top left corner and the side length of the sub-image-blocks fixes (see Fig. 2).

Define (x_i, y_i) as the coordinate of the pixel on the top left corner and a^* as the side length of sub-image-blocks, so the variation range of x_i and y_i is $[1, 24 - 3a + 1]$, while a^* ranges from 1 to 8. The total number of MB-LGP masks N_M in a 24×24 square block can be calculated by:

$$N_M = \sum_{a^*=1}^{8} (25 - 3a^*)^2 = 1436$$

Advantages of MB-LGP

As the generalization of LGP, MB-LGP has inherited most advantages of its predecessor, It can keep invariant both locally and globally [7]. Besides that, as MB-LGP uses the average pixel value in image boxes to represent the structure of objects, it is more robust against noises and more powerful to present the structure of human faces, especially in low resolution conditions. By comparing the pixel values as well as the patterns of LBP, LGP and MB-LGP, Fig. 3 is a good proof of MB-LGP's robustness against noise and strong representation ability. Here, $LBP_{8,8}$ and $LGP_{8,8}$ ($LBP_{8,8}$ and $LGP_{8,8}$ stands for the LBP and LGP operator with 8 pixels surrounding the centre pixel on the radius of 8), operators were used to extract the features from the original image for LBP and LGP, while $MB - LGP_8$ is adopted to extract the MB-LGP features.

The upper part of the Fig. 3 shows us the pixel and the average pixel values extracted from a low resolution testing image from CMU+MIT database. As we can see in the figure, LGP and MB-LGP could roughly reveal the eyes and mouth

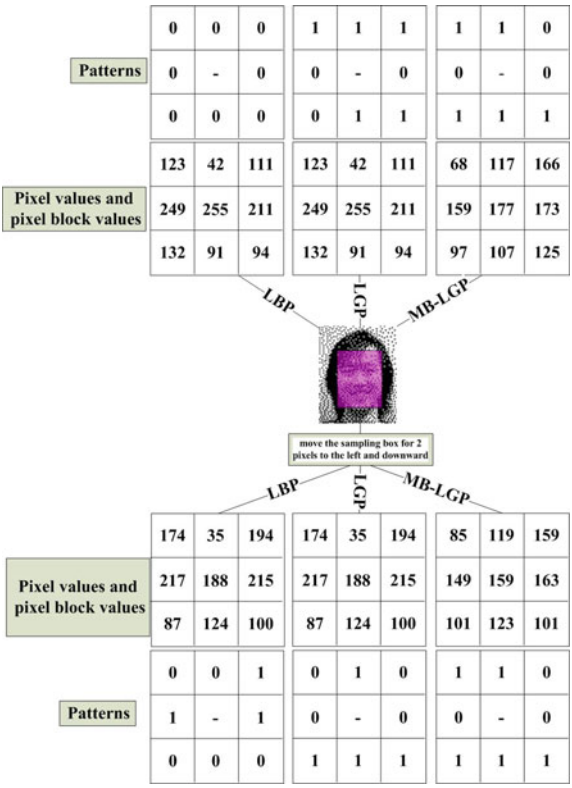

Fig. 3 Patterns extracted by $LBP_{8,8}$, $LGP_{8,8}$ and $MB-LGP_8$ on low resolution image and the variation of patterns when the scanning window translated

of the person while LBP showed nothing at all. In the under part of the figure, the sampling box was translated for 2 pixels to the left and downward respectively. After that, the pixel values and patterns of both LBP and LGP changed sharply. However the average pixel value of MB-LGP had only fluctuated slightly and the pattern of it kept the same. The great robustness against noises and great ability to represent large scale structure make MB-LGP more distinctive than LBP, LGP and Haar like features. This will be shown in the experiment in Sect. 3.

2.2 Feature Selection

As we are finding a real-time method for face detection, the time consumed by extracting 1436 MB-LGP for hundreds and thousands of times, on a single image is surely unacceptable. On the other hand, although the number of MB-LGP has sharply decreased as we constrain the rectangular regions to be squares, there are still a lot of redundant information. In view of the two mentioned issues, feature selection is needed, and the latest Adaboost algorithm happens to be a perfect choice

to fulfil the task. The main idea of the boosting algorithm is to choose and combine weak classifiers one by one to form a strong detector. During the iterations, weights of each training sample are modified according to the classification result of the selected weak classifiers. As the iteration goes, most of the training samples have been correctly classified by the already selected weak classifiers, accordingly the weight of these samples will decrease, and more attention will be paid to the high weighted i.e. the misclassified instances. However, as the values of MB-LGP are non-metric, common weak classifiers like threshold function may not fit the need of the MB-LGP features. In order to solve this problem, Adaboost learning based on a 256-dimensional look up table was chosen. Here is the description of the design of the weak classifier.

$$s_m(\alpha) = \begin{cases} a_0, & \alpha = 0 \\ \cdots & \\ a_j, & \alpha = j \\ \cdots & \\ a_{255}, & \alpha = 255 \end{cases}$$

where α is the value of the extracted MB-LGP feature, $a_j (j = 1, 2, \ldots, 256)$ is a confidence weight indicating that whether the checked window contains a face or not. The method of weak classifier training by Adaboost learning can refer to that in [7].

2.3 Classifier Construction

To improve the detection accuracy and reduce the calculation, a cascaded classifier of decision trees and Extreme learning machine is proposed. Cascade classifiers are perfect choices for local features, for instance Haar-like features and MB-LGP, as it can remove a large amount of obvious non-face image blocks with weak classifiers whose computation speed is quite fast and remain the sophisticated image blocks to the relatively time consuming yet high-precision classifiers. The early rejection of non-face images is the key to the high speed of this classifier, because most of the areas in an image are non-face areas. In our 4-stage cascade classifiers, the first three classifiers are simple multi-branch decision trees and the last one is an ELM classifier.

Extreme learning machine I. Introduction of Extreme learning machine
Extreme learning machine (ELM) is an emergent learning algorithm for single hidden layer feedforward neural networks (SLFNs) which randomly chooses its input weights and analytically determines its output weights. The essence of ELM is the un-tuned hidden layer and the bi-objective of the smallest training error as well as smallest norm of out put weight. It maintains a good generalization performance while extremely improves the computing speed of the feedforward neural networks. The application of moore-penrose generalized inverse matrix to deter-

mine the output weights has avoided many problems caused by gradient descent learning method, such as local minima, too much iteration times and the selection of performance index and learning rate. Here is the algorithm theory of ELM. Randomly choose N distinct samples (x_i, t_i), where $x_i = \{x_{i1}, x_{i2}, \ldots, x_{in}\}^T \in R^n$ and $t_i = \{t_{i1}, t_{i2}, \ldots, t_{im}\}^T \in R^m$, the standard SLFNs with hidden neurons and activation function $g(x)$ can select parameters of β_i, w_i, b_i to approximate these N samples with zero error.

$$f_{\tilde{N}}(x_j) = \sum_{i=1}^{\tilde{N}} \beta_i g(w_i, b_i, x_j) = t_j \quad \left(j = 1, 2, \ldots, \tilde{N}\right)$$

The equation can be abbreviate as

$$H\beta = T$$

where

$$H(w_1, \cdots, w_{\tilde{N}}, b_1, \cdots, b_{\tilde{N}}, x_1, \cdots, x_N) = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_{\tilde{N}} \cdot x_1 + b_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ g(w_1 \cdot x_N + b_1) & \cdots & g(w_{\tilde{N}} \cdot x_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m} \quad and \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m}$$

H is called the hidden layer output matrix of the neural network, whose ith row is the output of the ith hidden node.$w_i = [w_{i1}, w_{i2}, \cdots, w_{im}]^T \in R^m$ $\left(i = 1, 2, \cdots \tilde{N}\right)$ is the connection weight vector between the ith hidden node and the output node, b_i is the threshold of the ith hidden nodes. As it is shown in the related work of G.B.Huang [15], unlike traditional SLFNs, if the activation function is infinite differential, the input weight and the hidden layer biases of SLFNs can be arbitrarily given. So the only work of training a SLFN is to find a least-squares solution $\tilde{\beta}$ of the linear system $H\beta = T$. According to [16] the finding solution can be obtained by:

$$\tilde{\beta} = H^* T$$

Considering that the number of our training samples (250000) is much larger than the dimensionality of the feature space, we decide to select the alternative solution of ELM [17]. Consequently, the decision function of ELM classifier is

$$f(x) = sign\left(h(x)\left(\frac{1}{C} + H^T H\right)^{-1} H^T T\right)$$

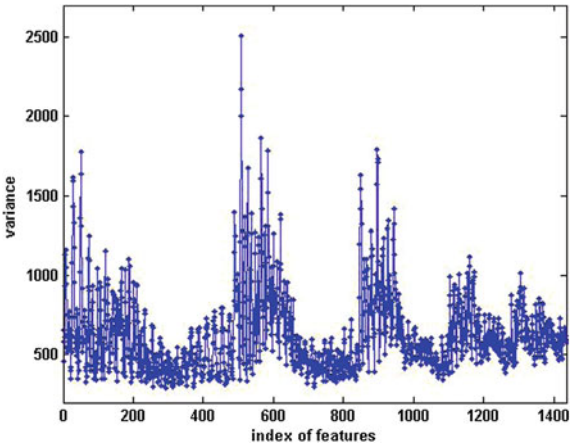

Fig. 4 The variance of histograms of each feature

where C is the error cost which controls the punishment for the classification error of samples.

II. ELM Training

Considering that the value of MB-LGPs are non-metric, the extracted features can not be sent to the learning machine for the training and testing task directly. To solve the similar problem, [18] used histograms to conduct the edge orientation of image blocks, which was metric, to represent the local texture and global shape of images. In this article we first select distinctive features and calculate the frequency of occurrence of the feature values in a 24×24 image window to form a histogram. The definition of the histograms are as follow:

$$H(i) = \sum_{x \in S} I(s(x) = i) \quad (i = 1, \cdots, 255)$$

where $H(\bullet)$ is a 255 dimensional vector which indicate the frequency of each feature value, S is the selected feature set, $I(\bullet)$ is a function which values one if the equation is true and values zero otherwise. Then, the ready-processed histogram data was sent to the ELM to train the strong classifier of the last stage.

a. Feature selection

In the first step, the total histogram of every dimension of the MB-LGP feature was calculated in face and non-face samples set separately. As a result, two 1436×255 matrixes H_f and H_{nf} were formed. Then do the subtraction between the two matrixes and take the absolute value: $H = \left| H_f - H_{nf} \right|$. Calculate the variance of each row vector of H (see Fig. 4). The distinctive ability of each MB-LGP feature was judged by the variance of the corresponding row. The larger the variance is the more discriminating the feature is.

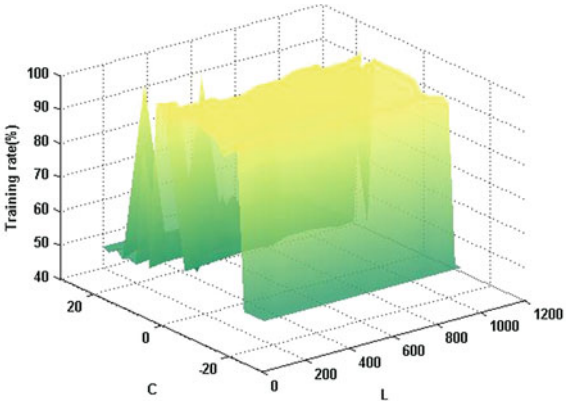

Fig. 5 Training performance of ELM

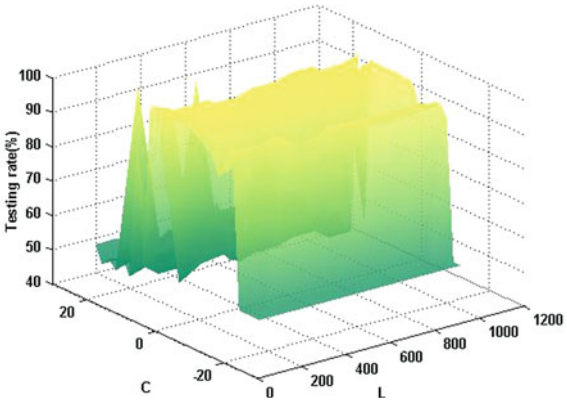

Fig. 6 Testing performance of ELM

b. Training

Sort the features by the variance value of each row of H in a descending order to form a candidate queue. The first 100 features with greater distinctive ability were added to the selected set at the very beginning, after that, five features were moved to the selected set at a time orderly until the training and testing accuracy get higher than 99.5 %. Each time the ELM was training, the parameters of C and L were tuned in the range of $\left\{2^{-24}, 2^{-23}, \cdots, 2^{24}, 2^{25}\right\}$ and $\{100, 200, \cdots, 1000\}$ separately. Figures 5 and 6 shows the performance of training and testing of ELM, here C is the exponent of error costs.

The selected feature set met the requirement at the feature number of 215.

Combine the multi-branch decision trees together with the ELM Our four stage classifier was formed by combining the multi-branch decision trees with the ELM. Detailed detection procedure works as follows: (1) Contract the detecting image into different scales to form an image pyramid [19], so as to acquire faces of different sizes. In our experiment, original images were repeatedly reduced in a ratio of 0.89.

(2) Extract features from the image sequence by a 24×24 detection window with different mappings according to a different stage of classifier. (3) Send the features to classifier and do the classification. (4) Merge the adjacent rectangles.

3 Experiment and Results

To evaluate the discriminating ability of MB-LGP and judge the performance of the proposed method, two experiments had been conducted in this section: (1) Comparing the discrimination capacity of MB-LGP with Haar-like features, LBP and LGP. (2) Applying the proposed detector to CMU+MIT face database. In the experiment, 10, 000 sample faces with different illumination conditions, face expressions, orientations and pose (in a range of $[-15°, +15°]$) were collected on the internet. The clipping of face areas were determined by the position and distance between eyes. Let the coordinate of the left eye to be (0, 0), the distance between both eyes to be α, then the up-right square area with the side length of 2α whose top left corner lay at $(-0.5\alpha, -0.5\alpha)$ was considered to be the face area and was clipped to be the final face sample. We gather the other 90, 000 samples by slightly enlarge the face area, shifting the sampling window horizontally and vertically as well as rotating the sampling window by $\pm 15°$. 150, 000 non face samples were collected by randomly scan different images of scenery and buildings without human faces. At last, all the sample images were resized to the scale of 24×24.

3.1 Feature comparison

In this part of experiment, 100,000 face samples and 150,000 non-face samples were randomly divided into two equal parts, one for training and the other for testing. Boosting classifiers of 30 features was trained to compare the distinguishing ability of the four representing method. The same experiment had been carried out for ten times, the average experimental data was used to do the judgement. As we can see in Fig. 7, MB-LGP's error rate was relatively lower than LGP and LBP and much better than Haar-like features. This is the best proof of the discriminative ability of MB-LGP.

When it came to the ROC curves (see Fig. 8) of the four features, MB-LGP also did a better job than the three other features. It shows that MG-LGP has detection rate of 72.6 % at the very beginning, comparing with the 70 % of LGP, 64.2 % of LBP and 55 % of Haar-like features MG-LGP acted better than the three other features. When the false positive gets higher, the performance of MB-LGP keeps better than the three other features.

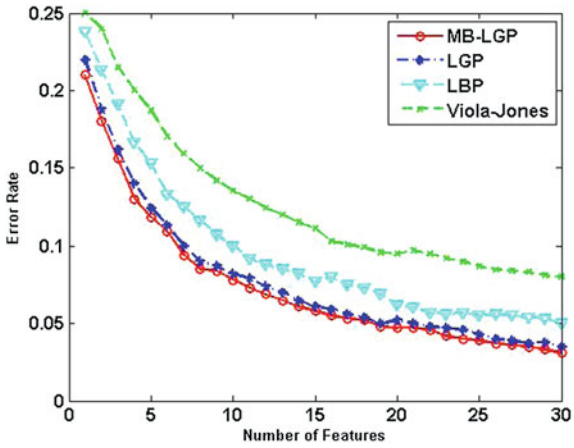

Fig. 7 The tendency of error rate of four features when number of weak classifiers increases

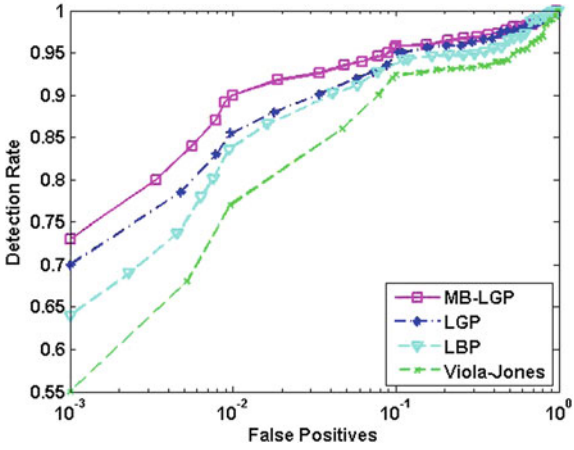

Fig. 8 ROC curves of MB-LGP, LGP, LBP and Haar-like classifiers

3.2 Experiment on CMU+MIT Face Database

In this section, all the 250,000 faces and non-face samples were used to train the cascaded classifiers. In each training stage, the detection rate were controlled to be 99 % while the false positive rate to be 4 %. During the training, the false face samples and the correct non-face samples were weeded out, the remaining samples were selected to train the strong classifier for the next stage. Here is the information

Table 1 Stages and feature numbers of cascaded classifiers

Features and Classifiers	MB-LGP+ELM	MB-LGP	LGP	LBP
Number of stages	4	5	5	6
Number of features	398	423	478	542

Table 2 Face detection rate on CMU+MIT database for different detectors

False positive	2	3	4	8	10	42	45	150	167	179	187	198
MB-LGP+ELM			0.88				0.918		0.935			
MB-LGP	0.86					0.908				0.923		
LGP		0.84										0.92
LBP				0.79				0.903				
Viola&Jones					0.8						0.91	

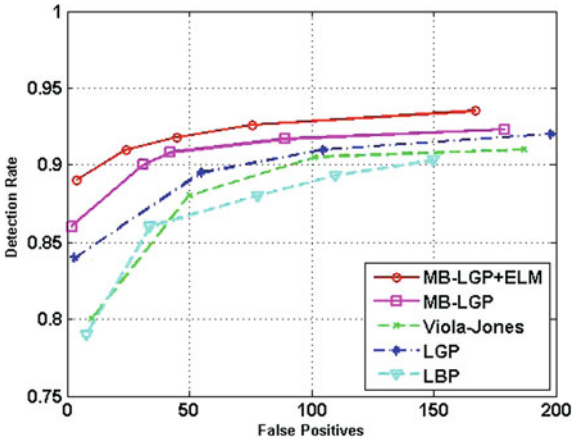

Fig. 9 ROC curves using CMU+MIT database

of the cascaded classifiers (Table 1). Part of the detection result of the five detectors are listed in Table 2. As expected, due to the distinctive capability of MB-LGP and ELM our detector had better performance than the others (Fig. 9).

4 Conclusion

An extension of the newly local feature LGP called MB-LGP was introduced in this article. Through the introduction and experiment we have proved its great invariance against both local and globe variation, perfect distinctive capability and low dimensionality. Moreover, based on the newly discovered strong feature, a cascade

face detector with four stages which were constructed by combining the boosting classifiers with Extreme Learning Machine was proposed. The experiment carried on the famous CMU+MIT face database had proved the great detection ability of the cascaded classifier.

References

1. T. Ojala, M. Pietikainen, T. Maenpaa, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. **7**(13), 971–987 (2002)
2. H. Jin, Q. Liu, H. Lu et al., Face detection using improved LBP under bayesian framework. *Proceedings of Third International Conference on Image and Graphics*. IEEE, pp. 306–309 (2004)
3. X. Tan, B. Triggs, Enhanced local texture feature sets for face recognition under difficult lighting conditions. *Analysis and Modeling of Faces and Gestures* (Springer, Berlin, 2007), pp. 168–182
4. S. Liao, S. Chung, Face recognition by using elongated local binary patterns with average maximum distance gradient magnitude. *Computer Vision-ACCV 2007* (Springer, Berlin, 2007), pp. 672–679
5. G. Zhao, M. Pietikainen, Dynamic texture recognition using local binary patterns with an application to facial expressions. IEEE Trans. Pattern Anal. Mach. Intell. **29**(6), 915–928 (2007)
6. L. Paulhac, P. Makris, Y. Ramel, Comparison between 2D and 3D local binary pattern methods for characterisation of three-dimensional textures. *Image Analysis and Recognition* (Springer, Berlin, 2008), pp. 670–679
7. Jun. B, Kim. D, Robust face detection using local gradient patterns and evidence accumulation. Pattern Recogn. **45**(9), 3304–3316 (2012)
8. P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001*. IEEE, 1: I-511-I-518, vol. 1 (2001)
9. M. Jones, P.Viola, Fast multi-view face detection. Mitsubishi Electric Research Lab, 2012. TR-20003-96, vol. 3 (2003)
10. J. Xu, Y. Dou, Z. Pang, A reconfigurable architecture for rotation invariant multi-view face detection based on a novel two-stage boosting method. EURASIP J. Adv. Sig. Process. **2009**, 54 (2009)
11. R. Xiao, L. Zhu, H.J. Zhang, Boosting chain learning for object detection. *Proceedings of Ninth IEEE International Conference on Computer Vision*. IEEE, pp. 709–715 (2003)
12. J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). Ann. Stat. **28**(2), 337–407 (2000)
13. S.Z. Li, Z.Q. Zhang, Floatboost learning and statistical face detection. IEEE Trans. Pattern Anal. Mach. Intell. **26**(9), 1112–1123 (2004)
14. K. Zeng, Y. Tang, F. Liu, Parallization of Adaboost algorithm through hybrid MPI/openMP and transactional memory. *19th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. IEEE, pp. 94–100 (2011)
15. Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: a new learning scheme of feedforward neural networks. *Proceedings of IEEE International Joint Conference on Neural Networks*. IEEE, vol. 2, pp. 985–990 (2004)

16. H.A. Rowley, S. Baluja, T. Kanade, Neural network-based face detection. IEEE Trans. Pattern Anal. Mach. Intell. **20**(1), 23–38 (1998)
17. G.B. Huang, H. Zhou, X. Ding et al., Extreme learning machine for regression and multiclass classification. IEEE Trans. Syst. Man Cybern. Part B Cybern. **20**(1), 23–38 (1998)
18. Y.W. Li, E. Zhu, R.Y. Chen et al., A vehicle classification approach based on edge orientation histograms. J. Comput. Inf. Syst. **8**(16), 6979–6989 (2012)
19. R. Zabih, J. Woodfill, Non-parametric local transforms for computing visual correspondence. *Computer VisionECCV'94* (Springer, Berlin, 1994), pp. 151–158

Freshwater Algal Bloom Prediction by Extreme Learning Machine in Macau Storage Reservoirs

Inchio Lou, Zhengchao Xie, Wai Kin Ung and Kai Meng Mok

Abstract Understanding and predicting dynamic change of algae population in freshwater reservoirs is particularly important, as algae-releasing cyanotoxins are carcinogens that would affect the health of public. However, the high complex nonlinearity of water variables and their interactions makes it difficult in modeling its growth. Recently extreme learning machine (ELM) was reported to have advantages of only requirement of a small amount of samples, high degree of prediction accuracy and long prediction period to solve the nonlinear problems. In this study, the ELM-based prediction and forecast models for phytoplankton abundance in Macau Storage Reservoir (MSR) are proposed, in which the water parameters of pH, SiO_2, alkalinity, Bicarbonate (HCO_3^-), dissolved oxygen (DO), total Nitrogen (TN), UV_{254}, turbidity, conductivity, nitrate, total nitrogen (TN), orthophosphate (PO_4^{3-}), total phosphorus (TP), suspended solid (SS) and total organic carbon (TOC) selected from the correlation analysis of the 23 monthly water variables were included, with 8 years (2001–2008) data for training and the most recent 3 years (2009–2011) for testing. The modeling results showed that the prediction and forecast (based on data on the previous 1st, 2nd, 3rd and 12th months) powers were estimated as approximately 0.83 and 0.90 respectively, showing that the ELM is an effective new way that can be used for monitoring algal bloom in drinking water storage reservoir.

Keywords Algal bloom · Phytoplankton abundance · Extreme leaning machine · Prediction and forecast models

I. Lou · Z. Xie (✉) · K. M. Mok
Faculty of Science and Technology, University of Macau, Macau SAR, China
e-mail: zxie@umac.mo

W. K. Ung
Laboratory and Research Center, Macao Water Co. Ltd., Macau SAR, China

F. Sun et al. (eds.), *Extreme Learning Machines 2013: Algorithms and Applications*, Adaptation, Learning, and Optimization 16, DOI: 10.1007/978-3-319-04741-6_8,

1 Introduction

Freshwater algal bloom is one of water pollution problem that occurs in eutrophic lakes or reservoirs due to the presence of excessive nutrients. It has been found that most species of algae (also called phytoplankton) can produce various cyanotoxins including *microcytins, cylindrospermopsis* and *nodularin*, which have directly impact on the water treatment processes and consequently affect the health of public [1]. Thus it is of great importance to understand the population dynamics of algae in the raw water storage units. However, modeling the algae population in such a complicated system is a challenge, as the physical, chemical and biological processes as well as the interaction among them are involved, resulting in the highly nonlinear relationship between phytoplankton abundance and various water parameters.

Computational artificial intelligence techniques have been developed as the efficient tools in recent years for predicting (without considering time series effect) or forecasting (considering time series effect) algal bloom. Previous studies [2] have used the principle component regression (PCR), i.e., principal component analysis (PCA) followed by multiple linear regressions (MLR), to predict chlorophyll-a levels, the fundamental index of phytoplankton. However, the intrinsic problem of PCR is that the variables data set used as the input of the model have high complex nonlinearity, expecting that PCR alone is inadequate for prediction and the prediction results were unsatisfactory. With the development of artificial intelligence models, artificial neural network (ANN) such as back propagation (BP) was applied to predict the algal bloom by assessing the eutrophication and simulating the chlorophyll-a concentration. ANN is a well-suited method with self-adaptability, self-organization and error tolerance, which is better than PCR for non-linear simulation. ANN has been used for predicting the chlorophyll concentration [3–5]. However, this method has such limitations as requirement of a great amount of training data, difficulty in tuning the structure parameter that is mainly based on experience, and its "black box" nature that is difficult to understand and interpret the data [2, 6].

Considering the drawbacks of the both methods, recently extreme learning machine (ELM) is thought as the best solution. ELM is a simple and efficient learning algorithm that was developed recently. In the name of ELM, extreme means that its learning speed is extremely fast while it has higher generalization than the gradient-descent based learning [7]. Furthermore, ELM can be used to solve issues like local minima, improper learning rate and over-fitting which are very possible in traditional ANN [4, 7]. Examples [3–5, 7] also showed that ELM possesses a superior performance than other conventional algorithms on different benchmark problems from both regression and classification areas. There are some existing works [8–10] with using ELM and through these works it can be seen that ELM could have a very good performance for some engineering applications. By far, as the best knowledge of authors, there is no existing application of using ELM on prediction or forecast the phytoplankton abundance in algal blooms.

In this study, it is attempted to develop an ELM-based predictive model to simulate the dynamic change of phytoplankton abundance in Macau Reservoir given a variety

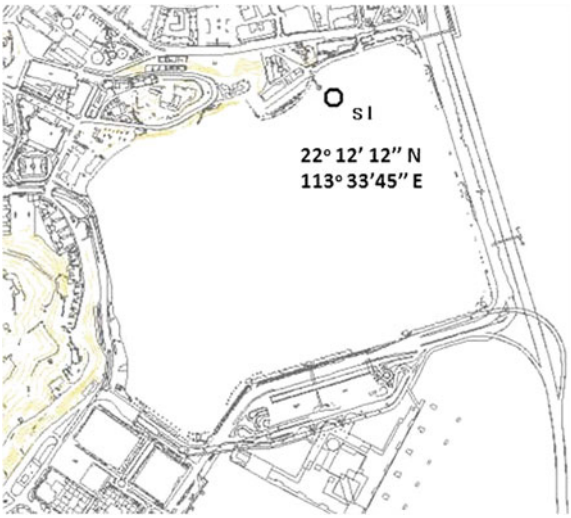

Fig. 1 Location of the MSR

of water variables. The measured data from 2001 to 2011 were used to train and test the model. The present study will lead to better understanding of the algal problems in Macau, which will help to develop later guidelines for forecasting the onset of algae blooms in raw water resources.

2 Materials And Methods

2.1 MSR and Water Parameters Measurement

Macau is situated 60 km southwest of Hong Kong, and experiences a subtropical seasonal climate that is greatly influenced by the monsoons. The difference of temperature and rainfall between summer and winter are significant though not great. Macau Main Storage Reservoir (MSR) (Fig. 1), located in the east part of Macau peninsula, is the biggest reservoir in Macau with the capacity of about 1.9 million m^3 and the water surface area of 0.35 km^2. It is a pumped storage reservoir that receives raw water from the West River of the Pearl River network, and can provide water supply to the whole areas of Macau for about 1 week. MSR is particularly important as the temporary water source during the salty tide period when high salinity concentration is caused by intrusion of sea water to the water intake location. In recent years, there were reports (Macao Water Co. Ltd., unpublished data) that the reservoir experienced algal blooms and the situation appeared to be worsening.

Macau Water Supply Co. Ltd. is responsible for water-quality monitoring and management. Location in the inlet of the reservoir was selected for sampling. Samples were collected in duplicate monthly from May 2001 to February 2011 at 0.5 m from

the water surface. A total of 23 water quality parameters, including hydrological, physical, chemical and biological parameters, were monitored monthly. Precipitation was obtained from Macau Meteorological Center (http://www.smg.gov.mo/www/te_smgmail.php). Imported volume, exported volume and water level were recorded by the inlet and outlet flow meters, based on which the hydraulic retention time (HRT) can be calculated. Turbidity, temperature, pH, conductivity, chloride (Cl^-), sulfate (SO_4^{2-}), silicon (SiO_2), alkalinity, bicarbonate (HCO_3^-), dissolved oxygen (DO), ammonium (NH_4^+), nitrite (NO_2^-), nitrate (NO_3^-), total nitrogen (TN), phosphorus (PO_4^{3-}), total phosphorus (TP), suspended solid, total organic carbon (TOC) and UV_{254} and iron (Fe) were measured according to the standard methods [11, 12]. The phytoplankton samples were fixed using 5 % formaldehyde and transported to laboratory for microscopic counting.

In this work, correlation analysis was conducted to identify the water parameters which were significantly correlated with phytoplankton abundance. Only the parameters with the correlation coefficients greater than 0.3, are selected as inputs in the ELM models. It was also noted that the parameters selected in forecast models are different from those in the prediction models, as the water parameters in previous data were also used in the correlation analysis. In our study, two types of forecast models were used, depending on the monthly data used as the inputs. Forecast model 1 was based on the last 3 months data, while forecast model 2 was based on the last 3 months data as well as the previous 12th month data, i.e., the previous 1st, 2nd and 3rd and 12th month data. The purpose of adding the previous 12th month data in the forecast model 2 is to take the historical effect the last year that have similar environmental conditions, as the environmental conditions, such as temperature, influence the growth of phytoplankton.

2.2 Extreme Learning Machine

ELM originally was proposed as a learning scheme for single-hidden-layer feed-forward neural networks (SLFNs). Then, it was extended to the generalized SLFNs where the hidden layer needs not be neuron alike [13, 14]. In the past, gradient descent based approaches were used for feed-forward neutral networks, and all parameters need to be tuned which usually take a long time. While for ELM which the basic idea is that, the model has only one hidden layer, and the parameters of this hidden layer, including the input weights and biases of the hidden nodes, need not to be tuned. On the contrary, these hidden nodes parameters are assigned randomly, which means that they may be independent of the training data [14]. After these input weights and hidden layer biases are assigned randomly, SLFNs can be treated as linear system and output weights which link hidden layer to the output layer can be calculated using generalized inverse operation [15–17]. References [3, 4, 7] proposed and proved the theory of ELM. In order to make it clear on ELM and its application in water treatment prediction, here the fundamental theory of ELM in [7] will be briefly re-introduced first as follows:

Consider a training data set $\boldsymbol{D}$ of N arbitrary distinct samples $(\boldsymbol{x}_i, \boldsymbol{t}_i)$, where $\{\boldsymbol{x}_i\} \in R^m$ is the m × 1 input vector and $\{\boldsymbol{t}_i\} \in R^n$ is the n × 1 target vector. Standard SLFNs with $\tilde{N}$ nodes and activation function $g(x)$ are mathematically modeled a

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i\left(x_j\right) = \sum_{i=1}^{\tilde{N}} \beta_i g\left(\boldsymbol{w}_i, b_i, \boldsymbol{x}_j\right) = \boldsymbol{o}_j, \qquad 1 \le j \le N \tag{1}$$

where $\boldsymbol{w}_i$ is weight vector connecting the $\boldsymbol{i}$th hidden node and the input nodes, β_i is the weight vector connecting the $\boldsymbol{i}$th hidden node and the output nodes, and $\boldsymbol{b}_i$ is the threshold of the $\boldsymbol{i}$th hidden node.

Since the goal is to find the relation between $\boldsymbol{x}_i$ and $\boldsymbol{t}_i$, if the SLFNs can approximate the training data with zero error (i.e., $\sum_{j=1}^{\tilde{N}} \left\|\boldsymbol{o}_j - \boldsymbol{t}_j\right\| = 0$), then there exists β_i, $\boldsymbol{w}_i$, and $\boldsymbol{b}_i$ such that Eq. (2) is satisfied.

$$\sum_{i=1}^{\tilde{N}} \beta_i g\left(\boldsymbol{w}_i, b_i, \boldsymbol{x}_j\right) = \boldsymbol{t}_j, \quad 1 \le j \le \boldsymbol{N} \tag{2}$$

The above N equations can be written compactly as

$$\boldsymbol{H}\beta = \boldsymbol{T} \tag{3}$$

where

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{h}\left(\boldsymbol{x}_1\right) \\ \vdots \\ \boldsymbol{h}\left(\boldsymbol{x}_N\right) \end{bmatrix} = \begin{bmatrix} g\left(\boldsymbol{w}_1, b_1, \boldsymbol{x}_1\right) & \cdots & g\left(\boldsymbol{w}_{\tilde{N}}, b_{\tilde{N}}, \boldsymbol{x}_1\right) \\ \vdots & \ddots & \vdots \\ g\left(\boldsymbol{w}_1, b_1, \boldsymbol{x}_N\right) & \cdots & g\left(\boldsymbol{w}_{\tilde{N}}, b_{\tilde{N}}, \boldsymbol{x}_N\right) \end{bmatrix}_{N \times \tilde{N},} \tag{4}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_N^T \end{bmatrix}_{\tilde{N} \times n} \text{ and } \boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_N^T \end{bmatrix}_{N \times n} \tag{5}$$

$\boldsymbol{H}$ is called the hidden layer output matrix of SLFN. $\boldsymbol{h}(\boldsymbol{x}) = g\left(\boldsymbol{w}_1, b_1, \boldsymbol{x}\right), \ldots,$ $g\left(\boldsymbol{w}_{\tilde{N}}, b_{\tilde{N}}, \boldsymbol{x}\right)$ is called the hidden layer feature mapping. The $\boldsymbol{i}$th column of $\boldsymbol{H}$ is the $\boldsymbol{i}$th hidden node output with respect to inputs $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$. The $\boldsymbol{i}$th row of $\boldsymbol{H}$ is the hidden layer feature mapping with respect to the $\boldsymbol{i}$th input $\boldsymbol{x}_i$.

According to the proofs in [3, 4], if the activation function is infinitely differentiable, the input weight vectors $\boldsymbol{w}_i$ and hidden layer biases $\boldsymbol{b}_i$ can be randomly assigned. Moreover, these parameters are not necessarily tuned and the hidden layer output matrix $\boldsymbol{H}$ can actually remain unchanged once random values have been assigned in the beginning of learning.

Different from traditional learning algorithms, ELM tends to reach not only the smallest training error but also the smallest norm of output weights [18]:

$$\text{Minimize: } \|\boldsymbol{H}\beta - \boldsymbol{T}\|^2 \text{ and } \|\beta\| \tag{6}$$

Then, if the number $\tilde{N}$ of hidden neurons is equal to the number N of distinct training samples (i.e., $\tilde{N} = N$), the matrix $\boldsymbol{H}$ is square and invertible, which means that the output weights β can be analytically calculated by simply inverting $\boldsymbol{H}$, and thus the SLFNs can approximate these training samples with zero error. However, most of the times the number of hidden nodes is much less than the number of distinct training samples (i.e., $\tilde{N} \ll N$), and thus $\boldsymbol{H}$ is a non-square matrix and there may not exist β_i, $\boldsymbol{w}_i$ and b_i, and Eq. (3) cannot be satisfied. Fortunately, since $\boldsymbol{w}_i$ and $\boldsymbol{b}_i$ are fixed, Eq. (3) becomes a linear system, and the smallest norm least square method can be used instead of the standard optimization method to estimate the output weights.

$$\beta = \boldsymbol{H}^{\dagger}\boldsymbol{T} \tag{7}$$

where $\boldsymbol{H}^{\dagger}$ is the Moore–Penrose pseudoinverse of matrix $\boldsymbol{H}$ [16], which can be calculated using the orthogonal projection method [17]:

$$\boldsymbol{H}^{\dagger} = \left(\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\right)^{-1}\boldsymbol{H}^{\mathrm{T}} \text{ when } \boldsymbol{H}^{T}\boldsymbol{H} \text{ is nonsingular} \tag{8}$$

$$or\ \boldsymbol{H}^{\dagger} = \boldsymbol{H}^{\mathrm{T}}\left(\boldsymbol{H}\,\boldsymbol{H}^{\mathrm{T}}\right)^{-1} \text{ when } \boldsymbol{H}\,\boldsymbol{H}^{\mathrm{T}} \text{ is nonsigular} \tag{9}$$

where the superscript T means matrix transposition.

Based on this learning algorithm, the training time can be extremely fast because only three calculation steps are required: 1. randomly assign hidden nodes parameters; 2. calculate the hidden layer output matrix $\boldsymbol{H}$; 3. calculate the output weight β. Moreover, since the output weights are calculated analytically using inverse matrix, it ensures that the results are global and hence better prediction accuracy and generalization performance can be achieved. After training, the output function of ELM for an unseen vector $\boldsymbol{X}$ (take one output node case as an example) can be expressed as:

$$f\,(\boldsymbol{X}) = \boldsymbol{h}\,(\boldsymbol{X})\,\beta \tag{10}$$

2.3 Performance Indicators

The performance of models was evaluated using the following indicators: square of correlation coefficient (R^2) that provides the variability measure for the data reproduced in the model; mean absolute error (MAE) and root mean square error (RMSE) that measure residual errors, providing a global idea of the difference between the observation and modeling. The indicators were defined as below by Eq. 11–15.

$$R^2 = 1 - \frac{F}{F_o} \tag{11}$$

$$F = \sum \left(Y_i - \hat{Y}_i \right)^2 \tag{12}$$

$$F_o = \sum \left(Y_i - \overline{Y_i} \right)^2 \tag{13}$$

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} \left(\hat{Y}_i - Y_i \right)^2 \tag{14}$$

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(\hat{Y}_i - Y_i \right)^2} \tag{15}$$

where n is the number of data; Y_i and $\overline{Y_i}$ are observation data and the mean of observation data, respectively, and $\hat{Y}_i$ is the modeling results Table 1.

3 Results and Discussion

The correlation of $\log_{10}$ phytoplankton and water parameters for forecast model and prediction model were shown in Table 2. Parameters with correlation coefficients greater than 0.3 (highlighted in bold) will be retained in the models. It was also noted that the parameters selected in forecast models are different from those in the prediction models, as the water parameters in previous data (past record) were also used in the correlation analysis. In the forecast models of ELM, phytoplankton abundance (t) is a function of water parameter (t-1), water parameter (t-2) and water parameters (t-3), where t-1, t-2, t-3 and t-12 represent the 1, 2, 3 and 12 months prior to time t. Thus there were only 9 parameters used in the prediction models and 23 time-lagged parameters selected for the forecast models.

After the correlation analysis, it comes to the testing of the models invoked two parts, the accuracy performance and the generalization performance. Accuracy performance is to test the capability of the model to predict the output for the given input set that originally used to train the model, while generalization performance is to test the capability of the model to predict the output for the given input sets that were not in the training set. In order to prevent the model that is memorizing the inputs instead of generalized learning, both performance checks need to be considered. In the present research, the performance indexes for ELM-based models were averaged with 50 runs.

In the application of ELM in this work, for the predication models, after the correlation analysis, 9 parameters such as pH, SiO_2 are selected as the independent variables, and phytoplankton abundance is selected as the induced variable (target value). Then, the data from May of 2005 to December of 2008 are used to train the model, and data from January of 2009 to February of 2011 are used to test the model. In the training process, the cross-validation approach as mentioned previously is adopted to obtain the optimal combination of parameters for the testing. Specifically, the training data are divided into 10 about the same size groups that 9 groups for

Table 1 Correlation analysis of prediction and forecast model

Parameters	Prediction model	Forecast model Time lagged (month)			
		t-1	t-2	t-3	t-12
Turbidity	−0.03	0.00	−0.01	−0.06	−0.25
Temperature	0.19	0.21	0.19	0.14	0.22
pH	**0.49**	**0.42**	**0.38**	**0.33**	**0.33**
Conductivity	−0.08	0.01	0.14	0.21	−0.24
Cl^-	0.01	0.10	0.22	0.28	−0.16
SO_4^{2-}	−0.03	0.03	0.14	0.22	−0.28
SiO_2	**0.33**	**0.31**	0.16	0.04	−0.08
Alkalinity	**−0.34**	**−0.30**	−0.21	−0.12	**−0.36**
HCO_3^-	**−0.46**	**−0.40**	**−0.32**	−0.24	**−0.38**
DO	**0.39**	**0.35**	**0.34**	**0.31**	0.18
NO_3^-	−0.29	−0.22	−0.22	−0.15	**−0.35**
NO_2^-	−0.10	−0.08	−0.02	0.03	−0.23
NH_4^+	0.11	0.10	0.08	0.25	0.05
TN	**0.68**	**0.60**	**0.53**	**0.46**	0.23
UV_{254}	**0.56**	**0.55**	**0.48**	**0.47**	−0.07
Fe	−0.14	−0.06	−0.04	−0.08	−0.27
PO_4^{3-}	0.02	0.06	0.06	0.03	0.11
TP	0.08	0.05	0.02	0.00	−0.21
Suspended solid	**0.31**	**0.35**	**0.31**	0.23	−0.10
TOC	**0.38**	**0.33**	0.29	**0.35**	0.07
HRT	−0.12	−0.11	−0.13	−0.16	0.10
Water level	0.13	0.05	0.01	−0.02	0.10
Precipitation	−0.09	0.05	0.11	0.06	−0.05
Phytoplankton abundance	–	**0.82**	**0.71**	**0.62**	0.24

training and the rest 1 group is used to test the model trained by the previous 9 groups' data. Then, this (9 groups training and 1 group testing) is repeated for 9 times (10 times in total). And then, parameters of the one process which has the best testing performance in these 10 repeats will be used as the optimal parameters combination in the 'real' testing process which has the data from January of 2009 to February of 2011. The forecast model basically follows the same steps of the prediction model, while the only difference between these two models is that effect of time series is included in the forecast model. So, in the forecast model, only the previous 3 months' data are included in the training process.

The performance of prediction and forecast models were shown in Table 2. The results indicated that the ELM were successful in the prediction and forecast phytoplankton abundance in MSR, with the R^2 greater than 0.82 for both training and testing data sets. Compared to the prediction model, ELM had better performance with the R^2 of 0.8637 (0.8702), RMSE of 0.2246 (0.3643), and MAE of 0.1794 (0.2565) for training (testing), suggesting that the historical water parameters including the phytoplankton abundance has effect on the prediction, which can improve the pre-

Table 2 Performance indexes of the prediction and forecast models

Performance index	Prediction model		Forecast model			
	Accuracy performance (Training set)	Generalization performance (Testing set)	Accuracy performance (Training set)		Generalization performance (Testing set)	
			t-1, 2, 3	t-1, 2, 3, 12	t-1, 2, 3	t-1, 2, 3, 12
R^2	0.82492	0.83217	0.8637	0.9009	0.8702	0.9033
RMSE	0.25798	0.30129	0.2246	0.2236	0.3643	0.3166
MAE	0.19976	0.23663	0.1794	0.1597	0.2565	0.2258

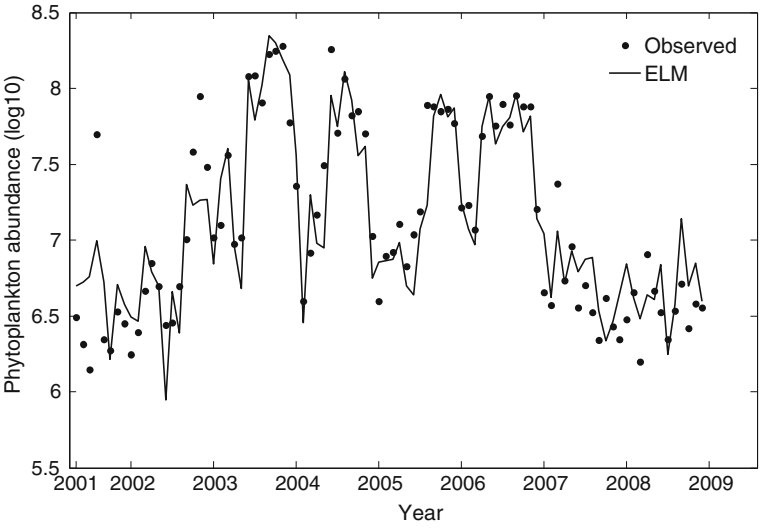

Fig. 2 Observed and predicted phytoplankton level for the training and validation data set of the prediction models

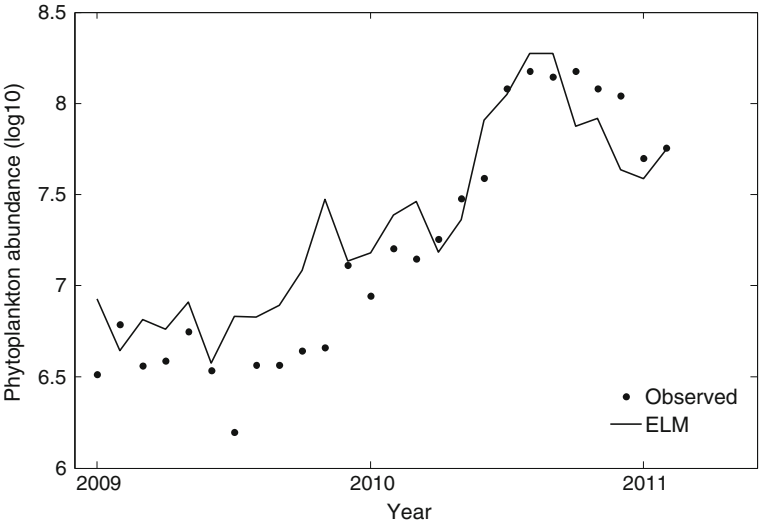

Fig. 3 Observed and predicted phytoplankton level for the testing data set of the prediction models

diction power. Furthermore, when including the previous 12th month data as input in the forecast model, the prediction power of the forecast model can increase up to 0.9 with the RMSE of 0.2236 (0.3166), and MAE of 0.1597 (0.2258) for training (testing). These results further confirmed the historical effects on the model accuracy and generalization performance, and also implied that take the previous 12th month

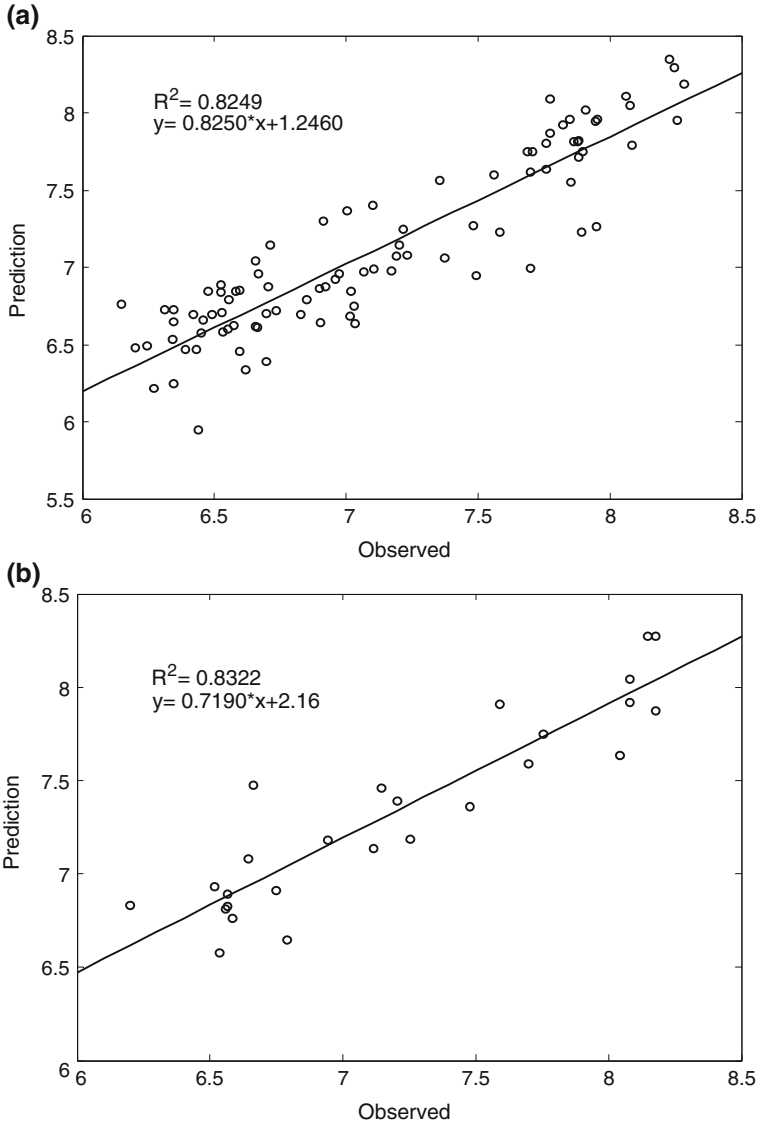

Fig. 4 ELM results for the training and validation (**a**) and testing (**b**) data set of the prediction model

data as memorizing learning can improve the prediction power in the forecast model. Besides, further compared with our previous study [19] for forecast of phytoplankton abundance using support vector machine (SVM) with R^2 of 0.86, the present study using ELM have better prediction power with R^2 of 0.9.

The observed data versus the modeling data were shown in Fig. 4 (prediction model), Fig. 7 (forecast model 1) and Fig. 10 (forecast model 2), and the observed

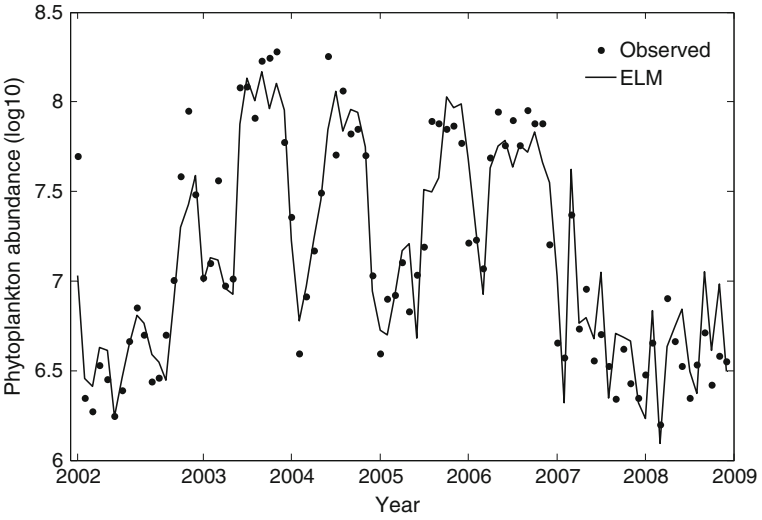

Fig. 5 Observed and predicted phytoplankton level for the training and validation data set of the forecast model 1 that based on the previous 1st, 2nd and 3rd months data

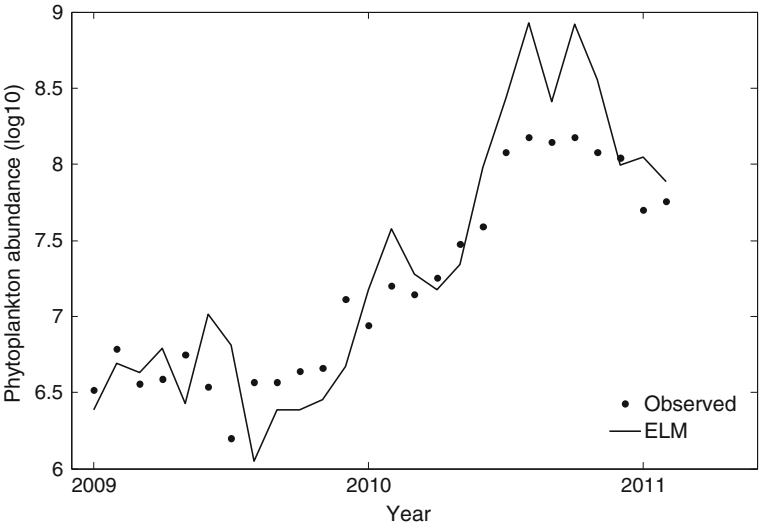

Fig. 6 Observed and predicted phytoplankton level for the testing data set of the forecast model 1 that based on the previous 1st, 2nd and 3rd months data

and modeling phytoplankton abundance change over time were listed in Figs. 2 and 3 (prediction model) and Figs. 5 and 6 (forecast model 1) and Figs. 8 and 9 (forecast model 2). These results confirmed that ELM can handle well the non-linear relationship between water parameters and phytoplankton abundance.

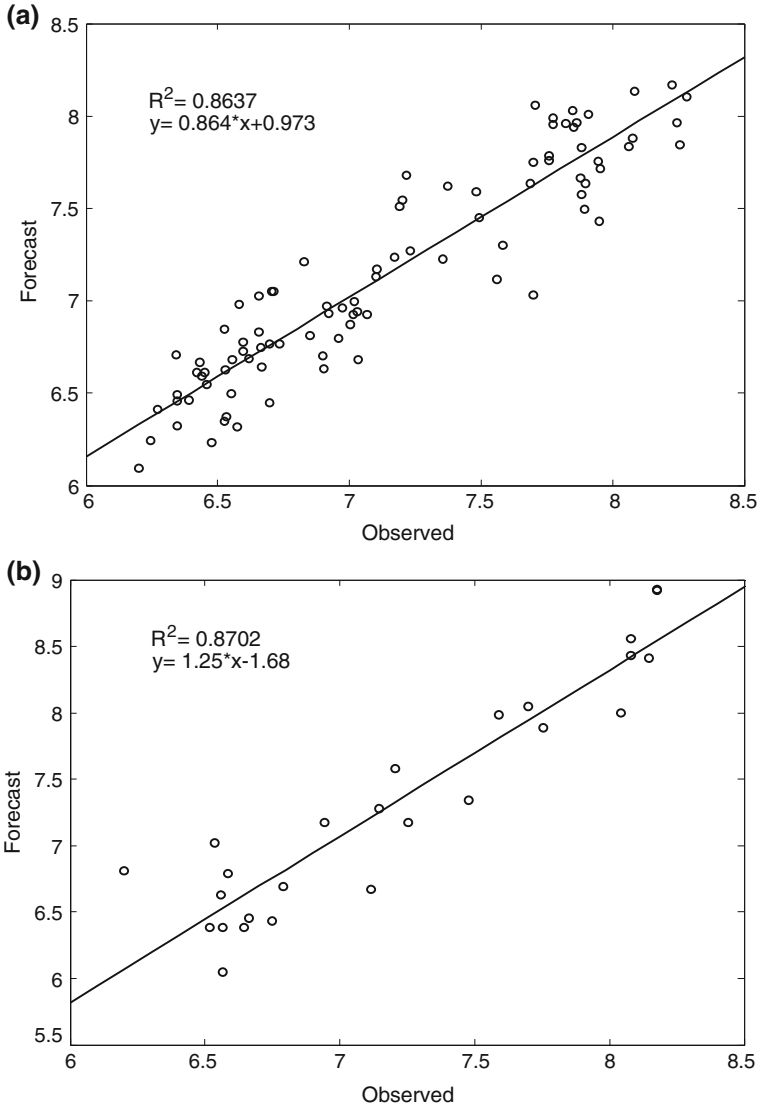

Fig. 7 ELM results for the training and validation (**a**) and testing (**b**) data set of the forecast model 1 that based on the previous 1st, 2nd and 3rd months data

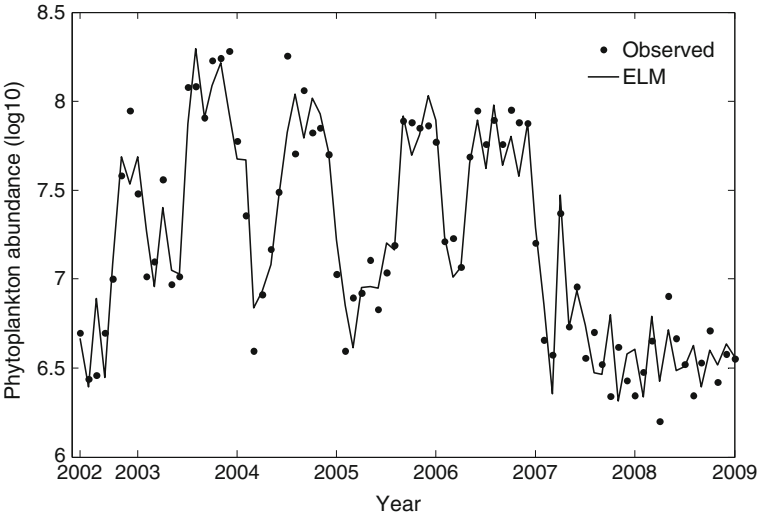

Fig. 8 Observed and forecasted phytoplankton lever for the training and validation data set of the forecast model 2 that based on the previous 1st, 2nd, 3rd and 12th months data

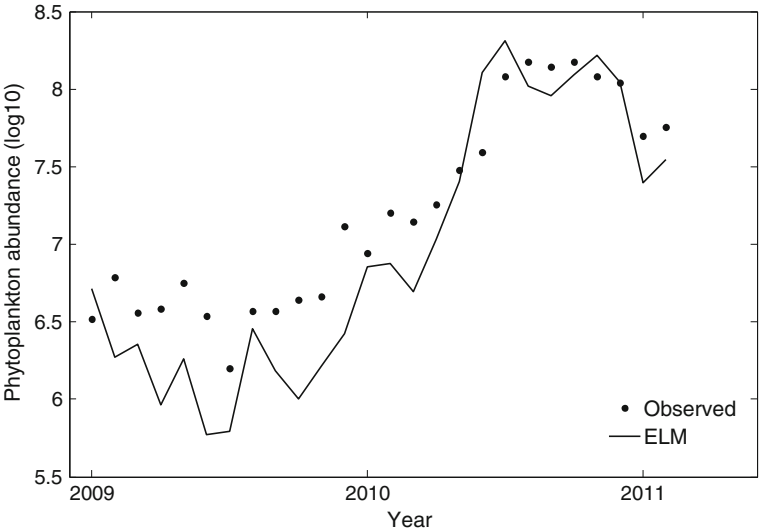

Fig. 9 Observed and predicted phytoplankton level for the testing data set of the forecast model 2 that based on the previous 1st, 2nd, 3rd, and 12th months data

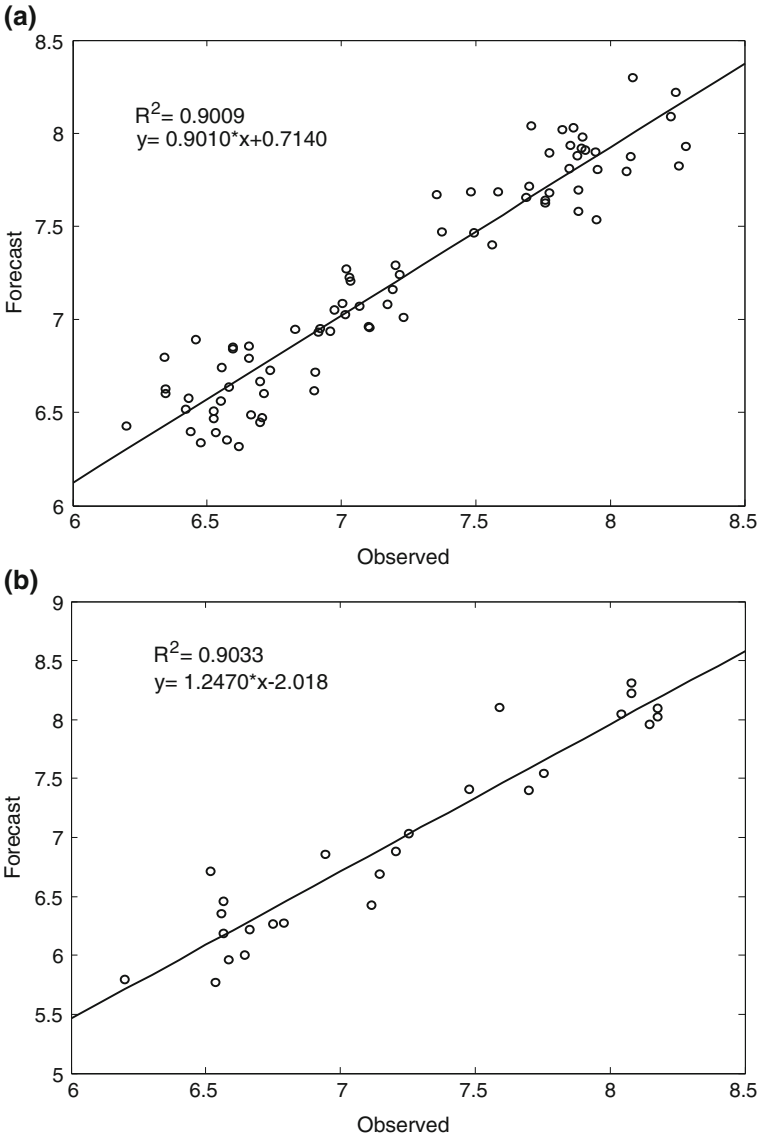

Fig. 10 ELM results for the training and validation (**a**) and testing (**b**) data set of the forecast model 2 that based on the previous 1st, 2nd, 3rd, and 12th months data

4 Conclusions

The ELM-based prediction and forecast models for phytoplankton abundance in MSR are proposed in this study. 15 water parameters with the correlation coefficients against phytoplankton abundance greater than 0.3 were selected, with 8 years (2001–2008) data for training and cross validation, and the most recent 3 years (2009–2011) for testing. The results showed that the forecast model have better performance with the R^2 of up to 0.9 than prediction model with the R^2 of 0.83, implying that the algal bloom problem is a complicated non-linear dynamic system that is affected not only by the water variables in current month, but also by those in a couple of previous months. In addition, including the previous 12th month data in the forecast model, ELM in the study showed superior forecast power and root mean square errors, indicting that the historical water parameters and phytoplankton abundance have impact on the phytoplankton dynamics of the reservoir. These results will provide an effective way for water quality monitoring and management of drinking water storage reservoirs. In addition, additional numerical approaches and optimization algorithms can be applied to enhance the performance [20–22].

Acknowledgments We thank Macao Water Co. Ltd. for providing historical data of water quality parameters and phytoplankton abundances. The financial support from the Fundo para o Desenvolvimento das Ciências e da Tecnologia (FDCT) (grant # FDCT/016/2011/A) and Research Committee at University of Macau are gratefully acknowledged.

References

1. Z. Selman, S. Greenhalgh, R. Diaz, *Eutrophication and Hypoxia in Coastal Areas: a Global Assessment of the State of Knowledge* (World Resources Institute, Washington, 2008)
2. J. Pallant, I. Chorus, J Bartram, *Toxic Cyanobacteria in Water, SPSS Survival Manual* (SPSS inc., Chicago2007)
3. G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks. IEEE Int. Conf. Neural Networks Conf. Proc. **2**, 985–990 (2004)
4. G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications. Neurocomputing **70**, 489–501 (2006)
5. G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans. Neural Networks **17**(4), 879–892 (2006)
6. R. Hecht-Nielsen, Kolmogorov's mapping neural network existence theorem, in *Proceedings of 1st IEEE Internetional Jopint Conference of Neural Networks* New York, 1987
7. K.I. Wong, P.K. Wong, C.S. Cheung, C.M. Vong, Modeling and optimization of biodiesel engine performance using advanced machine learning methods. Energy, doi:10.1016/j.energy.2013.03.057
8. A.H. Nizar, Z.Y. Dong, Y. Wang, Power utility nontechnical loss analysis with extreme learning machine method. IEEE Trans. Power Syst. **23**(3), 946–955 (2008)
9. Y. Xu, Z.Y. Dong, K. Meng, R. Zhang, K.P. Wong, Real-time transient stability assessment model using extreme learning machine. IET Gener. Trans. Distrib. **5**(3), 314–322 (2011)

10. Z.L. Su, K.M. Ng, J. Soszyńska-Budny, M.S. Habibullah, Application of the LP-ELM model on transportation system lifetime optimization. IEEE Trans. Int. Transp. Syst. **12**(4), 1484–1494 (2011)
11. L.L. Rogers, F.U. Dowla, Waster Resour. Res. **30**, 457 (1994)
12. APHA, *Standard Methods for the Examination of Water and Wastewater, A. W. W. A. a. W. E. F.* (American Public Health Association, Washington, 2002)
13. G.-B. Huang, L. Chen, Convex incremental extreme learning machine. Neurocomputing **70**(16–18), 3056–3062 (2007)
14. G.-B. Huang, L. Chen, Enhanced random search based incremental extreme learning machine. Neurocomputing **71**(16–18), 3460–3468 (2008)
15. G.-B. Huang, D.H. Wang, Y. Lan, Extreme learning machines: a survey. Int. J. Mach. Learn. Cybern. **2**(2), 107–122 (2011)
16. R. Penrose, A generalized inverse for matrices. Proc. Cambridge Philos. Soc. **51**(3), 406–413 (1955)
17. C.R. Rao, S.K. Mitra, *Generalized Inverse of Matrices and its Applications* (Wiley, New York, 1971)
18. G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification. IEEE Trans. Syst. Man Cybern. Part B: Cybern. **42**(2), 513–529 (2012)
19. Z. Xie, I. Lou, W.K. Ung, K.M. Mok, Freshwater Algal Bloom Prediction by Support Vector Machine in Macau Storage Reservoirs. Math. Probl. Eng. **2012**, 397473, 12 (2012)
20. C. Cattani, S. Chen, G. Aldashev, Information and modeling in complexity. Math. Problem. Eng. **2012**, AID 868413 (2012)
21. S. Chen, Y. Zheng, C. Cattani, W. Wang, Modeling of biological intelligence for SCM system optimization. Comput. Math. Methods Med.**2012**, AID 769702 (2012)
22. P. Lu, S. Chen, Y. Zheng, Artificial intelligence in civil engineering. Math. Probl. Eng. **2013**, AID 145974 (2013)

ELM-Based Adaptive Live Migration Approach of Virtual Machines

Baiyou Qiao, Yang Chen, Hong Wang, Donghai Chen, Yanning Hua, Han Dong and Guoren Wang

Abstract Due to having many advantages, virtualization technology has been widely used and become a key technique of cloud computing. Live migration of virtual machines is the core and key technique of virtualization fields, but the existing pre-copy live migration approach has the problems of low copy efficiency and long total migration time, so we propose an extreme learning machine (ELM) based adaptive live migration approach of virtual machines (ELMBALMA) in this chapter. Firstly, the approach uses the ELM algorithm to classify the virtual machines according to the type of the running applications, and then choose the best suitable migration algorithms for each type of virtual machines, thereby reduce the time of live migrating of virtual machines. In addition, we proposed a memory compression based live migration algorithm (MCBLMA) for the memory-intensive application scene. The algorithm uses a weight-based measurement method of writable working set, which can accurately measure the writable working set, so that it can reduce the amount of dirty memory page transmission, meanwhile it uses a memory compression algorithm to compress memory pages to be transmitted, and thus reduces the data transmission time. Preliminary experiments show that the proposed approach can significantly reduce the memory pages transmitted, the total migration time and the downtime of virtual machines.

Keywords Virtual machine · Live migration · Memory compression · ELM

B. Qiao (✉) · Y. Hua · H. Dong · G. Wang
National Ocean Information Center, Tianjin, China
e-mail: qiaobaiyou@ise.neu.edu.cn

B. Qiao · Y. Chen · H. Wang · D. Chen · G. Wang
College of Information Science and Engineering, Northeastern University, Shenyang, China

F. Sun et al. (eds.), *Extreme Learning Machines 2013: Algorithms and Applications*, Adaptation, Learning, and Optimization 16, DOI: 10.1007/978-3-319-04741-6_9,

1 Introduction

Virtualization technology plays an important role in the development of the computer technology; it is widely used in the fields of services and resources integration, system security and distributed computing. It can enhance the resilience and flexibility of system architecture, implement resources partition and aggregation, package services transparently and so on; it can greatly enhance the resource utility, reduce the computing cost, and has become one of the important supporting technologies of current mainstream cloud computing systems. As one of the key techniques of virtualization, the live migration of virtual machines can seamlessly and completely migrate a running virtual machine from one physical machine to another physical machine. It is usually used along with the resource monitoring and load balancing algorithm to schedule resources automatically in the cloud computing systems, achieving load balancing among physical machines. This greatly increases the automatic management capabilities of resource of cloud computing center, reducing operating and maintenance costs, enhancing the system availability and scalability, so live migration of virtual machines has attracted more and more people's attention.

Xen [1] is a very excellent open source virtualization hypervisor of x86 platform, which has high performance approximate to the original system, it is able to support full virtualization and has been widely used. However, virtualization platforms including Xen mainly use pre-copy migration approach to implement live migration of virtual machines. But the original pre-copy approach cannot be fit for all application scenes, especially for the memory-intensive applications. The main drawback of this algorithm is that its iterative pre-copy process needs to transfer many memory pages repeatedly when the migration memory pages are modified frequently, this leads to the repeated transmission of a large number of memory pages, not only adds the total migration time, but also takes up network bandwidth and reduces system performance. To solve this problem, we present an extremely learning machine (ELM) based adaptive live migration approach (ELMBALMA) on the basis of deeply studying the original pre-copy memory migration approach. This approach fully considers the application scenarios of different virtual machines, and firstly classifies the virtual machine (VM) into two types which are memory-intensive VMs and non-memory-intensive VMs, and then choose the best suitable migration algorithms for each types of VMs, thereby reduce the overall time of live migrating of VMs, meanwhile, we proposed a memory compression based live migration algorithm (MCBLMA) for the memory-intensive application scene. The algorithm uses a weight-based measurement method of writable working set, which can accurately measure the writable working set, so that it can reduce the amount of memory pages transmission, it also uses a memory page compression algorithm to compress memory pages which will be transmitted. The source physical host uses the compression algorithm to compress memory pages before they are sent, and the destination physical host will recover the memory pages after receiving the compressed data and decoding the data with the

corresponding decoding algorithm, thereby further reducing the data transfer time. Experiments show that the proposed approach can effectively reduce the amount of memory pages transferred, the total migration time and downtime of VMs.

2 Related Works

In recent years, virtual machine live migration and Extreme Learning Machine (ELM) techniques are given much attention by scholars at home and abroad, and a lot of relative works are done. ELM is a new learning algorithm for single hidden layer feed-forward networks (SLFNs), it is proposed by Huang et al. [2, 3]. ELM not only can avoid a number of iterations and the local minimum, but also have better generalization, robustness and controllability, so it is widely used in regression and classification problems. Cao and et al. proposed an effective ELM (EELM) [4], which is used for the image classification. Engin Avci [5] proposed a combination of an adaptive feature extraction and classification using optimum wavelet entropy parameter values. The features used in this study are extracted from radar target echo signals. Herein, a genetic wavelet extreme learning machine classifier model (GAWELM) is developed for expert target recognition. [6] proposed model-namely, the multiple extreme learning machines (MELMs)—shows promising performance under numerous assessing criteria and constructs a pre-warning model to assist decision makers in making an appropriate decision in a turbulent economic climate. Zheng et al. [7] proposed a novel approach for text categorization based on a regularization extreme learning machine (RELM) in which its weights can be obtained analytically, and a bias-variance trade-off could be achieved by adding a regularization term into the linear system of single-hidden layer feed forward neural networks. In the field of virtual machine, live migration algorithm is one of the main research directions, and many research results have been made. many typical virtual machine migration algorithms are presented such as stop-and-copy [8], pre-copy [9], post-copy [10] and etc., the stop-and-copy migration algorithm is suitable for small memory system, pre-copy algorithm is the mainstream live migration algorithm of virtual machine currently, it copies memory pages on the source machine to the target machine in iterative manner, the algorithm has the problem which the total migration time is too long. Hines proposed post-copy algorithm, which first transfer state of the CPU and necessary memory information to the target host, at the same time start the virtual machine, then transfer necessary memory pages from the source host to the target host dynamically. [11] proposed a new fast and transparent virtual machine migration mechanism which use Re-Virt framework and combined the checkpoint recovery idea with the track playback technology, but the method waste some performance. [12] proposed an adaptive data compression algorithm, which select the appropriate compression algorithm to compress memory pages based on the characteristics of the memory page data. [13] proposed an improved pre-copy scheme, It adds a data structure called to_send_last bitmap to mark memory pages that modified frequently, In the final round of the iterative process of copying, transfer memory pages marked

by the to_send_last bitmap. [14] proposed a hierarchical copy algorithm based on Xen pre-copy algorithm, which layered memory pages according to its modification frequency, and adjusted the transmitting strategy of memory pages in the first iteration, the first iteration does not send all memory pages, and set writable working set testing in advance of the stage of pre-migration and resource reservation of original algorithm. [15] proposed a Slowdown Scheduling Algorithm, which decrease the CPU resources which have been assigned to migration domain, and reduces the dirtying page rate according to the decrease of CPU activity. But it can add response time. [16] proposed a memory migration mechanism named Microwiper. Microwiper includes two policies which are transferred according to memory page modification rate and transmission regulator. The former refers to the virtual machine memory is divided into a series of regions, migrates according to the rate of modification of the region, the latter refers to adjust the page transmission according to the network bandwidth.

3 Analysis of Xen Pre-copy Migration Algorithm

The virtual machine (VM) migration is mainly related to two important performance indicators which are downtime and total migration time, according to the weight of two indicators, different algorithms are designed to meet different application requirements. Pre-copy method well balances the contradiction between downtime and total migration time, and it is used by the live migration of the VMware, Xen and other mainstream virtualization platform. Below are the main steps of the pre-copy algorithm. Assume that the source host is A and the destination host is B, and then the whole Xen pre-copy algorithm consists of the following five steps:

Step 1: Source Reservation. At this stage, the source host A send request to the destination host B, the first thing need to be done is to make sure whether there are enough resource to run the migration virtual machine. If so, reserve the resources which has the same size as the virtual machine migration, or the VM is still running in A, and it may choose other host as the destination host.

Step 2: Iterative Pre-Copy. At this stage, the memory of the VM will be copied to the destination host B by A in an iterative manner; the VM in A is still in the state of running and offer service. In first round of iteration copy, all the VM memory pages should be copied from A to B. In the latter iteration, only copy the memory pages modified on the last transmission process.

Step 3: Stop-and-Copy. The process of the iteration will be stopped when meeting the given conditions. Next is to suspend the migration VM of the source host A, redirect the network connection of VM to destination host B, and begin to transmit the state of the CPU and the left dirty pages. After the end of this stage, A and B have the same copy of the suspended VM. If the migration fails, the VM in A could still restore to run.

Step 4: Commitment. The target host B informs Host A that virtual machine image has been successfully received. Once the message is conformed, the source host A will destroy the original VM.

Step 5: Activation. In this phase, host B activate the migrated VM, bind the device drivers to the destination host B, and broadcast the new IP address.

In the above migration algorithm of Xen, memory iterative pre-copy is the important factor affecting migration performance. The VM memory migration of Xen is copy memory pages from the source host to the destination host iteratively. The first round of transmission includes all pages of memory of the VM, and for later iterations, the n-th iteration of the transmission only includes the modified pages in the n-1th iteration process. But from a systems point of view, the memory pages frequently modified are to be repeatedly transferred for many times. This increases the number of memory pages for each iteration transmission and total migration time, it has the negative impact on the migration performance. These pages changed frequently are called Writable Working Set (WWS), so precise determination of WWS can avoid repeat transmission of the pages and reduce the iteration number. In order to accurately determining WWS and migrate memory efficiently, Xen divide virtual machine memory pages into three categories, using three page bitmap variables, to_send, to_skip and to_fix to denote. To_send denotes the pages that become dirty in the former round of the iterative process and those pages are need to be transmitted in this iteration; To_skip is the bitmap that Xen introduced in order to reduce the memory pages retransmission, it marks the pages that are modified more frequent and can be skipped in the current round of iteration. To_fix marks the pages that have not been mapped and will be transferred in the final round of stop and copy stage.

We can see from above analysis, Xen pre-copy algorithm mainly through two consecutive collection of dirty page information to two bitmaps to_send and to_skip, then compare this two bitmaps information to determine the WWS. Obviously, such a method of testing WWS is too simple, especially under the memory pages being modified frequently, the information collected is not sufficient to predict modification state of a page, and this leads the WWS measurement inaccurate, and makes many memory pages are transmitted repeatedly more times, meanwhile, the bigger the memory size of virtual machine, the greater the amount of data to be transferred, and those increase the total migration time, which makes the pre-copy migration algorithm not suitable for memory-intensive application scene.

4 ELM Based Adaptive Live Migration Approach

We can see from the above analysis, the pre-copy algorithm does not efficiently process VM live migration in memory-intensive application scenarios, so we combine ELM techniques and propose an ELM based adaptive live migration approach of virtual machines (ELMBALMA) to solve the problem. Next we first introduce the ELM, then describe the ELM based VM live migration framework and related algorithms in detail.

4.1 ELM

Extreme learning machine (ELM) is a new learning algorithm for single-hidden layer feedforward neural networks (SLFNs), which is proposed by Huang et al. [2, 3]. It not only can avoid a number of iterations and the local minimum, but also have better generalization, robustness and controllability, and is widely used in regression and classification problems. It randomly assigns the input weights and hidden layer biases and then analytically determines the output weights of SLFNs. ELM can achieve better performance than other conventional learning algorithms for classification [17]. Also, it is less sensitive to user-specified parameters, and can be developed faster and more conveniently [18].

Given $\mathrm{X_i} = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in R^n$ and $\mathrm{t_i} = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in R^m$, and an activation function g(x), standard SLFNs with N arbitrary $(\mathrm{x_i}, \mathrm{t_i})$ samples are modeled as

$$\Sigma_{i=1}^{L} \beta_i g_i(x_j) = \Sigma_{i=1}^{L} \beta_i g(w_i \cdot x_j + b_j) = o_i, \quad (j = 1, \ldots, N) \tag{1}$$

where L is the number of hidden layer nodes, $\mathrm{w_i} = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ is the weight vector between the ith hidden node and the input nodes, $\beta_\mathrm{i} = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ is the weight vector between the ith hidden node and the output nodes, and b_i is the threshold of the ith hidden node.

The output of ELM has been modeled as follows.

$$f(x) = \Sigma_{i=1}^{L} \beta_i G(a_i, bi, x) \tag{2}$$

where

$$\begin{aligned} &H(w_1, \ldots, w_L, b_1, \ldots, b_L, x_1, \ldots, x_L) \\ &= \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_L \cdot x_1 + b_L) \\ \vdots & \ldots & \vdots \\ g(w_1 \cdot x_N + b_N) & \cdots & g(w_L \cdot x_N + b_L) \end{bmatrix}_{N \times L} \end{aligned} \tag{3}$$

Because standard SLFNs with activation function $g(x)$ can approximate these L samples with zero error, it means $\sum_{j=1}^{L} \|o_j - t_j\| = 0$ and there exist β_i, w_i and b_i which satisfy the following equation:

$$\sum_{i=1}^{L} \beta_i g(w_i \cdot x_j + b_i) = t_j \qquad j = 1, \ldots, N \tag{4}$$

The equation above can be expressed compactly as follows:

$$H\beta = T \tag{5}$$

H is called the hidden layer output matrix of the neural networks.

Given a training set $\aleph = \{(x_i, t_i) | x_i \in R^n, t_i \in R^m, i = 1, \ldots, N\}$, activation function $g(x)$ and hidden node number L, algorithm ELM is describe as algorithm 1.

Algorithm 1. ELM.

1. for (i=1 to L)
2. randomly assign input weight w_i
3. randomly assign bias b_i
4. endfor
5. calculate H
6. calculate β=H$^+$T

4.2 ELM Based Virtual Machine Live Migration Framework

Because different types of virtual machines whose load type and application scenarios are different, so each virtual machine live migration algorithm can efficiently support only certain types of virtual machine live migrations. Such as pre-copy live migration algorithm is suitable for non-memory-intensive application scenarios, and post-copy live migration algorithm is suitable for the time-sensitive application scenarios. Therefore, classify virtual machines into different types according to the load type and application characteristics of the virtual machines, then select the most appropriate algorithms to achieve the live migration of different type virtual machine has become an important way to improve the overall performance of virtual machine migration within a cluster.

Therefore, in this chapter we present a ELM based virtual machine live migration framework, which takes advantage of current popular ELM technique to achieve accurate classification of virtual machines in a cluster, and chooses the most suitable migration algorithms to perform live migration of virtual machines in some application scenarios, thereby shortens the overall virtual machine migration time and improves the efficiency of VM live migration.

Figure 1 shows the ELM-based virtual machine live migration framework, which mainly consists of ELM classifier and two migrate algorithm modules. Firstly, we use the training data to train the ELM classifier, and then classified virtual machines into memory-intensive and non-memory-intensive types according to the feature data of the application on the virtual machines, memory-intensive virtual machines are migrated using the memory compression based live migration algorithm (MCBLMA) which is proposed specifically for the migration of memory-intensive virtual machines and presented in subsequent sections in detail, no-memory-intensive virtual machines are migrated using the original pre-copy live migration algorithm. Obviously, the framework can use the most appropriate migration algorithms to achieve the migration of two type virtual machines, which can enhance the overall efficiency of migration of virtual machines within a cluster. In fact, the migration framework is a combination of the ELM technology with two kinds of migration algorithms, so it is an adaptive method of virtual machine live migration.

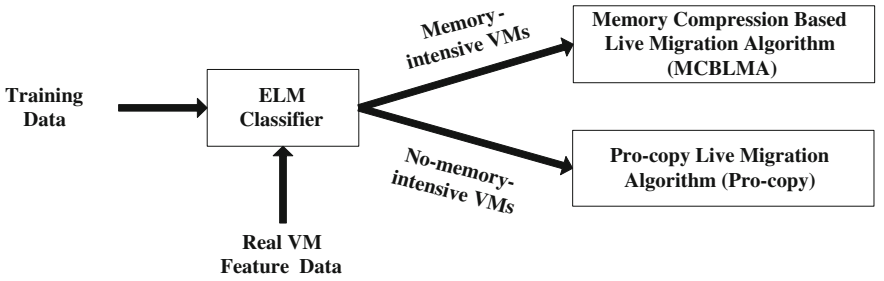

Fig. 1 The framework of ELM based live migration of virtual machine

4.3 Classification of Virtual Machines

ELM algorithm has good generalization performance, so we choose ELM classifier to classify virtual machines. The following is the related contents about classification of VMs, such as the selection of VM feature data, data preprocessing and the selection of ELM model parameters.

(1) The feature data selection of virtual machine Using ELM classifier to classify virtual machines, the first issue is to select the properties of virtual machines. Although virtual machines have a lot of properties, we only select three feature data which is related to the live migration of virtual machines as feature data, the three features are as follows:

   (a) Memory size of virtual machine (V). Memory size is the key factor to affect VM live migration. Generally, the bigger of the VM memory size, the greater amount of data should be migrated and the longer of migration time.
   (b) Rate of dirty page (D). Rate of dirty page reflects the speed of memory dirty pages generated, it represents the memory access feature of application programs, it is a core factor to affect the performance of VM migration. It directly affect the migration copy iteration round number and the amount of data transferred per round, which will ultimately affect the total migration time and the total amount of data transmission;
   (c) Network transmission speed (R). The network transmission speed is also a core parameter to affect live migration performance; it directly affects the migration algorithm convergence speed. The greater of network bandwidth, the smaller of data transfer time;
   Because the extraction of above three feature parameters of virtual machines is relative simple, it can achieve a more accurate classification of virtual machines. in a certain extent.

(2) The data preprocessing
   In order to achieve better efficiency and accuracy of ELM classification, we make use of Fuzzy Clustering Method to preprocess the original data, and then

input them into ELM classifier, so that we can train and test the ELM classifier. The details of Fuzzy Clustering Method are as following:

(a) Establish fuzzy similar matrix S
The formula of fuzzy similar matrix S is as Eq. (6):

$$S_{ij} = \begin{cases} 1, & i = j; \\ 1 - C \sum_{k=1}^{m} |h_{ik} - h_{jk}|, & i \neq j. \end{cases} \tag{6}$$

where, S_{ik} is property value for the ith row kth column; S_{jk} is property value for the jth row kth column; C is a constant, so that it makes $0 \leq s_{ij} \leq 1; i, j = 1, 2, \ldots, n$, where m is the number of the sample property, n is the number of the sample.

(b) Calculate equivalence matrix for fuzzy matrix $t(S)$
Calculate the transitive closure for S through convolution, that is S multiply itself, e.g. $S^k = S^{2k}$, then multiply again, until $S^k = S^{2k}$. So the fuzzy equivalence matrix $t(S) = S^k = S^{2k}, k \in N$. The fuzzy equivalence matrix data work as the input of ELM sample data.

(3) The selection of ELM Model parameters
The main parameter for ELM classification model is the number of hidden layer neuron, it relates to the number of input layer neutron. So we initialize the number of hidden layer neutron by 20, then according to the training error to increase properly, so as to achieve the proper number of hidden layer node number. If the error is greater than the allowed bound, we increase the number of hidden layer nodes, otherwise, we decrease it. We choose Sigmoid as the activation function, the equation is as following:

$$g(x) = 1/(1 + e^{-x}) \tag{7}$$

5 Memory Compression Based Live Migration Algorithm

According to the above analysis of the pre-copy algorithm, the pre-copy migration algorithm is not suitable for memory-intensive application scene; the main reason is that its writable work set (WWS) measurement is inaccurate, and this leads to many memory pages are transmitted repeatedly, increases the migration time. To solve the problem, we propose a memory compression based live migration algorithm of VMs (MCBLMA). Below we will describe the proposed algorithm from the following aspect: the main idea, weight-based WWS measurement approach and the memory pages compression algorithm.

5.1 Main Idea

According to principle of program locality, the more recently the instructions and data are used, the more likely they are used later. So the memory pages modified recently has the higher probability to be modified again. Based on this principle, we propose a weight-based WWS measure approach which can determine WWS accurately; meanwhile we develop a memory compression algorithm which can compress the memory pages to transmit. The main ideas are described as below:

(1) In each round of the iterative process, we repeatedly collect dirty page information of VM and assign an appropriate weight, and then compute the weight of each dirty page. The larger weight value of the dirty page shows its higher modification frequency and will be likely to be used again in the next period of time, and the smaller weight value of a dirty page shows the lower modification frequency and in the future over a period of time is not likely to be used. And then we set a threshold, and only those dirty pages whose weight is less than the threshold will be transferred in each iteration copy process. In this way, we can predict the memory page modification trends of VMS, and accurately measures the WWS, thereby reducing the amount of pages to be transmitted during migration time.
(2) Designing a memory compression algorithm to compress the memory dirty pages, and hence reduce the amount of data to be transmitted and improve the data transmission efficiency in each iteration process, shorten the migration time and optimize the migration performance.

5.2 Weight-Based WWS Determining Approach

The proposed approach, which determines the WWS by the way of adding weight, is mainly through collecting the dirty pages in memory and then calculates the weight of each dirty page to determine the WWS. The collection of the dirty pages is support by the page operation XEN_DOMCTL_OP_CLEAN provided by the Xen platform. It directly copy the bitmap of the dirty page in memory to the array dirty_maps and then set the bitmap null. After the next given time_slot, we need to collect the dirty pages again and then repeat this operation until we reach the scheduled times. The time_slot can be given and is initialized as 900ms in the platform Xen3.4.2 and will reduce 80ms in each iteration process. The Fig. 2 is an example of the collection of the dirty pages.

We can see from the Fig. 2, the dirty page information of memory are collected many times, all of them are stored in the dirty_maps array, then using the information to calculate the weight of each dirty page. If we use Dirty_Weight_i to express the weight of dirty page i, the calculation method is bellowed as Eq. (8):

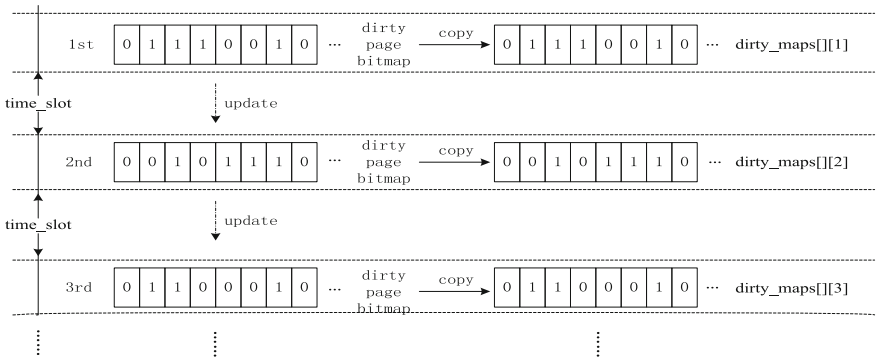

Fig. 2 The demonstration of collecting dirty page information

$$Dirty\_Weight_i = Dirty\_maps_{[i][1]} * Weight_1 + Dirty\_maps_{[i][2]} * Weight_2 + \ldots + Dirty\_maps_{[i][ct]} * Weight_{ct} \quad (8)$$

In the formula above, dirty_maps[i][j] means the modifying information of memory page i collected in the j-th time, its value is 0 or 1, 1 means the page is changed, and 0 means the page doesn't be changed; Weight_j that can be specified means the weight of dirty page information collected in the j-th time; Actually, the weight of dirty pages is the product of each modifying information of them multiplies each page weight of each collecting time, and then sum them.

We can know from the Eq. (8), setting the weight of dirty pages in each collection time not only influence the weight of the whole dirty pages mostly, but also influence the performance of memory compression based live migration algorithm directly. We can know from the discussion above, the weight of pages that collected latest should be bigger than those earlier, which may measure the WWS accurately. We can set weights with different systems, such as binary system, $2^0, 2^1, \ldots 2^i \ldots$; quaternary system, $4^0, 4^1, 4^2, \ldots, 4^i$, etc. In order to make the weight of pages smaller for the convenient calculation, we take the binary system as the weight of pages that each time collected. We set the weights $\text{weight}_1, \text{weight}_2, \ldots, \text{weight}_{ct}$ of pages that collected in first time, second time, ..., collect_times time as $2^0, 2^1, \ldots, 2^{ct-1}$. This way not only satisfies the need of the dirty page weight of each collection time is different, and the weight of latest collection is higher in one magnitude than that of last time. The Fig. 3 is an example of calculating page weights, where CT is 4, the weights of 4 collecting times and the weight calculated of ten dirty pages are shown in the figure.

When calculating the page weight, it needs to compare it with the predefined threshold to decide whether to send it this round of iterative copy. It's important to set the value of the threshold, if too big it can cause some frequently changed pages are looked as not frequently changed pages, and to be transferred repeatedly, this increases the network loads and total migration time; if too small it can lead

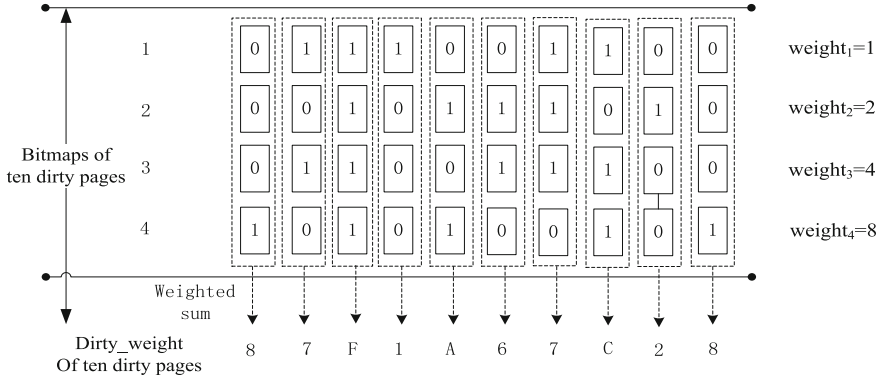

Fig. 3 An example of calculating page weights

some not frequently changed pages are mistaken as frequently changed pages to be sent at the last, so as to increase the transferred page numbers of the final round of iteration, and to prolong the downtime of that machine. So the threshold must have a moderate value. There will be a threshold setting the highest number of weight of the collection times, which is $2^{(ct-1)}$. So we can mark the memory pages into three categories according to the threshold: (1) the pages that their dirty_weight value are equal to zero have never been modified and will be sent to the destination host in the first round of iteration; (2) the pages that the weights are greater than zero but less than the threshold can be thought as not frequently modified pages and will be sent at this round of iteration; (3) the pages that their weights are greater than the threshold are looked as frequently updated pages and will be updated recently, and will be skipped to transfer in this round of iteration. This can improve the efficiency of page transferring, reduce the times of iteration, and shorten the total migration time.

5.3 Compress Algorithm of the Memory Pages

We can see from the pre-copy algorithm that reducing the amount of dirty pages in each iteration is an important method to reduce the migrating time. In addition to the determination of the WWS, the most intuitive way is to use the way of memory page compression transmission. In this way, we can reduce the amount of the transmission data, transmission time and speed up the convergence rate. But the compression process itself will bring some time cost, which may offset the benefits of compression itself. Here we make use of a mathematical model to describe the change of migration time and the amount of dirty pages transmitted when using MCBLMA algorithm.

Assuming that MCBLMA algorithm requires N times iteration copies and the amount of data transmission in each iteration is $V_i (0 < i < N)$, the data transmission

time of each iteration is T_i, the network transmission speed is R, the average dirty page rate is D($0 <= D <= 1$), compression rate per memory page is Z($0 <= Z <= 1$), compression time per memory page is t_c. According to the MCBLMA algorithm, the first iteration processing time T_0 and the amount of data transmission V_0 can be computed by the following equation respectively:

$$V_0 = V_m \times Z, T_0 = V_m \times Z \div R + V_m \times t_c \tag{9}$$

where V_m is the initial memory size of virtual machine, V_i and T_i of the rest iteration can be computed by the following equation respectively:

$$V_i = T_{i-1} \times D \times Z, \ 1 \leq i \leq N \tag{10}$$

$$\begin{aligned} T_i &= T_{i-1} \times D \times Z \div R + D \times T_{i-1} \times t_c \\ &= (D \times Z \div R + D \times t_c) \times T_{i-1} \\ &= (D \times Z \div R + D \times t_c)^i \times T_0 \end{aligned} \tag{11}$$

As the network transmission speed R, the virtual machine memory size V_m is fixed, and the dirty pages rate D is determined by the application characteristics and can be considered as fixed, in this case, we can see from Eq. (11), the processing time of each iteration T_i is decide only by compression rate per memory page Z and compression time per memory page t_c. so we should consider to shorten the migration time of virtual machine from the two aspect. However, as mentioned earlier, under normal circumstances compression speed and compression ratio is contradictory, so when choose a compression algorithm we should make a reasonable compromise between compression speed and compression rate, and make the migration time shortest. On the basis of studying the related character encoding compression algorithms, we consider the requirement of virtual machine live migration comprehensively and design a memory live compression algorithm (M2LZO) which is based on the character code model. The basic idea is to build a dictionary, and for the input string, use hash approach to find the match strings, for a matched string, get its location and the same string length, then output the generated encode according to the compression encoding format. The algorithm is an improvement of LZO compression algorithms and the main improvement is search matching method. LZO algorithm uses twice hash lookup matching method which can not support the best match search, thus affecting the compression rate. So we use a two level hash matching method. Because it increase the second level hash matching which is more than the original LZO algorithm, it can match a longer string, and has a better compression ratio, at the same time it doesn't reduce the compression speed of the original algorithm.

M2LZO algorithm is made up of three parts: duplication degree checking algorithm of memory pages, M2LZO encoding algorithm and M2LZO decoding algorithm. Firstly, it checks the duplication degree of the memory pages generated by the MCBLMA algorithm in the source host. Secondly, it uses different compression levels of M2LZO compression algorithm to compress the memory pages according

to their duplication degrees, and then send them to the destination host. In the destination host, the M2LZO decoding algorithm is used to decode the compressed memory pages, and then apply MCBLMA algorithm to recover the memory pages. So the MCBLMA algorithm is actually composed of weighted migration pre-copy algorithm, memory page repeatability checking algorithms and memory live compression algorithm M2LZO. As M2LZO algorithm is similar to LZO algorithm, the detail description is omitted.

5.4 Process Flow of the Proposed Algorithm

The process flow of the proposed algorithm MCBLMA is generally the same with that of original xen pre-copy algorithm migration, while the difference lies in the memory migration. There are two differences; the one is that the measuring of WWS is not simply comparison of the bitmap to_send and into_skip to decide whether to transfer a memory page, but introducing the weight calculation of dirty pages to determine the memory pages transferring. The other is that the memory compression algorithm M2LZO is introduced to compress memory pages to reduce amount of data transmission. The detail processing steps of MCBLMA are as following:

(1) Collect collect_times (CT) times messages of memory dirty pages into dirty_maps array in the process of preliminary migration.
(2) When the iteration pre-copy begins, calculates the weight Dirty_weight of the memory pages in dirty_maps according to weight computing formula and mark those pages that meet the conditions of transferring into mem_page_send array.
(3) Collecting M requirements pages which satisfy condition into buffer, as the objects of this round iteration. The pages satisfying the requirements refer to: (a) the page is not in the final round of iteration, and its mem_page_send label not 1; Or (b) The page of the final round of iteration, and its bitmap to_fix corresponding label to be 1; Or (c) The page of the final round of iteration, and its bitmap to_send corresponding label to be 1, that is, the last dirty page of virtual machine. Here the bitmap to_fix is the same with the original pre-copy algorithm, but the bitmap to_send here marks the final dirty page of virtual machine, which is already not the same with the original algorithm.
(4) Compress and transmit the collected memory pages which satisfy the requirement. Firstly, repeatedly check the memory pages, and then use different compression level of M2LZO algorithm to compress memory pages according to their duplication degree and place the compressed pages into send buffer.
(5) Transfer the memory pages in the buffer, repeat the operations in steps (3), (4) and (5) until all the memory pages of virtual machine are scanned.
(6) When transferring memory pages of a round, determine whether to be the final round of iteration, if not repeat the operation of collecting dirty page messages in step (1), if it is the last round of iteration, copy the dirty messages into to_send directly.

(7) Repeat step (2), (3), (4), (5), (6) until meeting the exit conditions, then transfer the last memory pages. The exit conditions are the same with original pre-copy algorithm.
(8) Suspend VM, and directly copies dirty page information into to_send bitmap, and then transfer the related memory dirty pages which are corresponding with to_send bitmap.

The detail description of MCBLMA is described as algorithm 2:

Algorithm 2. memory compress based live migration algorithm (MCBLMA)
Algorithm description：

```
1.    set the to_send bitmap to 1;
2.    for(; ;) {
3.      iter++; send_this_iter = 0; skip_this_iter = 0;
4.      while(N<p2m_size) {
5.        Call xc_shadow_control() to copy the bitmap of the dirty pages to to_skip;
6.        for (batch = 0;(batch < MAX_BATCH_SIZE) && (N < p2m_size);N++ ){
            While(!last_iter && map the pages meet requirement in dirty_maps to
        mem_page_send) {
8.              Calculate the weight of the page;
9.              Decide whether to send the page; }
10.             if ( !last_iter &&test_bit(n, to_send) &&test_bit(n, to_skip) )
11.             skip_this_iter++;
12.       if (page N meets three conditions)
13.             put N to buffer region pfn_type[batch];
14.               batch++; }
15.         map the page in buffer to the corresponding memory;
16.         disconnect pfn mapping with mfn;
17.         for(batch1=0;batch1<=batch;batch1++) {
28.           read(pfn_type[batch];
20.           call the repeatedly checking algorithm of the memory page;
21.             if(memtest()==1)
22.                     call M2LZO (1);
23.             else if(memtest()==2)
24.             call M2LZO (2);
25.             else call M2LZO (3);
26.             put the compressed memory page to the buffer;} }
27.     if ( last_iter ) break;
28.     if (meet the exit condition) {
29.       last_iter = 1;
30.       suspend the virtual machine; }
31.     call xc_shadow_control to copy the bitmap of the dirty page to to_send; }
```

6 Performance Evaluation

In order to evaluate the effectiveness of the proposed approach and algorithms, we establish the corresponding experiment environment and make a large number of experiments to test performance about the proposed MCBLMA and ELMBALMA approach. In addition, we also make comparison with the traditional Pro-copy algorithm, below is the details of the experiments.

6.1 Experimental Environment

The test environment consists of six IBM PC servers (2 CPU, 32G Memory, 12T Disk), among which five servers are used as physical nodes and 1 server are used as administration node, which connect each other with Gigabit Ethernet and are installed with Centos6.4 (kernel 3.1.2) and Xen 4.1.2. The administration node also installs convirture 2.1.1 to manage the physical nodes. Meanwhile, we also install the migration modules modified and the related resource monitoring modules, all these constitute the complete experiment environment. On the above experiment platform, we create many virtual machines with 128M, 256M, 512M and 1024M memory. Every virtual machine (VM) is installed with CPU intensive, memory intensive and I/O intensive applications respectively, such as TCP-C, Linpak, Dbench, Memtester, DaCapo and so on. We also design the feature data extraction program and extract 100 feature data from 100 virtual machines, 60 feature data are used to train ELM classifier and the other 40 data are used for the testing of the accuracy of the ELM classifier.

6.2 Experiment Results

In this chapter, the ELM classifier is trained and tested firstly, and then the performance of the MCBLMA and ELMBAMA algorithms are tested and compared with the traditional Pro-copy algorithm. The detail performance analysis is as the following.

(1) ELM Classifier Performance Analysis
In order to test the accuracy of the ELM classifier, virtual machines with different memory size are created and several kinds of benchmark programs are installed to extract the data for training and testing the ELM classifier. About 100 feature data are extracted from different VMs; these data can reflects the memory access characteristics of different loads and applications scenarios. About 60 feature data are used to train and the rest 40 data are used to test the performance of ELM classifier. In the 60 training data, three types virtual machines: memory-intensive, CPU-intensive and I/O-intensive, each has 20 feature data. The initial hidden nodes of the ELM classifier is 20, but for the accuracy of the classification, the hidden nodes are set to 35 finally. In this condition, the experimental result is show in Table 1. We can see from the table that the accuracy of memory intensive is 93.3 %, and the accuracy of the CPU intensive and I/O intensive reach 84.6 and 91.7 % respectively, it shows the prediction accuracy of the ELM classifier is very high, the ELM classifier can meet the requirements of the VM classification completely, the data feature parameter selection and the ELM parameter setting are reasonable.

Table 1 The predication rate of ELM classifier

Item list	Memory intensive	CPU intensive	I/O intensive
Training data size	20	20	20
Testing data size	15	13	12
The number of accuracy prediction	14	11	11
The rate of predication (%)	93.3	84.6	91.7

(2) Performance Analysis of Migration Algorithm
As MCBLMA algorithm require to collect the dirty page information for many times to compute the weight to determine the writable working set, so the value of collection times (CT) of dirty pages need to set reasonably. Below we will firstly analyze the influence of CT on the performance of the MCBLMA algorithm from three aspects: total migration time, the amount of transmission pages and the downtime. Then we will give performance analysis and comparison of MCBLMA, ELMBAMA and Pro-copy algorithms.

(a) The influence of collection times
In order to test the effect of dirty page collecting times (CT), we create two virtual machines with 512M memory and 1G memory respectively, then we execute live migration of VMs under running three different applications: I/O intensive, CPU intensive and memory intensive applications. Figure 4 describe the total migration time varies with different CTs. We learn from the figure that as the CT increase, the total migration time decrease firstly and then increases. When CT is 4, the total migrate time become shortest, and when CT is greater than 5, the total migration time increases slightly, the reason is that collect dirty information can bring certain time costs, the more CT, the bigger time costs. Figure 5 describes the varying of the amount of transmission data with different CTs. We can see from it that the amount of transmission data decreases with CT increase, and when CT is greater than 5, the decreasing trend slows.
Table 2 shows the downtime. When CT is 4 or 5, the downtime is similar to the pre-copy algorithm. The downtime increases when the CT is greater than 5. Because the bigger the value of CT is, the more accurate the WWS is, and consequently the frequently modified memory pages are transferred at the final time, leading the increasing of VM downtime. Considering the total migration time, number of transmitting memory pages and downtime comprehensively, in MCBLMA, the value of CT should be set to 4 or 5.

(b) The average total migration time
The total migration time is an important indicator for measuring migration algorithm, the less the total migration time, the higher the performance of the algorithm. When the CT value of MCBLMA algorithm is 4, and 10 VMs of running 10 kinds of different benchmark applications respectively execute live migration, the varying of the average total migration time (ATMT) with

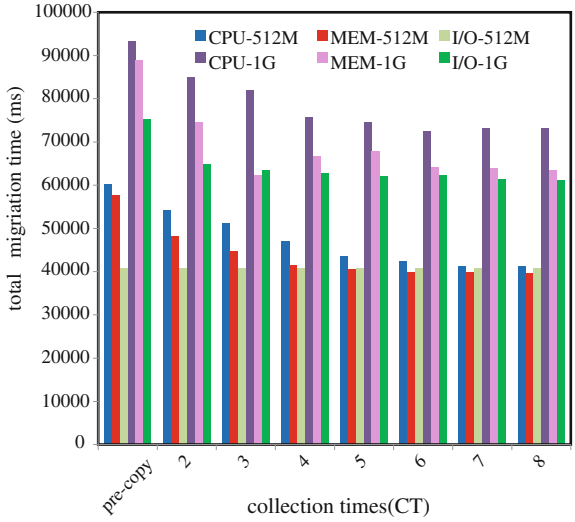

Fig. 4 The varying of total migration time with collecting times

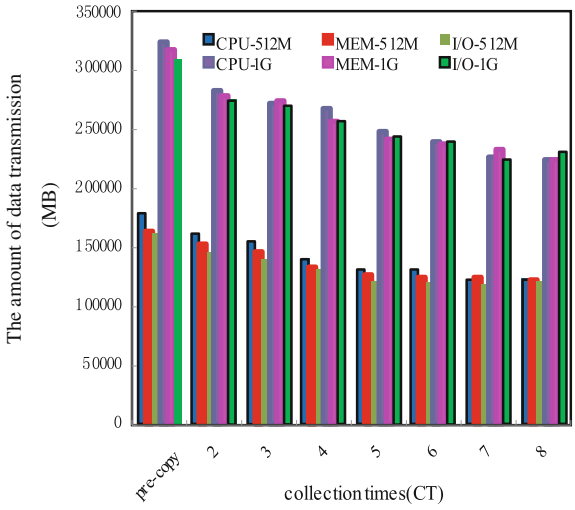

Fig. 5 The varying of the amount of data transmission with collecting times

different memory size of VMs is shown as Fig. 6. We can see from the figure, the ATMT of three algorithms increase with memory increase, and that of the pre-copy algorithm is the longest. ATMT of MCBLMA largely decreases compared with the pre-copy algorithm, this is because MCBAMA algorithm apply a memory compression algorithm and transmit the memory pages at a compression way, this can greatly reduce the amount of transmission data, thus reduce its ATMT, for the ELMBAMA algorithm can choose the best suitable algorithm to migrate the virtual server every time, so its ATMT is shortest.

Table 2 The comparison of downtime (ms) with load

Item list	Pre-copy	ct = 2	ct = 3	ct = 4	ct = 5	ct = 6	ct = 7	ct = 8
CPU-512M	732	689	715	730	745	763	801	832
MEM-512M	22926	22568	22875	22922	22987	22990	23014	23425
I/O-512M	683	650	685	690	710	725	785	810
CPU-1G	1056	956	1066	1071	1084	1093	1115	1132
MEM-1G	29164	29021	29132	29160	29301	29320	30020	30263
I/O-1G	965	930	975	986	1065	1086	1187	1208

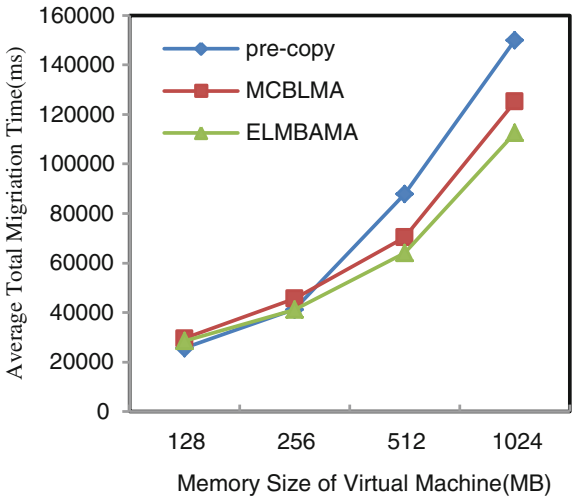

Fig. 6 The comparison of average total migration time with different memory size

(c) The total amount of memory data transmission

The total amount of memory data transmission directly influences the performance of the migration algorithm and further influences the total migration time. So it is another important indicator of the performance of the migration algorithm. Figure 7 shows the varying of the average total amount of memory data transmission (ATAMDT) with different memory size under the same condition as the Fig. 6. It shows that the ATAMDTs of three algorithms increase with the increase of the virtual machine's memory. And the ATAMDT of the pre-copy algorithm increase quickly with memory size increase, and that of MCBLMA algorithm decreases largely, this is because the dirty pages will increase with the increase of the memory. However, the MCBLMA algorithm can determine the working set more precisely and use the memory compression technique, so it can reduce its ATAMDT. The ATAMDT of the ELMBAMA algorithm is smallest; as the perspective of the ATAMDT, the ELMBAMA algorithm is best.

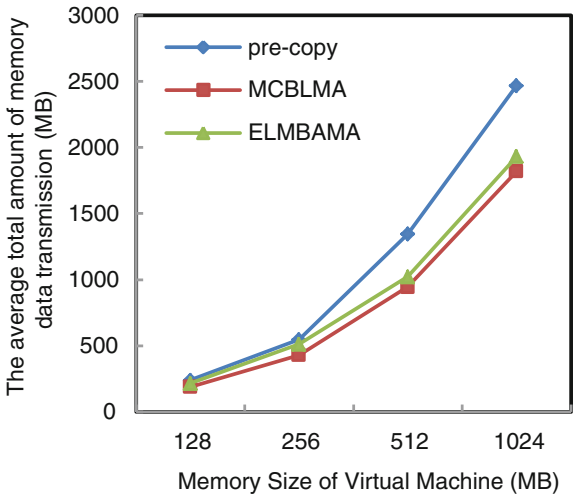

Fig. 7 The comparison of average total amout of memory data transmission with different memory size

Table 3 The comparison of downtime (ms) with memory-intensive workload

Item list	128M	256M	512M	1024M
Pre-copy-write	12897	28041	30540	32976
MCBLMA-write	11946	27546	29650	31246
ELMLBAMA-write	11786	27052	28023	30461
Pre-copy-read	322	329	350	403
MCBLMA-read	301	325	338	385
ELMLBAMA-read	293	312	315	360

(d) Average downtime

Downtime is a very important metrics in live migration of virtual machines. Long downtime will make the live migration lose its meaning, so good algorithm must be guaranteed within a certain time period to complete the live migration. Table 3 is comparison of the average downtime in three kinds of application in the process of live migration, which runs on the virtual machine with 128M, 256M, 512M, 1024M WWS respectively. It can be seen from the table, when writing memory and executing migration, the downtime increases proportionally with the WWS increases. This is because the larger the WWS, the more the memory pages are modified frequently, and those memory pages will be transferred only after the virtual machine is suspended, thus leads the longer downtime. This condition does not change for the improvement of the algorithm, because MCBLMA can only determine the WWS and it can not reduce the active memory pages. For read memory operation, the downtime is considerable less than that of the write operation. This is because the read operation does not change the contents of

the memory page, so the memory pages modified frequently are not many, and there is little pages need to send at the final stage. For the three algorithms, the average downtime of pro-copy algorithm is smaller than that of the others when WWS size is small; but with the increasing of WWS size, the average downtimes of ELMLBAMA and MCBLMA become lower than that of the pre-copy algorithm, this is because they use memory compression algorithm. So from the perspective of the downtime, the ELMBAMA algorithm is also the best one.

From the comparison of the above three aspects, we can draw a conclusion that when the collection times is 4, the proposed MCBLMA algorithm and the ELMBAMA approach are able to obtain a good balance between the total migration time and downtime, it can reduce the average total migration time, the average memory data transmission and the downtime significantly.

7 Conclusions

In this chapter, we propose the ELM based Adaptive algorithm for virtual machine live migration on the basis of the research and analysis on the existing live migration algorithms of virtual machines. The algorithm can use the most appropriate virtual machine migration algorithm according to different application types of virtual machines, thus shortening the time of the virtual machine live migration. In addition, we propose a memory compression based algorithm for the memory-intensive application. This algorithm is an improvement of the pre-copy algorithm and the WWS measurement approach which can determine the writable working set precisely, this can consequently reduce the transmitted memory pages. Meanwhile, it can reduce the amount of data which need to transmit by using the method of memory compression. The experiment results show that the proposed approach can reduce the total migration time largely, at the same time it does not increase the downtime, and has better migration efficiency. Next step we will consider classifier factors that affect the performance of the live migration to improve the migration algorithm performance on the basis of the existing algorithm.

Acknowledgments This research was supported by the National Natural Science Foundation of China (No. 61073063, 61173029, 61272182 and 61173030), the Ocean Public Welfare Scientific Research Project of State Oceanic Administration of China (No. 201105033), and National Digital Ocean Key Laboratory Open Fund Projects (No. KLDO201306).

References

1. Xen Hypervisor, http://www.xen.org/products/xenhyp.html
2. G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications. Neurocomputing **2006**(70), 489–501 (2006)
3. G.B. Huang, L. Chen, Convex incremental extreme learning machine. Neurocomputing **70**, 3056–3062 (2007)
4. F. Cao, B. Liu, D. Sun Park, Image classification based on effective extremelearning machine. Neurocomputing (IJON) **102**, 90–97 (2013)
5. E. Avci, A new method for expert target recognition system: genetic wavelet extreme learning machine (GAWELM). Expert Syst. Appl. (ESWA) **40**(10), 3984–3993 (2013)
6. S.-J. Lin, C. Chang, M.-F. Hsu, Multiple extreme learning machines for a two-class imbalance corporate life cycle prediction. Knowl.-Based Syst. (KBS) **39**, 214–223 (2013)
7. W. Zheng, Y. Qian, Text categorization based on regularization extreme learning machine. Neural Comput. Appl. (NCA) **22**(3–4), 447–456 (2013)
8. C.P. Sapuntzakis, R. Chandra, B. Pfaff, et al., Optimizing the migration of virtual computers. SIGOPS Oper. Syst. Rev. **36**(SI), 377–390 (2002)
9. C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, A. Warfield, Live migration of virtual machines, in *Proceedings of the Second Symposium on Networked Systems Design and Implementation (NSDI'05)* (2005), pp. 273–286
10. M.R. Hines, K. Gopalan, Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning, in *Proceedings of the ACM/Usenix International Conference on Virtual Execution, Environments (VEE'09)* (2009), pp. 51–60
11. H. Liu, H. Jin, X. Liao, L. Hu, C. Yu, Live migration of virtual machine based on full system trace and replay, in *Proceedings of the 18th International Symposium on High Performance, Distributed Computing (HPDC'09)* (2009), pp. 101–110
12. H. Jin, L. Deng, S. Wu, X. Shi, X. Pan, Live virtual machine migration with adaptive memory compression, in *Proceedings of the 2009 IEEE International Conference on Cluster Computing (Cluster 2009)* (2009)
13. F. Ma, F. Liu, Z. Liu, Live virtual machine migration based on improved pre-copy approach, in *IEEE International Conference on Software Engineering and Service Sciences (ICSESS)* (2010), pp. 230–233
14. Z. Liu, Q. Wenyu, T. Yan, H. Li, K. Li, Hierarchical copy algorithm for Xen live migration, in *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery* (2010), pp. 361–364
15. Z. Liu, W. Qu, W. Liu, K. Li, Xen live migration with slowdown scheduling algorithm, in *The 11th International Conference on Parallel and Distributed Computing, Applications and Technologies* (2010), pp. 215–221
16. Y. Du, H. Yu, G. Shi, J. Chen, W. Zheng, Microwiper: efficient memory propagation in live migration of virtual machines, in *39th International Conference on Parallel Processing* (2010), pp. 142–149
17. X. Wang, A. Chen, H. Feng, Upper integral network with extreme learning mechanism. Neurocomputing **74**(16), 2520–2525 (2011)
18. G.-B. Huang, X. Ding, H. Zhou, Optimization method based extreme learning machine for classification. Neurocomputing **74**(1–3), 155–163 (2010)

ELM for Retinal Vessel Classification

Iñigo Barandiaran, Odei Maiz, Ion Marqués, Jurgui Ugarte and Manuel Graña

Abstract Robust image segmentation can be achieved by pixel classification based on features extracted from the image. Retinal vessel quantification is an important component of retinal disease screening protocols. Some vessel parameters are potential biomarkers for the diagnosis of several diseases. Specifically, the arterio-venular ratio (AVR) has been proposed as a biomarker for Diabetic retinopathy and other diseases. Classification of retinal vessel pixels into arteries or veins is required for computing AVR. This chapter compares Extreme Learning Machines (ELM) with other state-of-the-art classifier building approaches for this tasks, finding that ELM approaches improve over most of them in classification accuracy and computational time load. Experiments are performed on a well known benchmark dataset of retinal images.

Keywords Retinal vessels · Arterio-venular ratio · Biomarkers · ELM

1 Introduction

Recent studies [11] point to the importance of the fundus imaging as a substantial part of a large number of diagnostic procedures for a wide variety of pathologies. This technique allows to obtain high resolution images of the internal structures of the retina, such as the micro-vascular tree or the optic disc, as shown in Fig. 1. Currently,

I. Barandiaran (✉) · O. Maiz
Vicomtech-Ik4 Foundation, País Vasco, Spain
e-mail: ibarandiaran@vicomtech.org

I. Barandiaran · I. Marqués · M. Graña (✉)
Computational Intelligence Group (UPV/EHU), País Vasco, Spain
e-mail: ccpgrrom@sc.ehu.es

J. Ugarte
ULMA Group, Oñati, Spain

F. Sun et al. (eds.), *Extreme Learning Machines 2013: Algorithms and Applications*, Adaptation, Learning, and Optimization 16, DOI: 10.1007/978-3-319-04741-6_10,

Fig. 1 Image of the retina obtained by fundus imaging

there is an increasing scientific evidence regarding the role played by micro-vascular diseases in relation to the pathologies associated with macro-vascular structures. Studies such as [11] have shown how a condition in coronary micro-vascular structure, may cause serious heart failure with risk of heart attack and death, without any pathological evidence in coronary macrovascular structures, so that periodic checks of such structures may not reveal the existence of pathology. Moreover, some dysfunctions in skin microvascularity which is estimated to be representative of the entire micro-human circulatory system, have been associated with increased risk of heart attack. However, studies over microvascularization are small relative to the affected population because they need laborious and very invasive techniques. For this reason, researchers are looking for non invasive alternatives and mechanisms allowing accurate analysis of microvascular structures. Retinal imaging allows studying different aspects of the microcirculation in-vivo, whose role in vascular or metabolic diseases is less clear than that of macrocirculation [10]. Image analysis with vascular morphometry techniques carried out over large populations point out correlations between retinal microvascular patterns and different cerebrovascular and cardiovascular diseases and metabolic disorders [16]. We focus on one retinal image biomarker with great diagnostic value, which is the arterio-venular ratio (AVR), computed as the quotient between the averages of the widths of several arterioles and venules. Alternatively, the AVR is also computed as the quotient of the central retinal artery equivalent (CRAE) and the central retinal vein equivalent (CRVE) [9].

The quantification of retinal bio-markers such the AVR, CRAE or CRVE over large populations requires automated tools for vessel segmentation and analysis. We are interested in low complexity and fast approaches that could allow the clinicians to be able to carry out large screening programs. There are two steps in this process:

(a) Image segmentation to obtain the location of the vessel pixels in the image
(b) Vessel pixel discrimination into arteries and veins, needed to compute the AVR.

Current methods for retinal vessel segmentation mechanisms [4] can be roughly categorized as those based on supervised learning [14] and unsupervised [2] techniques. Supervised learning techniques rely on hand labeled images for the off-line classifier training process. Segmentation process becomes a pixel classification into two different classes: vessel versus background. On the other hand, unsupervised approaches rely on image processing and analysis techniques specialized for vascular structures. In these approaches, hand labeled images are used only for validation,

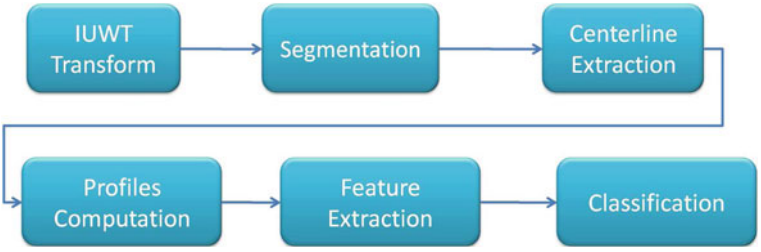

Fig. 2 Retinal image analysis pipeline

not for the construction or tuning of the segmentation algorithm. Overall, algorithms based on supervised classification report better segmentation results with a computational overhead due to training process. They are also dependent on the sample used for training and may suffer great errors on outlier retinal images. On the other hand, unsupervised techniques are less computationally demanding but difficult to tune, and sensitive to unexpected variations in the images. The vascular segmentation algorithm used in this chapter follows the approach proposed in [2], which uses an unsupervised and fast segmentation mechanism. The vessel pixel discrimination must be performed following a supervised learning approach, where specific features are proposed to exploit subtle image differences of the artery and vein vessels [13]. In this chapter we concentrate on the comparison of Extreme Learning Machine (ELM) [8] approaches against other state-of-the-art classifier building approaches for this task.

The chapter is structured as follows: Sect. 1 gives a brief overview of approaches dealing with retinal microvasculature analysis in fundus images. Section 2 presents the image processing pipeline for retinal vessels and the feature extraction approach for artery/vein discrimination, as well as a brief description of implementation details. Section 3 gives a brief review of ELM as supervised classification technique applied to the problem of artery/vein classification. Section 4 reports obtained classification results obtained with different supervised classification approaches. Finally, Sect. 5 gives a discussion about the implemented and tested approach, and addresses next lines of research and development.

2 Feature Extraction

Figure 2 depicts the image processing pipeline implementing our approach. First we perform the vessel segmentation. After image acquisition, we first perform a field of view (FOV) detection, selecting the region-of-interest for the following processes. Next, we apply an isotropic undecimated wavelet transform (IUWT) [2] at several wavelet scales. This transformation achieves contrast enhancement, increasing the difference between structures of different luminance value. We apply a thresholding operation on the IUWT contrast enhanced images, followed by a connected

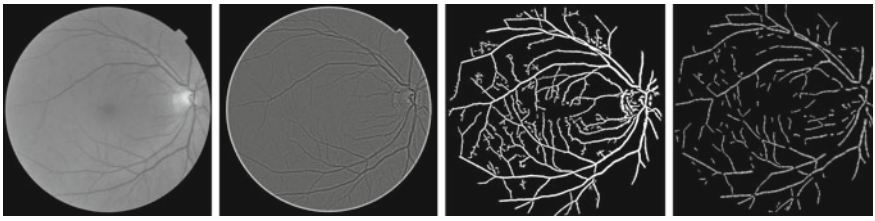

Fig. 3 Sequence of partial results of vessel segmentation. *Left to right* original image, FOV detection, IUWT contrast enhancement, vessel detection

Fig. 4 Extracted vessel profiles overlaid on the input image

component analysis removing spurious small connected components, falling below the minimum length of a vessel candidate to be measured. Segmentation threshold is set to 15–20 % of the lowest luminance value inside the region of interest (FOV). This value over-segments the images, ensuring that most of the vessel tree is retained. The center line is obtained by reducing vessel regions to one-pixel-wide skeletons using a thinning algorithm. Afterward, a branch detector is applied in order to identify and separate vessel bifurcations and vessels segment, and obtained vessel segments are approximated with B-spline curves for regularization, and curvature and section computation. Finally, we use a Full Width at Half Maximum (FWHW) algorithm to estimate vessel caliber along such sections. Figure 3 shows a sequence of partial results of the vessel segmentation. On the localized vessels, sections that are perpendicular to the local orientation of the B-spline representing the vessel are draw at regular intervals of the vessel centerline as shown in Fig. 4. These sections are then used for image feature extraction for vessel type discrimination by classification, prior to AVR calculation.

Several studies [15, 19] show that only photometric features are useful for artery/vein classification. Morphometric features such as width or tortuosity are pathological biomarkers, thus may change severally depending if the patient has a potential disease or not. Therefore, in our study we define only photometric features based on pixel luminance and chrominance information. More precisely, we extract the following features:

- Mean and standard deviation of green and red value in RGB color space along the vessel segment. We excluded blue channel because its signal-to-noise-ratio is very low compared with the other two, thus does not add discriminant capabilities for the classification.

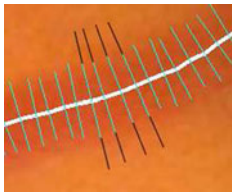

Fig. 5 Localization of outer and inner pixels along the sections drawn perpendicular to the vessel centerline

- Along the perpendicular sections we distinguish two parts, illustrated in Fig. 5: outer pixels lying at a distance from the centerline above 40 % of the estimated vessels width, depicted by black segments in Fig. 5, inner pixels lying at a distance below this threshold, depicted by white segments in Fig. 5. We compute the difference of the means of of green and red channels of the outer and inner pixels. These features model the contrast between foreground, i.e. vessels, and background.
- Mean and standard deviation of Hue channel in HSV color space along the vessel segment.
- Mean luminance inside the vessel, and the difference in luminance between outside and inside the vessel.

3 Extreme Learning Machines

3.1 Basic ELM

The Extreme Learning Machine (ELM) [7] is a very fast training algorithm for single-layer feedforward neural networks (SLFN). The key idea of ELM is the random initialization of the SLFN hidden layer node weights. Consider a set of M data samples $(\mathbf{x}_i, y_i)$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \Omega$. Then, a SLFN with N hidden neurons is modeled as the following expression:

$$\mathbf{y} = \Phi(\mathbf{x}) = \sum_{i=1}^{N} \beta_i \, f(\mathbf{w}_i \cdot \mathbf{x} + b_i), \; j \in [1, M], \tag{1}$$

where $f(x)$ is the activation function, $\mathbf{w}_i$ the input weights to the i-th neuron in the hidden layer, b_i the hidden layer unit bias and β_i are the output weights. The application of this equation to all available data samples can be written in matrix form as

$$\mathbf{H}\beta = \mathbf{Y},$$

where $\mathbf{H}$ is the hidden layer output matrix defined as the output of the hidden layer for each input sample vector, $\beta = (\beta_1 \ldots \beta_N)^T$ and $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_M)^T$.

The way to calculate the output weights β from the hidden-layer to the target values is computing the Moore–Penrose generalized inverse of the matrix $\mathbf{H}$, denoted as $\mathbf{H}^{\dagger}$. The mean least squares solution is $\beta = \mathbf{H}^{\dagger}\mathbf{Y}$.

The orthogonal projection method can be used to calculate the pseudo-inverse. In the case of $\mathbf{H}\mathbf{H}^T$ being non-singular, $\mathbf{H}^{\dagger}$would be obtained by $\mathbf{H}^{\dagger} = \mathbf{H}^T(\mathbf{H}^T\mathbf{H})^{-1}$. Thus, the output weights β are calculated

$$\beta = \mathbf{H}^T \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{Y}.$$

According to ridge regression theory [6], it was suggested [18] that Thikonov regularization [17] can be used to have better generalization performance. This regularization is achieved by adding a positive value $1/\lambda$ to the diagonal of $\mathbf{H}\mathbf{H}^T$. The calculation of the output weights is

$$\beta = \mathbf{H}^T \left(\frac{\mathbf{I}}{\lambda} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{Y}.$$

In our experiments, the basic ELM is denoted as "ELM", and the regularized ELM is denoted as "ELM(w/regul)". The implementation of both ELMs is available at [5].

3.2 OP-ELM

The Optimally Pruned Extreme Learning Machine (OP-ELM) was proposed in [12] with the goal of solving the problem that ELM faces with highly correlated variables. The basic ELM does not cope well with variables irrelevant to the problem at hand. The OP-ELM proposes a three-steps methodology, to address this problem:

1. Construct an SLFN using ELM.
2. Rank the best neurons using LARS algorithm. This process is akin to a "regularization" of the ELM. It uses Allen's PRESS [3] formula to L_1 regularize the ELM.
3. Select the optimal number of neurons using Leave-One-Out (LOO) criterion.

The LOO method is usually costly, since it requires to train the model on the whole data set except one sample for all the samples of the data. However, in the OP-ELM the situation is linear between the hidden layer and the output one. The LOO error has a closed matrix form, given by the PRESS method [3]. This closed form allows a fast computation of the MSE, and therefore the computation of the output weights is still computationally fast, and theoretically more robust than the original ELM to correlated variables. The code of OP-ELM is made available by Miche et al. at [1].

Table 1 Classification mean accuracy results from 10-fold cross-validation

Classifier	Accuracy
Naive bayes	82.7
MLP	91.1
SVM (Lineal)	73.3
SVM (RBF)	92.5
ELM	89.4
ELM (w/regul)	90.5
OP-ELM	**93.6**

Table 2 Training and testing times

Classifier	Training time	Testing time
Naive Bayes	0.05	0.01
MLP	4.64	0.01
SVM (Lineal)	15.49	0.07
SVM (RBF)	1.94	0.09
ELM	3.91[a]	0.03
ELM (w/regul)	1.11	0.04
OP-ELM	55	0.02

[a]Note that the greater training time of ELM compared to ELM(w/regul) is due to the use of SVD on the calculation of the pseudo-inverse in the case of ELM

4 Results

This section shows the comparative results obtained during the retinal vessels classification experiment. For this study we used the feature vectors of 5730 vessel sections, extracted from several images which have been labeled as arteries or veins by two human experts. For this evaluation we used several supervised classification approaches implemented in the public available Weka software http://www.cs.waikato.ac.nz/ml/weka/, version 3.7.9. We set each classifier learning algorithm parameters to their default values. In this evaluation we tested single classifier approaches, thus we did not include ensemble approaches such as Random Forest.

The results of 10-fold cross-validation experiment for each algorithm are summarized in Table 1. Worst results were obtained by SVM with linear kernel, hence indicating that the best decision boundary between artery and vein classes is not linear. OP-ELM obtains the best classification accuracy, followed by MLP and SVM with non-linear RBF kernel. Table 2 shows training and testing times of tested classifiers. As expected a simple Naive Bayes classifier is the fastest approach, while OP-ELM is the slowest approach. However, regarding testing times OP-ELM is one of fastest approaches. In our case, testing times are more important than training times, because our retinal quantification application is oriented to carrying out large population screening programs, where small differences in testing times will be amplified by the population size.

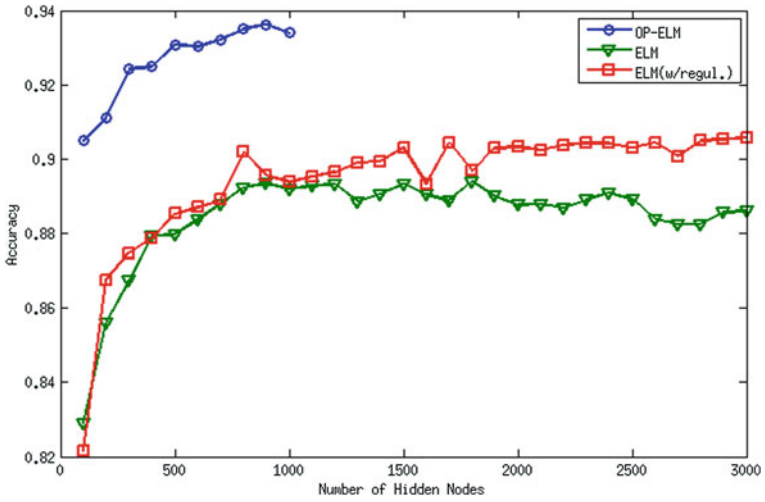

Fig. 6 Accuracy results for increasing hidden layer sizes

Figure 6 shows the results of an experiment using ELM, ELM with regularization and OP-ELM by evaluating classifier accuracy against the number of hidden nodes. As can be seen, OP-ELM outperforms basic ELM with and without regularization. Moreover, OP-ELM requires many fewer hidden nodes before convergence, compared with basic ELM.

5 Conclusion and Feature Work

In this chapter we have introduced a system for retinal image vessel segmentation and classification. Classifying retinal vessels into arteries or veins is a crucial step for retinal image quantification based on the extraction of biomarkers such as vessels tortuosity or arterio-venular ratio (AVR). Therefore, the final supervised classifier is a key element of this system. We have performed a comparative experiment between state-of-the-art classifiers and Extreme Learning Machines (ELM). Our results shows that the approach based on Op-ELM outperforms other supervised classification approaches such as SMV or MLP, in terms of accuracy and testing times.

In the future, we plan to implement an hybrid approach for retinal vessels classification, by fusing a supervised Classification using OP-ELM with unsupervised classification by using Fuzzy K-means. This approach would try to overcome the problems arising from the presence of inter-image contrast and luminosity variability, that are difficult to cope with a single Supervised Classification approach.

References

1. Available online at EIML Group. The op-elm toolbox. http://www.cis.hut.fi/projects/eiml/research/downloads/op-elm-toolbox 2009
2. P. Bankhead, C.N. Scholfield, J.G. McGeown, T.M. Curtis, Fast retinal vessel detection and measurement using wavelets and edge location refinement. PloS one **7**(3), e32435 (2012)
3. M.A. David, The relationship between variable selection and data augmentation and a method for prediction. Technometrics **16**(1), 125–127 (1974)
4. M.M. Fraz, P. Remagnino, A. Hoppe, B. Uyyanonvara, A.R. Rudnicka, C.G. Owen, S.A. Barman, Blood vessel segmentation methodologies in retinal images-a survey. Comput. Methods Programs Biomed. **108**(1), 407–433 (2012)
5. Guangbin Huang. ELM Homepage. http://www.ntu.edu.sg/home/egbhuang/elm_codes.html 2012
6. A.E. Hoerl, Application of ridge analysis to regression problems. Chem. Eng. Prog. **58**, 54–59 (1962)
7. G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications. Neurocomputing **70**(1–3), 489–501 (2006)
8. G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification. IEEE Trans. Syst. Man Cybern. **42**(2), 513–529 (2012)
9. D. Larry, D.L. Hubbard, R.J. Brothers, W.N. King, L.X. Clegg, R. Klein, L.S. Cooper, A.R. Sharrett, M.D. Davis, J. Cai, Methods for evaluation of retinal microvascular abnormalities associated with hypertension/sclerosis in the atherosclerosis risk in communities study. Ophthalmology **106**(12), 2269–2280 (1999)
10. G. Liew, J.J. Wang, P. Mitchell, T.Y. Wong. Retinal vascular imaging a new tool in microvascular disease research. Circ. Cardiovasc. Imaging **1**(2), 156–161 (2008)
11. G. Liew, J.J. Wang, Retinal vascular signs: a window to the heart? Revista Española de Cardiología (English Edition) **64**(6), 515–521 (2011)
12. Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, Op-elm: Optimally pruned extreme learning machine. IEEE Trans. Neural Networks **21**(1), 158–162 (2010)
13. M. Niemeijer, B. van Ginneken, M.D. Abràmoff, Automatic classification of retinal vessels into arteries and veins. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, 72601F (2009)
14. E. Ricci, R. Perfetti, Retinal blood vessel segmentation using line operators and support vector classification. IEEE Trans. Medical Imaging **26**(10), 1357–1365 (2007)
15. M. Saez, G-V. Sonia, G-P. Manuel, M.A. Barceló, P-S. Marta, G. Coll de Tuero, P-R. Antonio, Development of an automated system to classify retinal vessels into arteries and veins. Comput. Methods Programs Biomed. **108**(1), 367–376 (2012)
16. T.T. Thanh, T.Y. Wong, Retinal vascular manifestations of metabolic disorders. Trends Endocrinol. Metab. **17**(7), 262 (2006)
17. A. Tikhonov, Solution of inc orrectly formulated problems and the regularization method. Sov. Math. Doklady **5**, 1035–1038 (1963)
18. K.-A. Toh, Deterministic neural classification. Neural Comput. **20**(6), 1565–1595 (2008)
19. A. Zamperini, A. Giachetti, E. Trucco, K.S. Chin, Effective features for artery-vein classification in digital fundus images. 25th International Symposium on Computer-Based Medical Systems (CBMS), 1–6 (2012)

Demographic Attributes Prediction Using Extreme Learning Machine

Ying Liu, Tengqi Ye, Guoqi Liu, Cathal Gurrin and Bin Zhang

Abstract Demographic attributes prediction is fundamental and important in many applications in real world, such as: recommendation, personalized search and behavior targeting. Although a variety of subjects are involved with demographic attributes prediction, e.g. there are requirements to recognize and predict demography from psychology, but the traditional approach is dynamic modeling on specified field and distinctive datasets. However, dynamic modeling takes researchers a lot of time and energy, even if it is done, no one has an idea how good or how bad it is. To tackle the problems mentioned above, a framework is proposed in this chapter to predict using classifiers as core part, which consists of three main components: data processing, predicting using classifiers and prediction adjustments. The component of data processing performs to clean and format data. The first step is extracting relatively independent data from complicated original dataset. In the next step, the extracted data goes through different paths based on their types. And at the last step, all the data will be transformed into a demographic attributes matrix. To fulfill prediction, the demographic attributes matrix is taken as the input of classifiers, and the testing

Y. Liu (✉)
School of Software Engineering, Northeastern University, Shenyang 110819, China
e-mail: Liuy@swc.neu.edu.cn

T. Ye · C. Gurrin
School of Computing, Dublin City University, Dublin, Ireland
e-mail: yetengqi@gmail.com

C. Gurrin
e-mail: cathal@gmail.com

G. Liu
School of Computing, Northeastern University, Shenyang 110819, China
e-mail: liuguoqi@mail.neu.edu.cn

B. Zhang
College of Information Science and Engineering, Northeastern University,
Shenyang 110819, China
e-mail: zhangbin@mail.neu.edu.cn

F. Sun et al. (eds.), *Extreme Learning Machines 2013: Algorithms and Applications*, Adaptation, Learning, and Optimization 16, DOI: 10.1007/978-3-319-04741-6_11,

dataset comes from the same matrix as well. Classifiers in the experiments includes conventional state-of-the-art ones and Extreme Learning Machine, a new outstanding classifier. From the results of experiments based on two unique datasets, it is concluded ELM outperforms others. In the stage of prediction adjustments, two kinds of adjustments strategies are proposed corresponding to single target attributes and multiple target attributes separately, where single target attributes adjustments strategies include: adjusting the parameters of classifiers, adjusting the number of classes of target attributes and adjusting the public attributes. And multiple target attributes adjustment utilizes the outputs of first prediction as the inputs of second prediction to improve the accuracy of the first prediction. The framework proposed in this chapter consumes less time compared with traditional dynamic modeling methods, and there is no need to fully study the knowledge in various subjects for researchers using the framework because of the regular patterns. In addition, adjustment strategies have no restriction on the datasets; hence it will be useful universally. However, in some cases, dynamic modeling has the advantage of precision, resulting in better accuracy, but the results from the framework proposed in the chapter could provide as a comparison. In this work, a universal demographic attributes prediction framework is proposed to work on a variety of dataset with Extreme Learning Machine (ELM). The framework consists of three main components: First, processing raw data and extracting attribute features depending on different data types; Second, predicting desired attributes by classification; Third, improving the accuracy of classifiers through various adjustment strategies. Two experiments of different data types on real world prediction problems are conducted to demonstrate our framework can achieve better performance than other traditional state-of-the-art prediction methods with respect to accuracy. *abstract* environment.

Keywords Demographic attributes prediction · Extreme learning machine

1 Introduction

Demographic attributes prediction is to predict desired attributes information of human after gathering and analyzing all the attributes information of others. It is important and fundamental for many applications, such as recommendation, personalization, and behavior targeting [1]. Some but not many chapters and methods have been proposed to perform demographic attributes prediction [2, 3]. However, almost all methods require constructs diverse models depending on different datasets; while rest methods apply similar model on different curriculum. After performing discretization on raw data, the key of prediction is classification. Thus, as an excellent classifier, Extreme Learning Machine is taken as an important section in the model which could apply on diverse dataset and situations with little modification.

In the last few years, demographic attributes prediction attracted great attention from all over the world. In January 2009 Nokia Research Center Lausanne and its Swiss academic partners Idiap and EPFL started gathering demographic data through mobile phones and afterwards hold Mobile Data Challenge based on the

data [4]. There were three tasks in the Challenge and the third one was about demographic attributes prediction. Sanja Brdar [5] proposed k-nearest neighbors, radial basis function network and random forest as classification access. Kaixiang Mo [6] proposed Support Vector Machine to perform classification, which obtains better accuracy. NishKam Ravi [7] studied activity recognition from accelerometer data which accessed amazing prediction accuracy.

In previous work, almost all researches have analyzed their collected dataset and construct corresponding model which could not be applied to other datasets. The main contribution of this chapter is a model proposed which could be widely applied to variety of demographic attributes prediction with only little adjustment in the data processing stage. Raw data is divided into two groups: continuous variables and discrete variables. The two groups are different in data processing stage but similar in prediction stage. Concluded from the results, although the model is universal, it can produce fabulous accuracy sometimes.

The rest chapter is organized as follows: Sect. 2 briefly introduces ELM and SVM algorithms. Section 3 presents process on raw datasets. Section 4 demonstrates prediction and prediction adjustment. Section 5 compares experimental results of ELM with that of SVM. Conclusions and future work are in Sect. 6.

2 Brief Introduction of ELM

Countless classification theories and methods have been proposed to solve the problem and a lot of them are successful. One of the goals of this work is to compare the performance of Extreme Learning Machine with other classifiers [8].

Traditional learning speed of feedforward neural networks is in general far slower than required and it has been a major bottleneck in their applications for past decades [9]. Mainly because of two reasons: (*a*) Usual gradient-based learning algorithms are slow; and (*b*) all the parameters of the networks are tuned iteratively by using such algorithms. To tackle the issues and improve learning speed of feedforward neural networks [10], Huang et al proposed Extreme Learning Machine from single-hidden layer feedforward neural networks (SLFNs) which can randomly select the initial values for the hidden layer bias and input weights at the condition that the activation functions are infinitely differentiable [11].

For N arbitrary distinct samples $(\mathbf{x_i}, \mathbf{t_i})$, where $\mathbf{x_i} = [x_{i1}, x_{i2}, \cdots, x_{in}]^T \in \mathbf{R}^n$ are data vectors and $\mathbf{t_i} = [t_{i1}, t_{i2}, \cdots, t_{in}]^T \in \mathbf{R}^m$ are the target classes, standard SLFNs with $\tilde{N}$ hidden nodes and activation function $g(x)$ can be mathematically modeled as

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i(\mathbf{x_j}) = \sum_{i=1}^{\tilde{N}} \beta_i g(\mathbf{w_i} \cdot \mathbf{x_j} + b_i) = \mathbf{o_j}, \quad j = 1, \cdots N \tag{1}$$

where $\mathbf{w_i} = [w_{i1}, w_{i2}, \cdots, w_{in}]^T$ is the weight vector connecting the ith hidden node and the input nodes, $\beta_\mathbf{i} = [\beta_{i1}, \beta_{i2}, \cdots, \beta_{im}]^T$ is the weight vector connecting the ith hidden node and the output nodes, and b_i is the threshold of the ith hidden node. In addition [12], $\mathbf{w_i} \cdot \mathbf{x_j}$ represents the inner product of $\mathbf{w_i}$ and $\mathbf{x_j}$. And Extreme Learning Machine with $\tilde{N}$ hidden nodes with activation function $g(x)$ will approximate these N samples with zero error means that $\sum_{i=1}^{\tilde{N}} \left|\left|\mathbf{o_j} - \mathbf{t_j}\right|\right| = 0$, i.e., there exist β_i, $\mathbf{w_i}$ and b_i such that

$$\sum_{i=1}^{\tilde{N}} \beta_i g(\mathbf{w_i} \cdot \mathbf{x_j} + b_i) = \mathbf{t_j}, \quad j = 1, \cdots N \tag{2}$$

The above N equations can be denoted in the form of matrix as

$$\mathbf{H}\beta = \mathbf{T} \tag{3}$$

where

$$\mathbf{H}(\mathbf{w_1}, \cdots \mathbf{w}_{\tilde{\mathbf{N}}}, \mathbf{b_1}, \cdots \mathbf{b}_{\tilde{\mathbf{N}}}, \mathbf{x_1}, \cdots \mathbf{x_N}) = \tag{4}$$
$$\begin{bmatrix} g(\mathbf{w_1} \cdot \mathbf{w_2} + b_1) & \cdots & g(\mathbf{w}_{\tilde{\mathbf{N}}} \cdot \mathbf{x_1} + b_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ g(\mathbf{w_1} \cdot \mathbf{w_N} + b_1) & \cdots & g(\mathbf{w}_{\tilde{\mathbf{N}}} \cdot \mathbf{x_N} + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m} \quad and \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_{\tilde{N}}^T \end{bmatrix}_{N \times m} \tag{5}$$

and $\mathbf{H}$ is the hidden layer output matrix of the neural network; the ith column of $\mathbf{H}$ is the ith hidden node output with respect to inputs $\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_N}$.

3 Data Process

As different raw datasets are usually in different patterns and more or less contain some errors or exceptions, processing the raw datasets before taking them as prediction input is necessary and important. In addition, processing before predicting will improve the accuracy of result and efficiency of prediction. Raw data can be divided into two types: continuous variables and discrete variables. In fact, there is no continuous value in continuous variables. Continuous variables contain observations with no obvious intervals. Data process is shown as Fig. 1.

In our model, demographic attributes matrix provides as direct input of classifiers for prediction. As original datasets are complicated and differs from each other, our

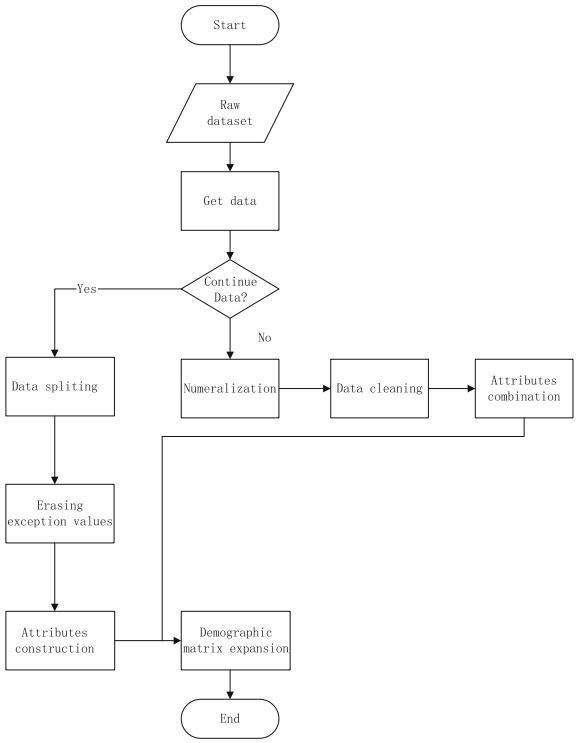

Fig. 1 Process on different types of data

model will transform the original datasets to demographic attributes matrices in the data processing stage. Data cleaning and attributes constructions are two main goals in data processing, as shown in Fig. 1. The first step is to extract logical consistent data from the original dataset. Continuous data and discrete data will go through different processing, which will be detailed explained later. After all the processing, at the end, demographic attributes matrix will be constructed for prediction.

Logical consistent data refers to data of same meaning: data from an extraction could be of same attributes from all people or of same attributes from one person, e.g. phone call records of same person or all sex choices from investigation. After an extraction, the following process differs depending on different data types, continuous data or discrete data. Normally speaking, discrete data comes from data of same attributes from group of people.

3.1 Demographic Attributes Matrix Representation

A demographic attributes matrix is a matrix represents information of a group of individuals, where each line vector is the attributes set of corresponding individual and each column vector represents corresponding attribute of every individuals.

Demographic attributes matrix is denoted by D and D_i^j denotes the element of ith line and jth column, representing the value of ith individual on jth attribute. The individual in the matrix could be other than person. If attributes of people are to be predicted, the individual in the matrix refers to person; while attributes between people, like closeness level, are to be predicted, the individual refers to pair of people.

To construct a demographic attributes matrix, three steps should be done: individuals identifying, attributes selecting and determining every elements in it. Individuals identification is easy to operate based on what to be predicted, while the other two steps are more difficult and more complex.

3.2 Continuous Variables Process

Data collected by sensors, such as accelerators, usually consists of continuous variables. As classifier is not able to work directly on continuous variables, several statistical characteristics are selected to represent the raw data. Resulted from the problem of sensors and rare situations, few data are extreme values, mostly extremely large. Although the number is few, these exceptions would have great impact on statistical characteristics presenting the original data. Thus, exceptions would be erased before feature construction. In addition, as enough number of testing cases is necessary for classifier to learn, the original data should be split wisely. In conclusion, continuous raw data will go through data splitting, data cleaning and feature construction in turn.

Hampel identifier are widely used to detect extreme values which defines Z':

$$Z' = \frac{|X_i - Median|}{\left(\frac{MAD}{.6745}\right)} \tag{6}$$

where X_i is each observation in the dataset, Median is the median of the X_i, and MAD is median absolute deviation (MAD) between X_i and Median. In the experiments, Hampel identifier is improved to perform much better according to the characteristic of the data. Pseudocode is shown below to utilize the improved Hampel identifier to detect exception values:

EXCEPTION_DETECTION($[x_1, x_2, \cdots, x_n]$)

1 $Exception\_detection \leftarrow []$ ▷ exception set is empty at the beginning
2 $med \leftarrow Median([x_1, x_2, \cdots, x_n])$ ▷ find median of input
3 **for** x_i in $[x_1, x_2, \cdots, x_n]$
4 **do** $x_i' = |x_i - median|$
5 $MAD \leftarrow Mean([x_1', x_2', ..., x_n'])$ ▷ figure out average distance
6 **for** x_i' in $[x_1', x_2', ..., x_n']$
7 **if** $\frac{|x_i - median|}{\frac{MAD}{0.6745}} > threshold$ ▷ compare the result with threshold
8 **then** $exception\_set \leftarrow [exception\_set, x_i]$

The MAD in the pseudocode prefers to mean absolute deviation instead of median absolute deviation in original Hampel identifier.

There are several statistical characteristics could represent original data, such as average number, variance and median, etc. Sometimes, rate and percentage could be more useful.

3.3 Discrete Variables Process

Discrete variables normally come from surveys or counters instead of sensors. Not like continuous variables, there is no extreme value in discrete variables, but instead null values may appear. Not all null values will affect the result, but those may affect will be deleted. Since some questions in survey have connections, related attributes will be erased as well. As answers from investigation in sentences could not be directly used in classifier, numeralization will turn the sentences into numbers before classification. At classification stage, attributes should be independent of each other; thus, related attributes should be combined into one. For a question, if all answers are of strings, they are numbered in continual positive integers. If all answers are of numbers, they remain. If in continuous data, data cleaning focuses on cleaning extreme values, however, null values are focused in discrete data.In continuous data, data cleaning focus on cleaning extreme values, however, null values are focus in discrete data.

Assume there are n questions, Q^i is the ith question. If the number of answers to Q^i is limited and values of answers are discrete, all the choices of answers are $< C_i^1, C_i^2, C_i^k >$, where k is the number of choices. If the choice of Q^i will influence the scope of choices of Q^j, Q^j depends on Q^i. If Q^c depends on Q^b and Q^b depends on Q^a, then Q^c depends on Q^a. Situations of circular dependency rarely happen, so they are not in the consideration, like Q^i depends on Q^j and Q^j depends on Q^i. The dependency relationship can be one to many, many to one or many to many. A dependency cluster contains elements where each two have dependency relationship and no one has relationship with elements on the outside.

Based on the assumptions above, it is feasible to combine all the answers of a dependency cluster to only one attribute. Because the number of questions is limited and there is no circular dependency, there will be elements which depend on no one and elements which no one depends on them. Combination algorithms are showed below:

COMBINATION($Q^1, Q^2, \cdots, Q^n$)

```
find elements which no one depends on them, D
for each Q^j in U
    do n_j ← 0
        for each C_i^k in answer of Q^i ▷ find all choices
            do for each C_j^k in answer of Q^j
                do if it is possible C_i^k and C_j^k appears at the same time
                    then n_j ← n_j + 1
        < C_j^1, C_j^2, ··· , C_j^n > ←< 1, 2, ··· , n_j > ▷ rearrange choices
D ← D − {Q^i}
if Q^j depends on other elements
    then D ← D + {Q^i}
```

4 Prediction Using Different Classifiers and Prediction Adjustment

After previous work on raw data, data at this stage is in the form of matrix, each row of which is an observation and each column is an attribute. The whole data will go into two dataset: training dataset and testing dataset.

It is possible that our universal model could produce results with accuracy much lower than expected. There are three possibilities which could cause the problem: (*a*) the attributes are unpredictable; (*b*) classification is not suitable for predicting the attributes; (*c*) processing on the data is not suitable not enough or the original dataset is fake. Although in the first two conditions, there is a great chance traditional dynamic modeling on specified dataset outperforms our model, it also could provide as a comparison to indicate how well the traditional model is. In the last condition, some adjustments could be done to improve the accuracy of classifiers.

4.1 Demographic Prediction Analysis

The columns of a demographic attributes matrix are consist of various attributes, some are always directly accessible and the rest are not always known. The directly accessible attributes attributes in demographic attributes matrix D, are denoted by $D \cdot \boldsymbol{a}$. For the rest attributes in the matrix, sometimes the values of them are unknown, thus prediction is used to figure out the values of those target attributes. Target attributes in a demographic attributes matrix are denoted by $D \cdot \boldsymbol{t}$. If all values of target attributes are unknown, the corresponding demographic attributes matrix is predicting matrix, denoted by P. In the experiments, classifiers are used to predict $P \cdot t$. In fact, the results of prediction will never be known exactly, but accuracy is an important factor for evaluating the classifiers.

It is feasible to obtain accuracy rate by testing classifiers using known data. The classifiers provide as a mapping $y = f(x, c)$, where c is parameters unrelated with dataset. c can be worked out using $T \cdot \boldsymbol{t} = f(T \cdot \boldsymbol{a}, c)$, since T is entirely known. Thus $P \cdot \boldsymbol{t}$ is predicted through $P \cdot \boldsymbol{t} = f(P \cdot \boldsymbol{a}, c)$. To get the accuracy of the a classifier, after c is figured out, $V \cdot \boldsymbol{t}'$ results from $V \cdot \boldsymbol{t}' = f(V \cdot \boldsymbol{a}, c)$. And the the accuracy of the mapping is the accuracy between $V \cdot \boldsymbol{t}'$ and $V \cdot \boldsymbol{t}$.

Based on the number of labels, prediction can be divided into two types: predicting one attribute and predicting multiple attribute. And predicting multiple attributes is implemented by predicting one attribute separately multiple times.

4.2 Prediction with Various Classifiers

Concerning the number of target attributes, there are 2 types of prediction methods: single target attribute prediction and multiple target attributes prediction. Pseudocode of single attribute prediction is shown below:

Input: T, $P.\boldsymbol{a}$
Output: $P.\boldsymbol{t}$

SINGLE_TARGET_PREDICT(T, P)

1 $classifier\_train(T.\boldsymbol{t}, T.\boldsymbol{a})$ ▷ use training set train classifier
2 $P.t_i \leftarrow single\_target\_predict(P.\boldsymbol{a})$

As almost all classifiers are not able to predict multiple attributes directly, multiple attributes prediction is performing single target attribute prediction multiple times.

Decision tree [13], Naive Bayes algorithm and SVM [14] are used as other classifiers to compare with ELM in performance. Multiple target attributes prediction pseudocode is shown below:

Input: T, $P.\boldsymbol{a}$
Output: $P.\boldsymbol{t}$

MULTIPLE_TARGET_PREDICT(INPUT)

1 **for** $P.t_i$ in $P.\boldsymbol{t}$
2 **do** $P.t_i \leftarrow single\_target\_predict(T, P.\boldsymbol{a})$ ▷ utilize $single\_target\_predict$
3 $P.\boldsymbol{t} \leftarrow [P.\boldsymbol{t}, P.t_i]$

4.3 Prediction Adjustment Strategies

There are 2 types of prediction methods, so there are 2 types of prediction adjustment strategies: single target attribute prediction adjustment strategies and multiple target attributes prediction adjustment strategy. Because multiple target attributes prediction is based on single target attribute prediction, single target attribute prediction adjustment strategies could also influence multiple target attributes prediction.

Single target attribute prediction adjustments strategy

1. Adjusting the parameters of classifiers is able to improve the accuracy of prediction. Some classifiers require parameters unrelated with dataset to predict, like ELM. For them, trying different parameters leads to different accuracy and highest accuracy will be chosen.
2. Adjusting the attributes is able to improve the accuracy of prediction. Normally speaking, more attributes provide more information, which improve the accuracy of prediction. And sometimes, key attributes contribute most to accuracy. The process is in Fig. 2. In the process, for every loop, there is one more attribute to be added. If the accuracy decreases, the attribute will be erased. As a result, all chosen attributes perform positive effect to the final prediction.
3. Reducing the number of target attribute levels leads to improve accuracy of prediction. Normally the number of target attribute levels is smaller, the accuracy is higher. Pseudocode of target attribute levels adjusting is shown below, where k is the final number of levels and $[C_1, C_2, ..., C_n]$ is original class indexes:

Input: $[C_1, C_2, \cdots, C_n], k$
Output: $[C_1, C_2, \cdots, C_n]$

CLASSIFICATION(INPUT)

$range \leftarrow n/k$
$remainder \leftarrow n\%k$
for i in $[1, \cdots, (1 + range) * remainder]$ ▷ add remainder
    do $C_i \leftarrow (i - 1)/(range + 1)$
for i in $[(1 + range) * remainder + 1, \cdots, n]$
    do $C_i \leftarrow (i - (1 + range) * remainder)/range + remainder$

As the original number of target attributes levels may not be divided exactly, classes will expand 1 scope.

Multiple target attributes prediction adjustments strategy Normally, the accuracy of classifiers increases when the number of attributes increases. Thus, it is possible to improve the accuracy of multiple target attributes prediction by taking predicted target attributes as known attributes to predict other target attributes. Corresponding pseudocode is shown below:

Input: T, $P.\boldsymbol{a}$
Output: $P.\boldsymbol{t}$

MULTI_TARGET_IMPROVE(INPUT)

$P.\boldsymbol{t} \leftarrow multi\_target\_predict(Input)$▷ $utilizemulti\_target\_predict$
for t_i in $P.\boldsymbol{t}$
    do if $classifier\_vadiate(T, [T.\boldsymbol{a}, T.t_i], i) > classifier\_vadiate(T, T.\boldsymbol{a}, i)$
        then $t_i' \leftarrow single\_target\_predict([Input, t_1, \cdots, t_i - 1, t_i + 1, \cdots, t_m])$
            $t_i \leftarrow t_i'$

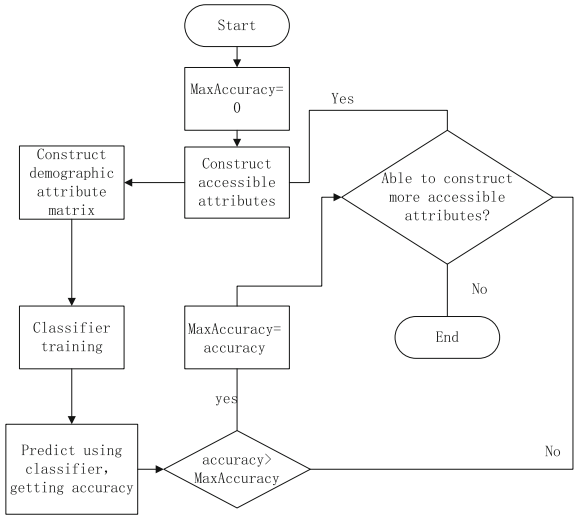

Fig. 2 Adjusting process of attributes

5 Experimental Results and Analysis

In this section, detailed experimental results of every components of model will be shown based on two datasets and performances of four different classifiers are compared for evaluation. There are three components in the model: data processing, classifying and adjusting classification, where performance comparisons between classifiers takes place in the classifying step. The four classifiers are decision tree, Naive Bayes algorithm, Support Vector Machine and Extreme Learning Machine.

In the experiments, all algorithms are implemented in Visual Studio 2010 and MATLAB R2012b. Experiments are run on a PC with Intel Corei5 2400, 3.10 GHz CPU, 3GB RAM and Windows XP operating system.

5.1 Raw Datasets Description

Two separate datasets are used to evaluate our model. The first dataset is full of accelerometer records and activity labels and the second dataset contains data from surveys and software applications in mobile phones.

The goal for first dataset is to predict the state of motions based on accelerometer records, which contains 7 labels of activity states: standing, walking, running, climbing up stairs, climbing down stairs, transporting and resting. Besides the labels, the original datasets mainly contains accelerations in three orthogonal directions and other support information, including timestamps, corresponding action labels and base station coordinates of recording. The 20.6 MB original dataset was split into matrix of 1436 observations [15].

Table 1 Files introduction from the second dataset

File name	Content	Collecting tool	Total line
closeness_adj.csv	Closeness value between each two informant	Survey	56
couples.csv	Marriage status and family information	Survey	56
BluetoothProximity.csv	Records of bluetooth contacts	Sensor	469924
SMS.csv	Records of SMS	Sensor	5301
VoiceCall.csv	Records of phone calls	Sensor	103468

There are two goals for second datasets: one is predicting closeness levels between arbitrary two people using data gathered through survey and data collected by software in cell phones; the other is predicting closeness levels and is couple or not between each pair using same data. It is assumed that is couple or not is known in first goal while unknown in second goal.

Instead of first dataset, data is collected by both surveys and sensors. The dataset has information of marriage status, closeness feeling, voice call records, message records and Bluetooth records. Marriage status includes which two is couple, sex of each one and how many children the one has. Closeness feeling indicates the closeness index with scope from 0 to 10. Voice call records shows when someone calls the other and if the other missed or not. Message records are similar to voice call but messages would surely arrive. Bluetooth records display when which two people get close enough. After decomposition, the 287 MB dataset consists of 5 files, detailed explanation is revealed in Table 1. Software applications on mobile phones are considered as sensor for convenience.

closeness_adj.csv consists of closeness levels between each two informants and there are 11 levels from 0 to 10, higher level indicating more close, where 0 denoting not known at all. couples.csv has information of which two informants is couple, how many children and sex. The software applications are more like counters rather than sensors: a Bluetooth contact, SMS receiving or sending and a phone call will trigger a recording.

5.2 Experimental Results of Data Processing

Exception detecting operations on first dataset is shown in Fig. 3. In the figure, red inverted triangle, blue circle and green point denote accelerations of three different orthogonal directions. The reason to utilize different shapes of different color is to distinguish each other. The black line in the figure is around 10^{11} of its original value to be distinguished from points below. And points above the black line are exception values.

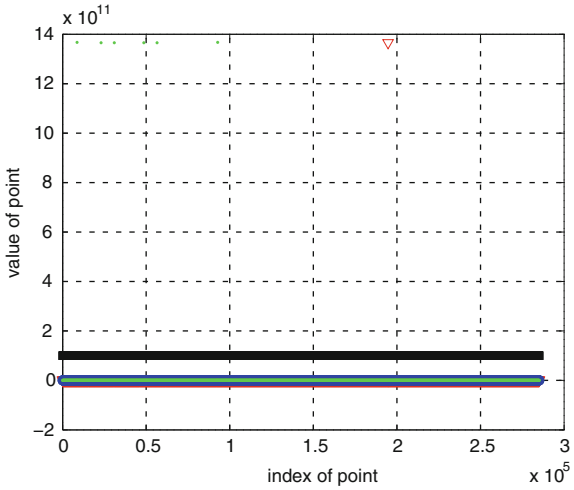

Fig. 3 Data processing of exceptions detection

Table 2 Performance of various classifiers using first dataset

Datasets	Training data	Training time	Testing data	Testing time	Accuracy
Decision tree	1436	0.0106	1436	0.0019	0.9582
Naive bayes	1436	0.1068	1436	1.5136	0.9582
SVM	1436	0.1829	1436	0.0248	0.9616
ELM	1436	0.3659	1436	0.1019	0.9999

5.3 Comparison Between Performances of Various Classifiers

In the section of classification, decision tree, Naive Bayes algorithm and Support Vector Machine provide as comparisons for Extreme Learning Machine.

1. Mean, standard deviation and median are selected as attributes from first dataset to predict motion states. There are 7 levels in target attribute, and RBF is chosen as activation function with parameter of 1, the result is shown in Table 2.
   Decision tree costs less training time and testing time than other algorithms, but with relatively low accuracy. Training time of Naive Bayes is about the same as that of others, but corresponding testing time is around hundred times of that of others. Accuracy of SVM is higher than that of decision tree and Naive Bayes, but consumes more training time. Accuracy of ELM is highest among 4 classifiers, but requires more time than SVM on both training time and testing time.
2. Is-couple, the number of children, ratio of Bluetooth contacts, ratio of SMS sending and receiving, and ratio of phone calls are selected as attributes from second dataset to predict closeness levels. There are 11 levels in target attribute, from 0 to 10, and RBF is chosen as activation function with parameter of 1, the result is shown in Table 3.

Table 3 Performance of various classifiers using second dataset to predict 11 target attribute levels

Datasets	Training data	Training time	Testing data	Testing time	Accuracy
Decision tree	3080	0.0115	3080	0.0185	0.8373
Naive bayes	3080	0.1257	3080	4.1247	0.8737
SVM	3080	0.0747	3080	0.0583	0.8386
ELM	3080	0.7819	3080	0.3874	0.8390

Table 4 Performance of various classifiers using second dataset to predict 3 target attribute levels

Datasets	Training data	Training time	Testing data	Testing time	Accuracy
Decision tree	3080	0.1136	3080	0.0012	0.8929
Naive bayes	3080	0.0482	3080	3.1450	0.8929
SVM	3080	0.4727	3080	0.0583	0.8996
ELM	3080	0.4988	3080	0.0731	0.9006

Decision tree costs less training time and testing time than other algorithms, but with lowest accuracy among 4 classifiers. Training time of Naive Bayes is about the same as that of others, but corresponding testing time is around hundred times of that of others, and it has the highest accuracy among 4 classifiers. Accuracy of ELM is second highest among 4 classifiers, but requires relatively more time than others.

3. Is-couple, the number of children, ratio of Bluetooth contacts, ratio of SMS sending and receiving, and ratio of phone calls are selected as attributes from second dataset to predict closeness levels. There are 3 levels in target attribute, from 0 to 2, which are classified from the original 11 levels. Linear kernel is chosen as activation function with parameter of 1, the result is shown in Table 4.
   Decision tree still costs less training time and testing time than other algorithms, but with relatively low accuracy. Training time of Naive Bayes is about the same as that of others, but corresponding testing time is around hundred times of that of others. Although ELM consumes more time on training and testing, its accuracy is highest among 4 classifiers, and only its accuracy is beyond 0.9.

5.4 Prediction Accuracy Results of Various Classifiers

There are single target attribute prediction adjustments strategy and multiple target attributes prediction adjustments strategy depending whether there is one target attribute or multiple target attributes to predict.

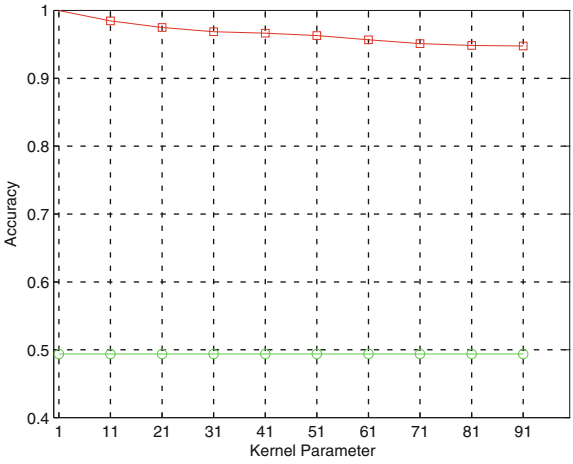

Fig. 4 Influence on accuracy with different parameters for ELM using first dataset (*Red bar*: RBF kernel, *green line*: linear kernel)

Single target attribute prediction adjustments strategy

1. Adjusting the parameters of classifiers is able to improve the accuracy of prediction. In this section, adjustment is performed on ELM using both datasets. Mean, standard deviation and median are selected as attributes from first dataset to predict motion states and there are 7 levels in target attribute. Under those conditions, linear kernel and RBF kernel is chosen separately as activation function with several corresponding parameters, as shown in Fig. 4.
   Using first dataset, RBF kernel performs much better than linear kernel and its accuracy gets higher with lower kernel parameter. However, the accuracy of linear kernel stays the same with different kernel parameters and its accuracy is much lower than that of RBF kernel. The highest accuracy from RBF kernel is close to 1, with kernel parameter 1.
   Is-couple, the number of children, ratio of Bluetooth contacts, ratio of SMS sending and receiving, and ratio of phone calls are selected as attributes from second dataset to predict closeness levels and there are 11 levels in target attribute, from 0 to 10. Under those conditions, linear kernel and RBF kernel is chosen separately as activation function with several corresponding parameters, as shown in Fig. 5.
   Using second dataset, accuracy of RBF kernel increases as corresponding kernel parameter decreases. However, the accuracy of linear kernel stays the same with different kernel parameters. The highest accuracy from RBF kernel is higher than that of linear kernel, but most values of RBF kernel is lower than linear kernel.
   Is-couple, the number of children, ratio of Bluetooth contacts, ratio of SMS sending and receiving, and ratio of phone calls are selected as attributes from second dataset to predict closeness levels and there are 3 levels in target attribute, from 0 to 2. The 3 levels comes from original 11 levels: 0 to 3 denotes almost unknown (0 now), 4–7 denotes known (1 now), 8–10 denotes known very well (2 now). Under those conditions, linear kernel and RBF kernel is chosen separately as

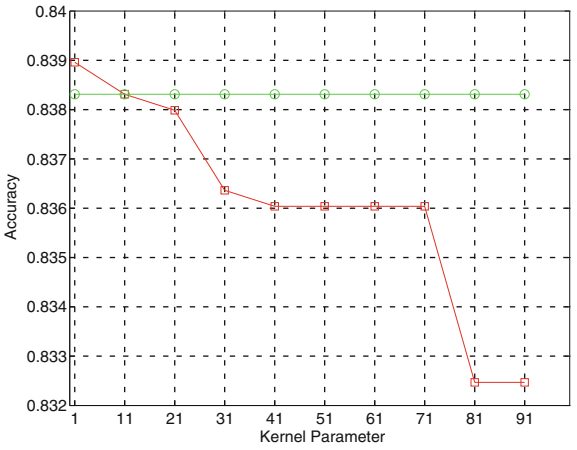

Fig. 5 Influence on accuracy with different parameters for ELM using second dataset with 11 target levels (*Red bar*: RBF kernel, *green line*: linear kernel)

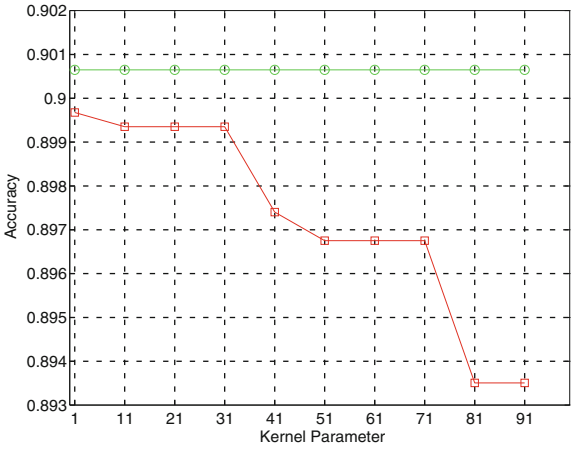

Fig. 6 Influence on accuracy with different parameters for ELM using second dataset with 3 target levels (*Red bar*: RBF kernel, *green line*: linear kernel)

activation function with several corresponding parameters, as shown in Fig. 6. Using second dataset, RBF kernel performs worse than linear kernel and its accuracy gets higher with lower kernel parameter. However, the accuracy of linear kernel stays the same with different kernel parameters and its accuracy is much higher than that of RBF kernel.

2. Adjusting the attributes is able to improve the accuracy of prediction. In the section, adjustment is performed on ELM using only first datasets.
   There are 7 levels in target attribute from first dataset to predict motion states. Different attributes combination are selected from mean, standard deviation and median. Linear kernel and RBF kernel is chosen separately as activation function with several corresponding parameters, as shown in Fig. 7. Every result shown in the figure is the mean from corresponding results of several different kernel parameters, because the highest accuracies are almost the same.

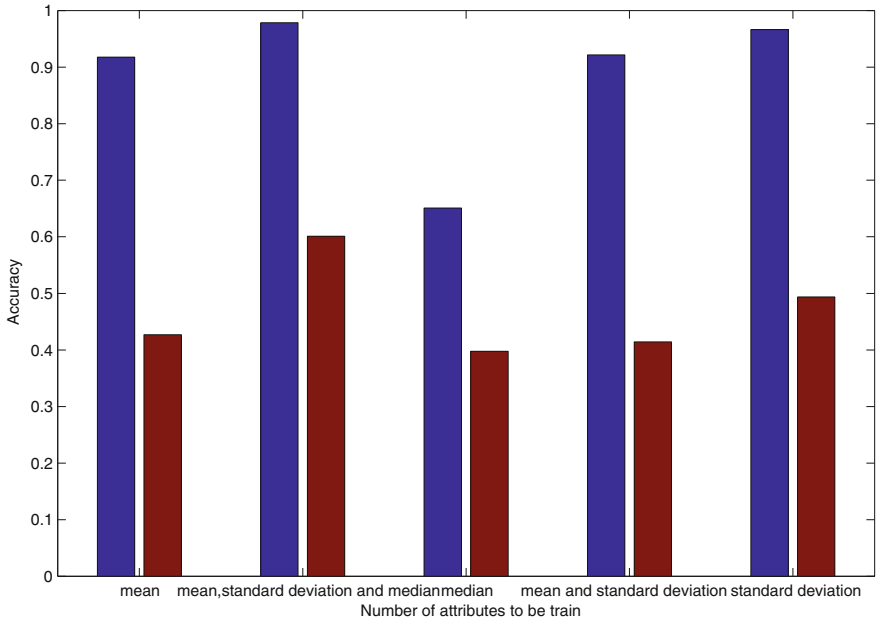

Fig. 7 Influence on accuracy with different parameters for ELM using first dataset (*Red bar*: RBF kernel, *green bar*: linear kernel)

Chosen attributes combinations are: mean, standard deviation, median, (mean and standard deviation), and (mean, standard deviation and median). It can be concluded from the figure that the last combination gets highest accuracy and normally, accuracy increase as the number of attributes increases.

3. Reducing the number of target attribute levels leading to advancing accuracy of prediction. In the section, adjustment is performed on ELM using only second datasets.
   The reason to use only second database is the target attribute levels in first dataset is fixed. Figure 8 shows the specified operation on changing the levels of target attribute from second dataset. There are original 11 levels and they are divided into 2, 3 and 5 levels as even as possible separately.
   Is-couple, the number of children, ratio of Bluetooth contacts, ratio of SMS sending and receiving, and ratio of phone calls are selected as attributes from second dataset to predict closeness levels. Every result shown in the figure is the highest from corresponding results of several different kernel parameters. It is concluded from Fig. 9 that accuracy of prediction increases as the number of target attribute levels decreases. However, in fact, 3 levels of target attribute is better than others, because index of emotions can not be too precise and 2 levels are too ambiguous.

Multiple target attributes prediction adjustments strategy As mentioned above, the accuracy of classifiers increases when the number of attributes increases. Thus, it

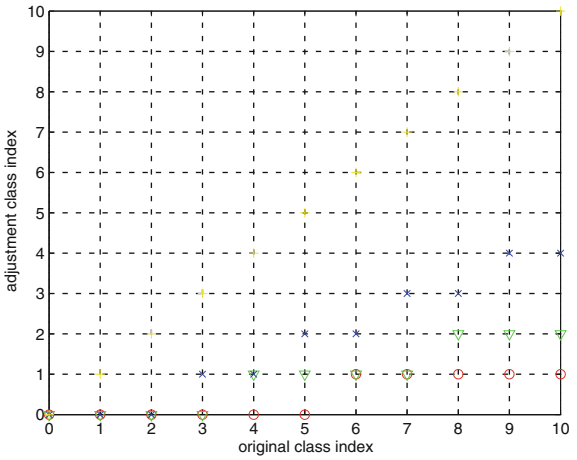

Fig. 8 Division on original 11 target attribute levels (*Red circle*: 2 levels; *green inverted triangle*: 3 levels; *blue cross*: 5 levels; *yellow* add sign: original 11 levels)

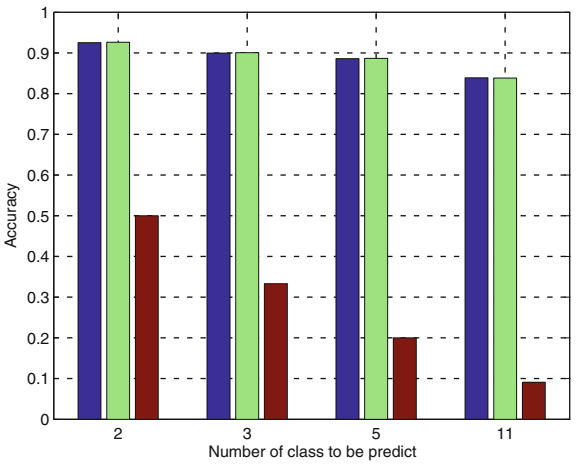

Fig. 9 Influence on accuracy with different number of target attribute levels for ELM using second dataset (*Red bar*: random algorithm; *green bar*: RBF kernel; *blue bar*: linear kernel)

is possible to improve the accuracy of multiple target attributes prediction by taking predicted target attributes as known attributes to predict other target attributes. The chosen predicted target attributes should be well predicted, corresponding to high prediction accuracy.

Only second dataset is used: assuming closeness levels and is-couple relationship are both target attributes. There are two choices of is-couple relationship, yes or no. And original 11 levels are chosen for closeness levels.

Firstly, two independent predictions are performed for the two target attributes separately using other known attributes: the number of children, ratio of Bluetooth contacts, ratio of SMS sending and receiving, and ratio of phone calls. Highest results are selected using ELM from different parameters in Table 5.

Table 5 Direct independent multiple target attributes prediction using ELM from different parameters

Target attribute	Kernel function	Kernel parameter	Accuracy
is-couple	RBF	1	0.9974
is-couple	RBF	11	0.9968
is-couple	linear	1	0.9655
is-couple	linear	11	0.9655
closeness	RBF	1	0.8377
closeness	RBF	11	0.8364
closeness	linear	1	0.8367
closeness	linear	11	0.8367

Table 6 Second time independent multiple target attributes prediction using data derived from Table 5

Target attribute	Kernel function	Kernel parameter	Accuracy
is-couple	RBF	1	0.9974
is-couple	RBF	11	0.9971
is-couple	linear	1	0.9961
is-couple	linear	11	0.9961
closeness	RBF	1	0.8377
closeness	RBF	11	0.8370
closeness	linear	1	0.8373
closeness	linear	11	0.8373

Table 7 Third time multiple target attributes prediction using complete attributes

Target attribute	Kernel function	Kernel parameter	Accuracy
is-couple	RBF	1	0.9974
is-couple	RBF	11	0.9968
is-couple	linear	1	0.9969
is-couple	linear	11	0.9969
closeness	RBF	1	0.8390
closeness	RBF	11	0.8383
closeness	linear	1	0.8383
closeness	linear	11	0.8383

Secondly, take is-couple as known attribute with other attributes to predict closeness levels and take closeness information for is-couple as well. Then predict both of them separately once again. Highest results are selected using ELM from different parameters in Table 6.

Compared with Table 5, almost all accuracies of is-couple some accuracy of closeness get higher and few accuracies of closeness become lower. The results show our multiple target attributes prediction adjustments strategy does work.

Thirdly, real is-couple value will be with other attributes to predict closeness levels and similar operation will be performed for is-couple as well. The highest results are selected using ELM from different parameters in Table 7.

There is no wonder almost all accuracies in Table 7 are higher than that of other tables, because real value is with 'prediction accuracy' of 1. And the results from our strategy are very close to directly prediction.

6 Conclusion and Future Work

The main contributions of our chapter includes:

1. constructing an universal model which could apply on a variety of dataset;
2. utilizing multiple state-of-art classifiers to compare with Extreme Learning Machine;
3. proposing some categories to improve the performance of Extreme Learning Machine.

Regarding our work, we are going to continue to improve the performance of our model based on ELM, which is intended to improve the accuracy of demographic attributes prediction. As to predicting action types based on accelerometer data, we are planning to increase the variety of action types. Besides predicting, more works will be done to study how to prevent predicting demographic information by protecting key demographic information.

Acknowledgments This work is supported by the National Natural Science Foundation of China under Grand No.61073062, No. 61100027, No.61202085, National Research Foundation for the Doctoral Program of Higher Education of China under Grand No. 20120042120010, Liaoning Province Doctor Startup Fund under Grand No.20111001, No.20121002, Fundamental Research Funds for the Central Universities under Grand No.N110417001No.N110417004.

References

1. J. Hu, H.-J. Zeng, H. Li, C. Niu, Z. Chen, Demographic prediction based on user's browsing behavior, in *Proceedings of the 16th international conference on World Wide Web* (ACM, 2007) pp. 151–160
2. S.M.S.J.T. Nadeem, M.C.Weigle, Demographic prediction of mobile user from phone usage. Age **1**, 16–21 (2012)
3. J.E. Blumenstock, D. Gillick, N. Eagle, Whos calling? demographics of mobile phone use in rwanda. Transportation **32**, 2–5 (2010)
4. J.K. Laurila, D. Gatica-Perez, I. Aad, J. Blom, O. Bornet, T.-M.-T. Do, O. Dousse, J. Eberle, M. Miettinen, The mobile data challenge: Big data for mobile computing research, pp. 1–8, 2012
5. S. Brdar, D. ulibrk, V. Crnojevi, Demographic attributes prediction on the real-world mobile data, in *MDC* 2012
6. E.Z. Kaixiang Mo, B. Tan, Q. Yang, Report of task 3: your phone understands you (2012)
7. N. Ravi, N. Dandekar, P. Mysore, M.L. Littman, Activity recognition from accelerometer data. Am. Assoc. Artif. Intell. **20**(3), 1541 (2005)
8. F. Cao, B. Liu, D. Sun Park, Image classification based on effective extreme learning machine. Neurocomputing **102**, 90–97, 15 February 2013

9. G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications. Neurocomputing **70**(1), 489–501 (2006)
10. C. Pan, D.S. Park, H. Lu, X. Wu, Color image segmentation by fixation-based active learning with elm. Soft Comput. **16**(9), 1569–1584 (2012)
11. X.-G. Zhao, G. Wang, X. Bi, P. Gong, Y. Zhao, Xml document classification based on elm. Neurocomputing **74**(16), 2444–2451 (2011)
12. I. Marques, M. Graña, Face recognition with lattice independent component analysis and extreme learning machines. Soft Comput. **16**(9), 1525–1537 (2012)
13. S.R. Safavian, D. Landgrebe, A survey of decision tree classifier methodology. Syst. Man Cybern. IEEE Trans. **21**(3), 660–674 (1991)
14. C.-W. Hsu, C.-J. Lin, A comparison of methods for multiclass support vector machines. Neural Networks IEEE Trans. **13**(2), 415–425 (2002)
15. N. Eagle, A. Pentland, Reality mining: sensing complex social systems. Pers. ubiquit. comput. **10**(4), 255–268 (2006)

Hyperspectral Image Classification Using Extreme Learning Machine and Conditional Random Field

Yanyan Zhang, Lu Yu, Dong Li and Zhisong Pan

Abstract Recent studies show that extreme learning machine (ELM) is a suitable, effective, and less time-consuming classifier with a wide range of applications. This chapter addresses the application of ELM to the remotely sensed hyperspectral image classification. In this chapter, the proposed hyperspectral image classification method consists of three steps: First, a semi-supervised feature extract algorithm is used for dimensionality reduction; Second, ELM is taken as a classifier; Finally, conditional random field (CRF) is taken to smooth the result of ELM classifier, where the probability estimation over each class obtained by ELM is used as unary potential function of CRF. The experimental results show that the proposed hyperspectral image classification method using both ELM and CRF achieves good classification performance on two real hyperspectral data sets in comparison to the methods using SVM and CRF.

Keywords Extreme learning machine · Support vector machine · Conditional random field · Semi-supervised dimensionality reduction · Hyperspectral image classification

1 Introduction

With the development of remote sensing systems, the resolution of remote sensing images is increasing improved. Hyperspectral images characterized by hundreds of different spectral bands contain a wealth of information on ground objects for researchers to extract. While the high spectral resolution involves some critical issues

Y. Zhang · D. Li · Z. Pan (✉)
College of Command Information Systems, PLA University of Science and Technology, Nanjing 210007, China
e-mail: hotpzs@hotmail.com

L. Yu
Institute of Communications Engineering, PLA University of Science and Technology, Nanjing 210007, China

F. Sun et al. (eds.), *Extreme Learning Machines 2013: Algorithms and Applications*, Adaptation, Learning, and Optimization 16, DOI: 10.1007/978-3-319-04741-6_12,

on the classification studies in hyperspectral images. The first issue is that the traditional classification methods for classifying hyperspectral images often cause Hughes Phenomenon [1] (which is intrinsic in high-dimensional data). In other words, when the dimension increases dramatically, the classification accuracy will not decrease only when there is a sufficient number of training samples. The second issue is that the independent training samples can not provide a complete description of hyperspectral images, discarding significant spatial information and resulting in many ill-posed classification problems. Namely, samples come from different classes with similar spectral features are often misclassified by the traditional *pixelwise* classifiers [2, 3] (i.e. support vector machine (SVM)). The third issue is related to the computational cost. Traditional classifiers (i.e. k nearest neighbor (k-NN) classifier and SVM) chosen to classify large hyperspectral images are usually time-consuming. For example, training a SVM on a whole image or a large training set may take a long time because the number of support vectors is often proportional to the number of training samples.

In the literature, the most direct way to avoid the problem of Hughes Phenomenon is dimensionality reduction. Existing dimensionality reduction methods can be roughly categorized into supervised, unsupervised and recently appeared semisupervised methods according to whether they use supervision information or not. One of the most popular supervised dimensionality reduction methods is linear discriminant analysis (LDA), also known as fisher discriminant analysis (FDA [4]) or discriminant analysis feature selection (DAFE [5]). Another widely used and efficient supervised dimensionality reduction methods for hyperspectral image is nonparametric weighted feature extraction (NWFE [6]). In recent years, many extensions on those two methods have been proposed, such as generalized discriminant analysis (GDA [7]) using kernel trick, nonparametric discriminant analysis (NDA [8]), marginal fisher discriminant analysis (MFA [9]) and local fisher discriminant analysis (LFDA [10]), decision boundary feature extraction (DBFE [11]) and so on. Above supervised dimensionality reduction methods using supervision information, such as class labels or pairwise constraints, have been proved efficiently but also time-consuming in many applications. While unsupervised methods directly use unlabeled data to guide the process of dimensionality reduction. Principal components analysis (PCA [12]) is one of the most typical unsupervised dimensionality reduction methods, which acquires the main components of samples using a linear transformation. Other widely used unsupervised methods include independent components analysis (ICA [13]), multidimensionality scaling (MDS [14]), nonparametric matrix factorization (NMF [15]), kernel PCA (KPCA [16]) and other manifold learning and sparse representation based methods [17, 18]. Above unsupervised methods not used any supervision information usually receive imprecise results in hyperspectral image classification. Combining the advantages of both supervised methods and unsupervised methods, semi-supervised dimensionality reduction methods which use supervision information as in supervised methods and keep intrinsic structural information (such as global variance of data and local structural information) as in unsupervised methods have been proposed in recent years. Typical semi-supervised dimensionality reduction methods include semi-supervised probabilistic

PCA (S2PPCA [19]), classification constrained dimensionality reduction (CCDR [20]), constraint based semi-supervised dimensionality reduction framework (SSDR [21]) and so on.

To involve the problem of imprecise estimation of hyperspectral samples with similar spectral properties, many hyperspectral images classification methods considering both spectral and spatial information have been proposed to decrease speckle-like errors caused by most traditional *pixelwise* classifiers. Spectral and spatial information are combined by two main methods: graph-based techniques which build a regularization on the samples with similar spectral properties and fixed-window-based methods which use markov random fields (MRFs) and their extensions, such as SCSVM and SCSVMF [22] modifying the decision function of SVM and using the spatial information in the original space and the feature space respectively. Although MRF is a classical statistical method for modeling the spatial contextual information, the observed data is assumed conditional independence in the MRF-based classifiers. However, there are strong correlation between adjacent pixels for hyperspectral images. A new discriminative probabilistic model, i.e. conditional random field (CRF [23]) relaxes conditions for MRF. Recent studies show that CRF is efficently used in object recognition and image segmentation. Ping Zhong et al. [24] developed an efficient local method to train a CRF under piecewise training framework for hyperspectral image classification. Chi-Hoon Lee et al. [25] used CRFs and SVMs to segment brain tumors. Zuchuan Li et al. [26] applied SVM and CRF into hyperspectral images classification. Although recent works show that SVM and CRF based models work well on many applications, the computational cost of classification large data sets such as hyperspectral images is very high.

In this chapter, we propose a fast ELM and CRF based model to classify hyperspectral images. ELM [27] is a simple least square based learning algorithm for single-hidden layer feedforward neural network (SLFN). Recent studies show that ELM is a suitable, effective, and less time-consuming classifier with a wide range of applications. Therefore, we take ELM as a classifier, and CRF is used for smoothing the result of ELM. The remainder of this chapter is organized as follows: Sect. 2 introduces the used dimensionality reduction algorithms (i.e. SSDRpca and SSDRsp [28]) and the corresponding ELM classification methods. Section 3 descries the CRF smoothing of the result of ELM classification. Section 4 shows the experimental results in two real hyperspectral data sets: the Indian Pines 92AV3C and the Washington DC mall. Section 5 draws conclusions of this chapter.

2 Dimensionality Reduction and Classification

2.1 Semi-Supervised Dimensionality Reduction

To avoid the Hughes phenomenon often happened in classification for high-dimensional data directly, we take semi-supervised dimensionality reduction from state-of-the-art dimensionality reduction methods to reduct the dimensionality of hyperspectral data.

Shiguo Chen et al. [28] presented a semi-supervised dimensionality reduction framework, which contains a discrimination term (based on pairwise constraints) and a regularization term (to characterize some property of the original dataset). The goal is to maximize the global objective function which is made up of two terms as shows in Eq. 1, where J_D is the discrimination term and J_R is the regularization term, β is the parameter.

$$J = J_D + \beta \bullet J_R \tag{1}$$

Using PCA criterion as the regularization term, Eq. 1 can be written as Eq. 2, which is the global objective function of $SSDR_{pca}$ as shown in [21]. Therefore, $SSDR_{pca}$ can be seen as a special case of the semi-supervised dimensionality reduction framework.

$$\begin{aligned} J_{SSDR_{pca}}(w) = & \frac{1}{2n_C} \sum_{(i,j)\in C} (w^T x_i - w^T x_j)^2 - \frac{\alpha}{2n_M} \sum_{(i,j)\in M} (w^T x_i - w^T x_j)^2 \\ & + \frac{\beta}{2N^2} \sum_{i,j} (w^T x_i - w^T x_j)^2 \end{aligned} \tag{2}$$

Using the sparse representation as the regularization term, Eq. 1 can be written as Eq. 3, which is the global objective function of $SSDR_{sp}$. Where s_i is the sparse reconstructive weight vector for x_i.

$$\begin{aligned} J_{SSDR_{sp}}(w) = & \frac{1}{2n_C} \sum_{(i,j)\in C} (w^T x_i - w^T x_j)^2 - \frac{\alpha}{2n_M} \sum_{(i,j)\in M} (w^T x_i - w^T x_j)^2 \\ & + \beta \bullet \left[-\frac{1}{N} \sum_i \| w^T x_i - w^T X s_i \|^2 \right] \end{aligned} \tag{3}$$

2.2 ELM Classification

ELM is a simple learning algorithm originally proposed for single-hidden-layer feed-forward neural networks (SLFNs) and then extended to the generalized SLFNs [27]. ELM is based on the Moore-Penrose generalized inverse instead of tuning the hidden layer.

Given N arbitrary distinct samples (x_i, t_i), where input variables are $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in \mathbb{R}^n$ and target values are $t_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in \mathbb{R}^m$, the output of standard SLFNs with $\tilde{N}$ hidden nodes are mathematicly modeled as

$$f(x) = \sum_{i=1}^{\tilde{N}} \beta_i \bullet g_i(x), \tag{4}$$

where $\beta_i \in \mathbb{R}^m$ is the weight vector connecting the ith hidden node and the output nodes, $g_i(x)$ is the activation function. With some simple algebraic derivation, above equation can be written as:

$$H\beta = T, \tag{5}$$

where $H_{N\times\tilde{N}}$ is called the hidden layer output matrix of the SLFN, $\beta_{\tilde{N}\times m}$ is the output weight matrix, and $T_{N\times m}$ is the target matrix. After a series of mathematical derivation, the smallest norm least-squares solution of the above linear system is as follow:

$$\hat{\beta} = H^{\dagger}T, \tag{6}$$

where $H^{\dagger}$ is the Moore-Penrose inverse of H.

As shown in [27], ELM has the ability to approximate any target continuous function and classify any disjoint regions. In this chapter, we address ELM to the classification of hyperspectral images. The probability estimates of each pixel over all the class labels acquired by ELM is considered for the following optimization work.

3 CRF Smoothing

Recent studies [24] proved that there are strong correlations between neighboring spectral bands and spatial neighbors in both the observations and label image. Whereas an ordinary classifier predicts a label for a single sample without regard to the neighboring samples. In order to involve the problem of imprecise estimation of hyperspectral samples with similar spectral properties but come from different land cover caused by conventional *pixelwise* classification methods, we develop a new hyperspectral image classification method (called ELM-CRF) based on CRF to combine spectral and spatial information to deal with the problems mentioned earlier.

CRF is a discriminative framework globally conditioned on the observation X. According to Hammersley-clifford theorem, the conditional distribution over labels Y given the observations can be described as follows:

$$P(Y|X) = \frac{1}{Z}\prod_{c}\psi_c(Y_c, X), \tag{7}$$

where $\psi_c(Y_c, X)$ is the potential function of the clique c, and Z is a normalizing constant known as the partition function [23]. The corresponding Gibbs energy function defined on unary and pairwise cliques is given by

$$E(X) = \sum_{i \in V} \psi_i(x_i) + \sum_{(i,j) \in E} \psi_{ij}(x_i, x_j), \tag{8}$$

where V is the set of all hyperspectral image pixels, E is a set of the neighborhood of each pixels. As shown in Eq. 8, the first term known as the unary potential reflects the probability of the given pixel x_i labeled as y_i. The second term known as the pairwise potential represents interaction between labels of neighboring sites. Minimizing the energy function Eq. 8, we can get the optimized parameters of CRF.

In this chapter, the energy function of CRF used in hyperspectral images classification is defined as follows: the unary potential ψ_i of ELM-CRF is defined directly by the probability outputs provided by our ELM classifier for each image pixel, while the unary potential of our comparison method (marked as SVM-CRF) in our experiments is defined by the probability outputs of SVM as in [29]. The pairwise edge potentials of ELM-CRF and SVM-CRF have the form of a Potts model as in [29, 30]. Namely the pairwise edge potential is

$$\psi_{ij}(x_i, x_j) = exp\left(-\frac{\|x_i - x_j\|^2}{\beta}\right) \delta(x_i \neq x_j), \tag{9}$$

where β is the model parameter. In other words, the pairwise edge potentials depend on the consistency of the two labels of neighboring pixels. If neighboring pixels have similar spectral values, ψ_{ij} favors the same category label for them. If neighboring pixels have dissimilar spectral values, they might be assigned different category labels. Therefore, the second term in Eq. 8 helps for smoothing the results of labeling the hyperspectral pixels as shown in our experiments.

4 Experiments

Experiments were conducted on two real hyperspectral airborne images, Indian Pines 92AV3C and Washington DC Mall,[1] described in the following:

Indian Pines 92AV3C is a 220-chanel 20-m resolution image (145 × 145 pixels) of a vegetation area from Northern Indiana that was recorded by the AVIRIS sensor on June 12, 1992. 50.7 % pixels of the image with the ground truth were categorized into 16 classes. A three-band false color image and the spatial distribution of the ground truth image are shown in Fig. 1. This dataset was randomly partitioned into a set of 1297 training samples and 1298 testing samples (set 1), all the pixels of 16 classes constitute another testing set (set 2), which are detailed in Table 1.

Washington DC Mall is a 210-chanel 2-m hyperspectral airborne data (1280 × 307 pixels) collected in the 0.4–2.4 μm region of the visible and infrared specra over a Mall in Washington DC. Noisy bands due to water absorption were removed,

[1] https://engineering.purdue.edu/~biehl/MultiSpec/hyperspectral.html

Table 1 Data sets used in the experiments

Properties	Indian pines 92AV3C	Washington DC mall
Image size	145×145	1280×307
#bands	220	191
#classes	16	7
#training samples	1297	2094
#testing samples (set 1)	1298	3040
#testing samples (set 2)	10366	392960

Table 2 Classification accuracy (%) on the Indian pines 92AV3C

Test sets	Classifiers	PCA	NWFE	SSDRpca	SSDRsp
Set 1	SVM	**7.70**	**82.51**	80.70	79.98
	ELM	6.93	80.28	**84.43**	**83.37**
Set 2	SVM	3.67	77.73	75.14	74.16
	SVM-CRF	3.67	91.47	88.18	86.90
	ELM	**23.81**	75.16	80.82	79.08
	ELM-CRF	**23.81**	**92.24**	**94.72**	**94.43**

resulting in 191 chanels. Seven classes of interest are considerd. A three-band false color image is shown in Fig. 2. This dataset was randomly partitioned into a set of 2094 training samples and 3040 testing samples (set 1), all the pixels of the whole hyperspectral image constitute another testing set (set 2), which are detailed in Table 1.

In this experiments, four dimensionality reduction methods were used here: unsupervised (PCA [12]), supervised (NWFE [6]) and semi-supervised ($SSDR_{pca}$ and $SSDR_{sp}$, the same constraints and parameter selection mechanism used as in [28]). The classification results were compared with those given by: (i) a traditional SVM classifier; (ii) a traditional ELM classifier; (iii) a SVM-CRF classifier based on the Potts model and estimates of the class posterior probabilities computed by SVM; (iv) a ELM-CRF classifier based on the Potts model and estimates of the class posterior probabilities computed by ELM. On testing set 1 of two data sets, two pixelwise classifiers (SVM and ELM) are used in the experiments. While on the testing set 2 of two data sets, two smoothing methods (SVM-CRF and ELM-CRF) are used to compare with traditional pixelwise classifiers (SVM and ELM).

We use classification accuracy to evaluate the results of methods proposed above. Tables 2 and 3 show the results on two hyperspedtral image data sets, respectively. Figures 1 and 2 show the classification maps of different classification methods mentioned above.

From the results of Table 2, we can see that the classification results of semi-supervised methods on testing set 1 of Indian Pines 92AV3C are superior to the unsupervised and supervised methods as proved in [28]. The results of two smoothing methods (SVM-CRF and ELM-CRF) are superior to pixelwise classifiers (SVM and

Table 3 Classification accuracy (%) on the Washington DC mall, accuracy on the testing set 2 was computed only on the pixels with ground truth as shown in Fig. 2d

Test sets	Classifiers	PCA	NWFE	SSDRpca	SSDRsp
Set 1	SVM	7.11	95.07	91.13	91.42
	ELM	**18.55**	**96.55**	**96.47**	**95.95**
Set 2	SVM	5.01	72.32	69.40	69.56
	SVM-CRF	5.01	72.15	71.68	71.88
	ELM	**6.02**	**75.88**	71.53	72.45
	ELM-CRF	5.15	74.53	**72.8**	**73.71**

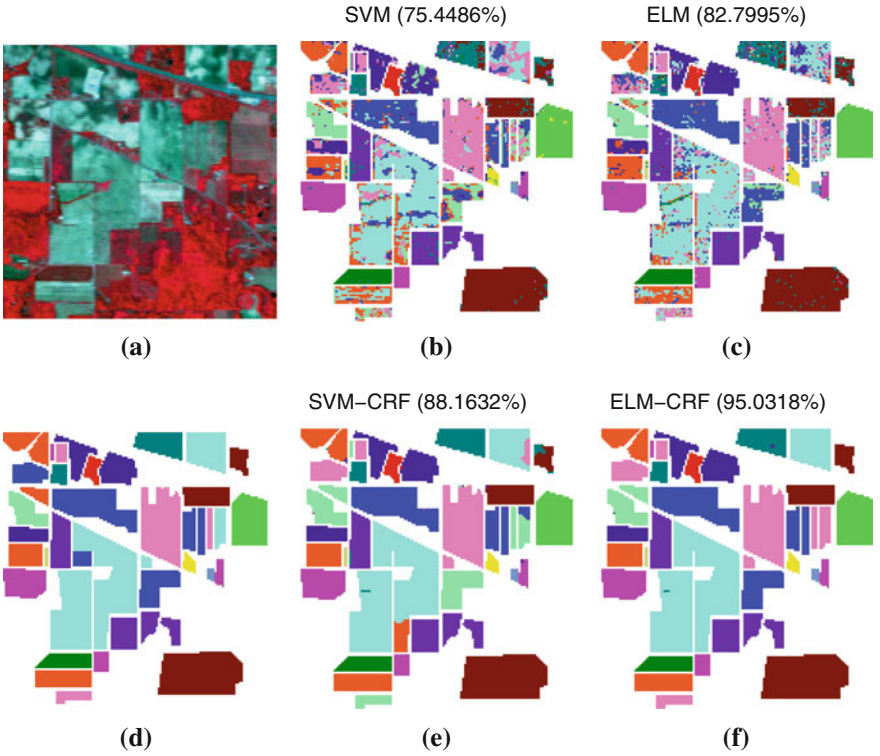

Fig. 1 Indian pines 92AV3C **a** Three-band color composite. **b** SVM pixelwise classification map. **c** ELM pixelwise classification map. **d** The ground truth. **e** SVM-CRF classification map. **f** ELM-CRF classification map

ELM). As shown in Table 3, the smoothing methods do not have significant effect on Washington DC Mall. One possible reason is that the accuracy was computed only on the pixels with ground truth, while the pixels with ground truth have a very small proportion in the whole Washington DC Mall hyperspectral image.

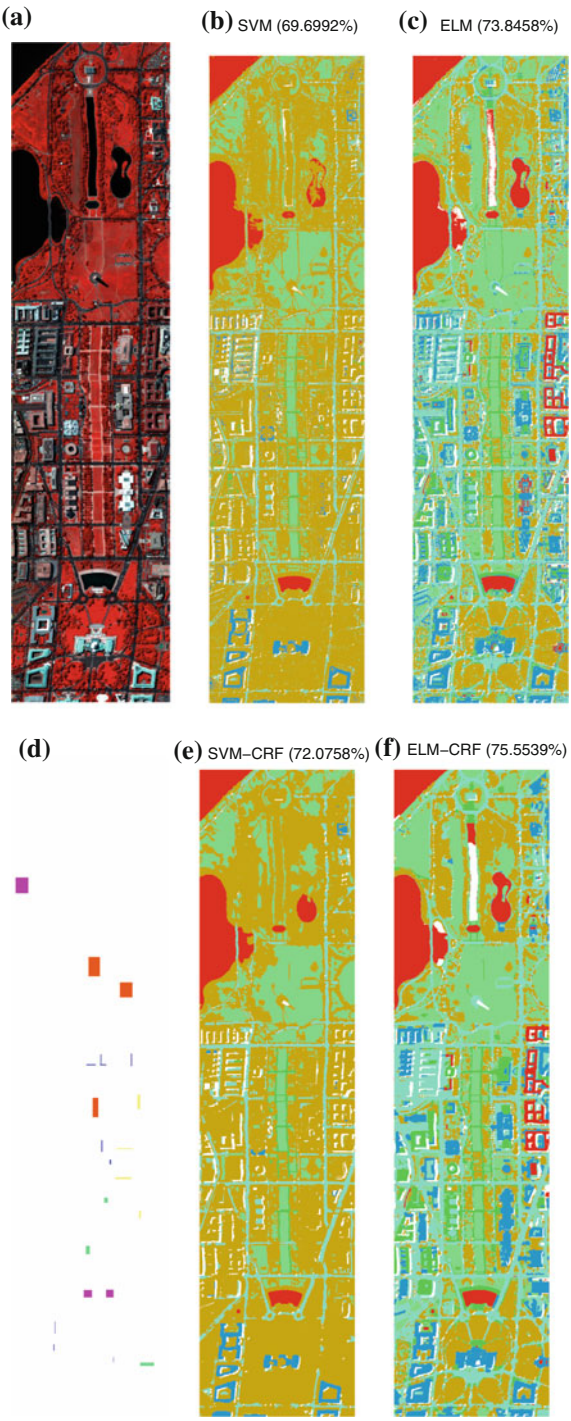

Fig. 2 Washington DC mall **a** Three-band color composite. **b** SVM pixelwise classification map. **c** ELM pixelwise classification map. **d** The ground truth. **e** SVM-CRF classification map. **f** ELM-CRF classification map. Accuracies on the classification maps was computed only on the pixels with ground truth as shown in Fig. 2d

Table 4 Computational cost (the unit of measurement is second) of different approaches on the Indian pines 92AV3C, where t_{dr} denotes the cost time of dimentionality reduction, the other symbols denote a similar meaning

Methods	t_{dr}	$t_{svmtrain}$	t_{svm_set1}	t_{svm_set2}	t_{crf_set2}
PCA	0.0463	10.8349	0.1910	1.1003	0.4741
NWFE	571.6478	59.8343	0.5750	5.1058	0.3573
SSDRpca	0.3974	10.9710	0.0965	0.7885	0.3537
SSDRsp	29.9099	15.1722	0.1481	1.0790	0.3452
Methods	t_{dr}	–	t_{elm_set1}	t_{elm_set2}	t_{crf_set2}
PCA	0.0549	–	0.3442	0.6256	0.2971
NWFE	570.4765	–	0.1964	0.5748	0.3384
SSDRpca	0.3477	–	0.3291	0.5417	0.4403
SSDRsp	0.3785	–	0.3179	0.3728	0.6145

Table 5 Computational cost (the unit of measurement is second) of different approaches on the Washington DC mall, where t_{dr} denotes the cost time of dimentionality reduction, the other symbols denote a similar meaning

Methods	t_{dr}	$t_{svmtrain}$	t_{svm_set1}	t_{svm_set2}	t_{crf_set2}
PCA	0.0542	12.1764	0.0478	2.9917	21.4838
NWFE	1858	43.5631	0.0003	0.0199	0.0432
SSDRpca	0.7885	11.538	0.0619	7.4123	28.9416
SSDRsp	44.6645	11.2901	0.0567	6.9673	27.8813
Methods	t_{dr}	–	t_{elm_set1}	t_{elm_set2}	t_{crf_set2}
PCA	0.0630	–	0.3244	9.8259	30.8901
NWFE	1886.6	–	0.0003	0.0128	0.0346
SSDRpca	0.8019	–	0.4531	10.1931	32.0403
SSDRsp	45.9748	–	0.2897	10.7319	33.8963

From Figs. 1 and 2, we can see that the speckle-like errors which appear in the $pixelwise$ classifiers are corrected by SVM-CRF and ELM-CRF. Here, the effect of CRF is similar to the expansion technology in image processing. Overall, the methods using both spectral and spatial contextual information are efficient in hyperspectral image classification. From Tables 2 and 3, we can also see that the results of ELM based classifiers are just about right with the results of SVM based classifiers, but the training time of SVM is more than ELM, which are detailed in Tables 4 and 5. That is because the parameters are random assighed in ELM, while SVM wastes most time on finding the optimal model parameters.

All the simulations for classification algorithms mentioned on our experiments are carried out in MATLAB R2011a environment running in a Dell PowerEdge R710 server with 16 Intel(R) Xeon(R) 2.93 GHZ CPUs and 32 GB memory.

5 Conclusions

This chapter has introduced a new spectral and spatial contextual information based classification method for hyperspectral images. The method is consistent of three steps: (i) dimensionality reduction using a semi-supervised dimentionality reduction framework; (ii) classification using ELM; (iii) smoothing the classification maps using CRF. Experimental results on two real experiments have demonstrated that the proposed method yields accurate classification maps within a short time compared to the *pixelwise* classifiers.

Acknowledgments We are grateful for financial support from the National Nature Science Foundation of China under Grant No. 61101202 and the National Technology Research and Development Program of China under Grant No. 2012AA01A510.

References

1. P.H. Hsu, Feature extraction of hyperspectral images using wavelet and matching pursuit. ISPRS J. photogramm. Remote Sens. **62**(2), 78–92 (2007)
2. D.A. Landgrebe, *Signal Theory Methods in Multispectral Remote Sensing*, vol. 29 (Wiley Interscience, Hoboken, 2005)
3. G. CampsValls, L. Bruzzone, Kernel-based methods for hyperspectral image classification. IEEE Trans. Geosci. Remote Sens. **43**(6), 1351–1362 (2005)
4. R.A. Fisher, The use of multiple measurements in taxonomic problems. Ann. Eugen. **7**(2), 179–188 (1936)
5. E.M. Mikhail, J.S. Bethel, J.C. McGlone, *Introduction to Modern Photogrammetry* (Wiley, New York, 2001)
6. B.C. Kuo, D.A. Landgrebe, Nonparametric weighted feature extraction for classification. IEEE Trans. Geosci. Remote Sens. **42**(5), 1096–1105 (2004)
7. G. Baudat, F. Anouar, Generalized discriminant analysis using a kernel approach. Neural Comput. **12**(10), 2385–2404 (2000)
8. K. Fukunaga, J.M. Mantock, Nonparametric discriminant analysis. IEEE Trans. Pattern Anal. Mach. Intell. **6**, 671–678 (1983)
9. S.C. Yan, D. Xu, B.Y. Zhang, H.J. Zhang, Q. Yang, S. Lin, Graph embedding and extensions: a general framework for dimensionality reduction. IEEE Trans. Pattern Anal. Mach. Intell. **29**(1), 40–51 (2007)
10. M. Sugiyama, Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. J. Mach. Learn. Res. **8**, 1027–1061 (2007)
11. C. Lee, D.A. Landgrebe, Feature extraction based on decision boundaries. IEEE Trans. Pattern Anal. Mach. Intell. **15**(4), 388–400 (1993)
12. H. Hotelling, Analysis of a complex of statistical variables into principal components. J. Educ. Psychol. **24**, 417–441 (1993)
13. J. Wang, C.I. Chang, Independent component analysis-based dimensionality reduction with applications in hyperspectral image analysis. IEEE Trans. Geosci. Remote Sens. **44**(6), 1586–1600 (2006)
14. L. Yang, Alignment of overlapping locally scaled patches for multidimensional scaling and dimensionality reduction. IEEE Trans. Pattern Anal. Mach. Intell. **30**(3), 438–450 (2008)
15. D. Seung, L. Lee, Algorithms for nonnegative matrix factorization. Adv. Neural Inf. Process. Syst. **13**, 556–562 (2001)

16. B. Schölkopf, A. Smola, K.R. Müller, Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput. **10**(5), 1299–1319 (1998)
17. J.B. Tenenbaum, V. De Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500), 2319–2323 (2000)
18. L.S. Qiao, S.C. Chen, X.Y. Tan, Sparsity preserving projections with applications to face recognition. Pattern Recognit. **43**(1), 331–341 (2010)
19. S.P. Yu, K. Yu, V. Tresp, H.P. Kriegel, M.R. Wu, Supervised probabilistic principal component analysis, in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, New York, 2006), pp. 464–473
20. J. A. Costa, A.O. Hero III, Classification constrained dimensionality reduction, in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005 (ICASSP'05)* vol. 5 (IEEE, 2005), pp. v-1077
21. D.Q. Zhang, Z.H. Zhou, S.C. Chen, Semi-supervised dimensionality reduction, in *Proceedings of the 7th SIAM International Conference on Data Mining* (2007), pp. 629–634
22. C.H. Li, B.C. Kuo, C.T. Lin, C.S. Huang, A spatial-contextual support vector machine for remotely sensed image classification. IEEE Trans. Geosci. Remote Sens. **50**(3), 784–799 (2012)
23. J. Lafferty, A. McCallum, F.C. Pereira, Conditional random fields: probabilistic models for segmenting and labeling sequence data, in *Proceedings of the Eighteenth International Conference on Machine Learning* (2001), pp. 282–289
24. P. Zhong, R.S. Wang, Learning conditional random fields for classification of hyperspectral images. IEEE Trans. Image Process. **19**(7), 1890–1907 (2010)
25. C.H. Lee, M. Schmidt, A. Murtha, A. Bistritz, J. Sander, R. Greiner, Segmenting brain tumors with conditional random fields and support vector machines, in *Computer Vision for Biomedical Image Applications* (Springer, 2005), pp. 469–478
26. Z.C. Li, J.W. Ma, R. Zhang, L.W. Li, Classifying hyperspectral data using support vector machine conditional random field. Geomat. Inf. Sci. Wuhan Univ. **36**(3), 306–310 (2011)
27. G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications. Neurocomputing **70**(1), 489–501 (2006)
28. S.G. Chen, D.Q. Zhang, Semisupervised dimensionality reduction with pairwise constraints for hyperspectral image classification. IEEE Geosci. Remote Sens. Lett. **8**(2), 369–373 (2011)
29. B. Fulkerson, A. Vedaldi, S. Soatto, Class segmentation and object localization with superpixel neighborhoods, in *IEEE 12th International Conference on Computer Vision, 2009* (IEEE, 2009), pp. 670–677
30. J. Shotton, J. Winn, C. Rother, A. Criminisi, Textonboost: joint appearance, shape and context modeling for multi-class object recognition and segmentation, in *Computer Vision-ECCV 2006* (Springer, 2006), pp. 1–15

ELM Predicting Trust from Reputation in a Social Network of Reviewers

J. David Nuñez-Gonzalez and Manuel Graña

Abstract Trust is a central concept in distributed systems, such as Ad Hoc Networks, Social Networks and Recommender Systems. Trust has a predictive component, it is a measure of the certainty that an agent has on the output from other agent. Hence, Trust is a key component of distributed decision making processes. It can be built from reputation, meaning the observation of the Trust values from other agents (the trusters) on the target agent (the trustee). In this chapter we take the point of view of predicting the Trust value on the basis of the reputation information that the agent may collect. When Trust values are categorical, or binary, the problem becomes a classification problem that can be tackled by Extreme Learning Machines (ELM). We perform experimental assessment of the value of ELM for this task on a benchmark database obtained from a real life recommender system.

Keywords ELM · Trust computation · Recommender systems

1 Introduction

Trust is a pervasive concern in human and computational interactions [1, 2]. We must trust the services we receive and we rely on for our work and daily life, ranging from the use of cars and transports to the use of internet services, or the interaction between computational agents performing searches or other delegated tasks. Trust in automation [3] has been identified as a major concern in the development of human centered computing, proposing active evaluation of trust strategies to correct unjustified trust or mistrust, and to assess their consequences. In general terms the

J. D. Nuñez-Gonzalez (✉) · M. Graña (✉)
Computational Intelligence Group, University of the Basque Country, UPV/EHU, Spain
e-mail: jdnunez001@ikasle.ehu.es

M. Graña
e-mail: ccpgrrom@sc.ehu.es

F. Sun et al. (eds.), *Extreme Learning Machines 2013: Algorithms and Applications*, Adaptation, Learning, and Optimization 16, DOI: 10.1007/978-3-319-04741-6_13,

trusters face some degree of risk when they decide to accept the outcome of the trustee. Trust management [4, 5] is related with the prediction of the expected risk or the affordable level of trust on the basis of all available information. When the source of information is the opinion of other agents, the witnesses, the system is based on Reputation.

In the field of distributed systems, such as Ad Hoc communication networks, Social Networks, Online Review Systems and Recommender Systems, the issue of managing Trust is critical for the function of the system. Recommender Systems are common in e-commerce for making personalized marketing. On the other hand, Online Review Systems (ORS) allow users to provide reviews of products and thus become a user-oriented Recommender System. To help the user to navigate the reviews, the ORS provides the possibility to state trust scores on the reviews, so that reviewers with more positive trust scores will merit more attention. The issue, then, is how the observed trust scores given by other users may influence the user, and may serve to predict his own trust value.

Specifically, in this chapter we are concerned with the use of classifiers trained with Extreme Learning Machines (ELM) to perform the prediction of the Trust on the basis of the reputation of the trustee obtained from witness agents that have common trust relations with the truster and the trustee. Reputation is modeled as a feature vector composed of the trust values of the witness agents on the trustee. We apply the approach on a trust database extracted from an ORS web service that has been used for the study of Trust metrics [6] and outlier controversial reviews [7]. We perform classification experiments assessing the generalization power of ELM compared with other state-of-the-art classifier training algorithms.

Contents of the chapter This chapter is organized as follows: Sect. 2 reviews some ideas about Trust in Social Networks. Section 3 provides a review of ELM basics. Section 4 describes the experimental database and the reputation feature extraction. Section 5 gives the experimental results. Finally, Sect. 6 gives some conclusions and directions of research work for the future.

2 Trust

Some philosophical definitions of Trust are:

- "the degree of subjective belief about the behaviors of (information from) a particular entity" [8]
- "the quantified belief by a truster with respect to the competence, honesty, security, and dependability of a trustee within a specified context" [1]
- "a particular level of the subjective probability with which an agent will perform a particular action, both before [we] can monitor such action (or independently of his capacity of ever be able to monitor it) and in a context in which it affects [our] own action" [9].

The literature contains the recognition of several properties of Trust that may be useful to understand or develop trust-based systems. Goldbeck et al. [10] identifies transitivity, asymmetry properties meaning that Trust relations tend to be transitive and that there is no guarantee that Trust is reciprocated, and personalization properties. Cho et al. [11] state that Trust is subjective, changes in time (dynamic) and is context-dependent. Some authors point that Trust is reflexive, an agent trust in itself always, non antisymmetric, meaning that mutual trust does not imply identity, and that Trust decays with time and (physical or virtual) distance [9, 12].

Increasing concerns about Trust and ethics in computing are appearing in the literature. From the philosophical reflections on Trust in Cyberspace [13] and automated systems [3] to the more precise proposals of ethical government in the field of robotics [14] motivated by their military and medical uses. The proposal includes models of moral emotions to enforce ethical robotic behavior. Distributed systems introduce a new dimension in Trust related issues: Trust becomes a factor in the computation/communication system. In Peer-to-Peer systems, nodes need to reason and establish Trust on their communicating nodes to protect themselves against attacks [15, 16]. Mobile Ad Hoc Networks (MANET) allow to create dynamical communication paths arising from temporary relations between nodes. Computing Trust is a critical capability in MANETs involving establishing, updating and revocating Trust [11, 17]. The trust management problem generalizes to wireless communications [18] encompassing MANETs, wireless sensor networks and cognitive radio. Current approaches to Trust management are based on fuzzy or probabilistic reasoning on the information available, however some machine learning techniques have been proposed, i.e. to detect attacks coming from malicious nodes for MANETs [19].

3 Extreme Learning Machines

Extreme Learning Machine (ELM) [20, 21] is a simple learning algorithm for Single-Hidden Layer Feedforward Neural network (SLFN). This method is based on the Moore-Penrose generalized inverse providing the minimum Least-Squares solution of general linear systems.

3.1 Basic ELM

For N arbitrary distinct samples $(\mathbf{x_i}, t_i)$, where input variables are $\mathbf{x_i} = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in \mathbb{R}^n$ and target values are $\mathbf{t_i} = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in \mathbb{R}^m$. The training of a standard SLFN with N hidden neurons and activation function $g(x)$ is mathematically modeled as solving the following equation to estimate the value of the SLFN parameters:

$$\sum_{i=1}^{hn} \beta_\mathbf{i} \cdot g(\mathbf{w_i} \cdot \mathbf{x_j} + b_i) = \mathbf{t_j}, \quad j = 1, \ldots, N. \tag{1}$$

where $\mathbf{w_i} = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ is the weight vector connecting the i-th hidden neuron and the input neurons, $\beta_\mathbf{i} = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ is the weight vector connecting the $\mathbf{i}$-th hidden neuron and the output neurons, and b_i is the threshold of the ith hidden neuron. $\mathbf{w_i} \cdot \mathbf{x_j}$ denotes the inner product of $\mathbf{w_i}$ and $\mathbf{x_j}$ and hn is the number of hidden neurons. The activation function can be the identity for the so-called linear kernel approaches, sigmoid for the Multilayer Perceptron approaches, or Gaussian for Radial Basis Function approaches [21].

The Eq. (1) can be written in matrix form as:

$$\mathbf{H}\beta = \mathbf{T}, \tag{2}$$

where $\mathbf{H}$, of size $N \times hn$, is the output matrix resulting of the SLFN hidden layer activated by the input samples, β is the output weight matrix of size $hn \times m$, and $\mathbf{T}$ is the target matrix with size $N \times m$. Training of SLFN is accomplished by computing the least-squares solution $\hat{\beta}$ of the linear system $\mathbf{H}\beta = \mathbf{T}$, given by $\hat{\beta} = \mathbf{H}^\dagger\mathbf{T}$, where $\mathbf{H}^\dagger$ is the Moore-Penrose inverse of $\mathbf{H}$.

4 Experimental Database and Reputation Features

The original database The Epinions site[1] is a social site where user provide reviews of products of any kind, from music to perfumes or construction hardware. These reviews are the base for the establishment of trust relations between users. Trust is a binary variable taking values in $\{-1, 1\}$: a user can choose to trust (1) another or not (-1). Negative trust values are not published in the web service, but the anonymized dataset provided for experimentation, which is available to the public,[2] contains also negative Trust values. This dataset has 841,372 data samples. Each sample is a triplet composed of two user indexing numbers (no personal data of any form is included) and the binary Trust value of the first user on the second user. Therefore, Trust relations define a directed graph, with weighted edges. Used database is unbalanced: 85.3 % of instances show positive trust (717,667 triplets), versus 14.7 % of negative trust instances (123,705 triplets). This data base has been used previously to perform computational experiments of Trust models [6, 7, 22, 23].

Reputation features From the original database of triplets, we build several databases of Reputation features, consisting on the observation of the Trust values of related users. Each database is made of samples composed of a feature vector of specific dimension and the desired trust value to be predicted. Construction of the database is as follows: For each triplet (A, B, t_{AB}) we construct a list of witness users $L_{AB} = \{C \,|(C, A, t_{CA}) \in \mathcal{D} \wedge (C, B, t_{CB}) \in \mathcal{D}\}$, where $\mathcal{D}$ denotes the original database of triplets. Given a feature vector dimension, i.e. d, we discard the triplet if $|L_{AB}| < d$. If $|L_{AB}| > d$, we perform a random selection of d witness

[1] http://www.epinions.com/

[2] http://www.trustlet.org/wiki/Extended_Epinions_dataset

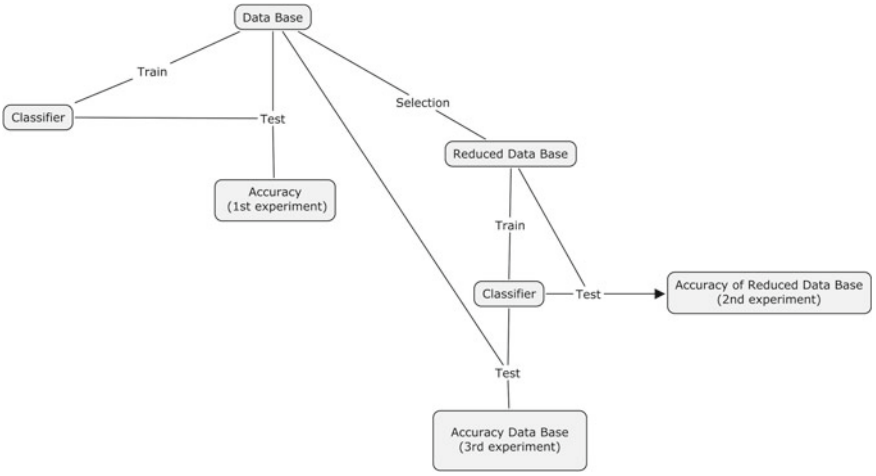

Fig. 1 Pipeline of the experimental works

nodes C obtaining L^*_{AB} such that $\left|L^*_{AB}\right| = d$. The input/output pair (X, Y) in the reputation feature database corresponding to triplet (A, B, t_{AB}) is constructed such as $X = \left\{t_{C,B} \,\middle|\, C \in L^*_{AB}\right\}$ and $Y = t_{AB}$. For $d = 10$, the reputation feature database has 735,757 samples with 14.86 % of class "−1" and 85.14 % of class "1". This reputation feature database will be published at the group's website[3] for third party assessment of results.

5 Experimental Results

Experimental pipeline The computational experiments follow the scheme of Fig. 1. The complete reputation feature database is used (in a 10-fold cross-validation) to build classifiers and test them, obtaining the first experiment accuracy results. In another pipeline path, we select a 10 % of the database samples, ensuring that the classes are well balanced. On this reduced database we perform a 10-fold cross-validation, obtaining the results of the second experiment. Finally, each of the classifiers constructed in the second experiment is further tested over the entire reputation feature database, obtaining the generalization results of the third experiment. The difference between the results of the third and the first experiment are representative of the generalization robustness of the approach considered. The competing classification algorithms are obtained from Weka.[4] When comparing computational times, it is convenient to keep in mind that ELM is implemented in Matlab, while Weka is implemented in Java.

[3] http://www.ehu.es/ccwintco/index.php/GIC-experimental-databases

[4] http://www.cs.waikato.ac.nz/ml/weka/

Table 1 Average accuracy of the **first** experiment 10-fold cross-validation and computation time in seconds

	3 features		5 features		7 features		10 features	
	% acc.	Time (s)	% acc.	Time (s)	% acc.	Time (s)	% acc.	Time (s)
AdaBoost	90	61	90	72	91	97	92	162
Spegasos function	89	146	91	152	92	168	92	205
Bayesian log. regr.	90	22	91	43	91	84	92	107
Logistic functions	90	63	91	73	92	87	92	93
Bagging	90	47	91	64	92	73	92	564
Decision table	90	39	91	90	92	69	92	135
Decision tree	90	72	91	78	92	82	92	210
ELM (20 hu)	90	1	91	1	92	2	92	2
ELM (50 hu)	90	5	91	5	92	6	**93**	6
ELM (70 hu)	91	5	91	5	92	6	**93**	6

Increasing reputation feature vector dimensions

Table 2 Average accuracy of the **second** experiment 10-fold cross-validation and computation time in seconds

	3 features		5 features		7 features		10 features	
	% acc.	Time (s)	% acc.	Time (s)	% acc.	Time (s)	% acc.	Time (s)
AdaBoost	82	57	82	54	83	59	84	74
Spegasos function	86	39	86	41	86	45	89	49
Bayesian log. regr.	82	43	87	44	87	49	89	56
Logistic functions	82	38	87	42	87	47	88	59
Bagging	86	42	86	44	88	51	89	55
Decision table	86	41	88	45	89	53	90	56
Decision tree	86	58	88	62	89	61	90	110
ELM (20 hu)	86	1	89	1	89	2	90	2
ELM (70 hu)	86	5	89	5	89	6	**91**	6

Increasing reputation feature vector dimensions

Table 3 Average accuracy of the **third** experiment 10-fold cross-validation and computation time in seconds

	3 features		5 features		7 features		10 features	
	% acc.	Time (s)	% acc.	Time (s)	% acc.	Time (s)	% acc.	Time (s)
AdaBoost	75	109	77	107	78	112	79	168
Spegasos function	75	143	77	167	79	177	80	237
Bayesian log. regr.	75	27	78	42	79	85	80	57
Logistic functions	75	56	78	63	79	78	80	84
Bagging	76	54	78	79	79	164	80	255
Decision table	76	39	78	61	79	83	80	98
Decision tree	76	61	78	62	79	69	80	116
ELM (50 hu)	76	5	78	5	79	6	**81**	6

Increasing reputation feature vector dimensions

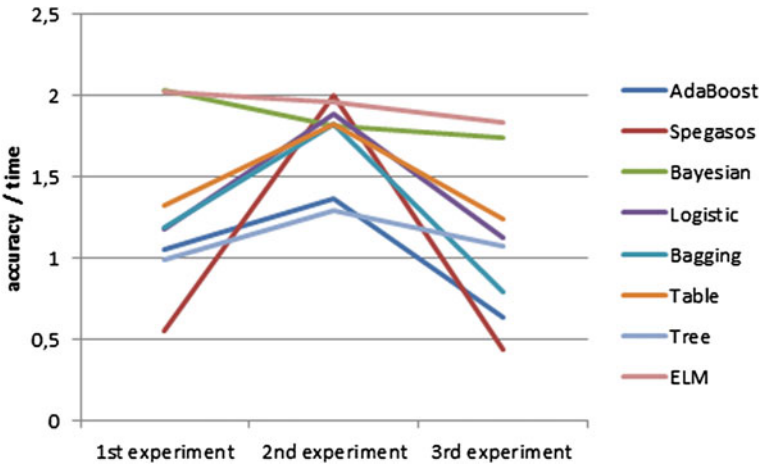

Fig. 2 Plot of the accuracy/time ration for the tested classifiers on the three experiments

First experiment The results of the 10-fold cross-validation on the whole database are given in Table 1. Overall the accuracy of the classifiers increases with the feature vector dimension, as well as the computational time measured in seconds. Best accuracy results are obtained by the ELM with 50 and 70 hidden units. Results are relatively similar in all classifiers, which can be interpreted as an indication of the discriminant power of the feature vectors.

Second experiment The results of the 10-fold cross-validation over the reduced feature database are given in Table 2. Again the best results are achieved by the ELM algorithm. We observe an expected reduction in computational time, and an unexpected reduction in accuracy that may be due to the sampling process. ELM are the most resilient architectures in this regard, with less accuracy loss. Results in this experiment are representative of the validation of the reduced database training.

Third experiment The results of the third experiment, consisting on applying the classifiers trained on each of the 10-folds on the reduced database to the entire database, are given in Table 3. The loss in accuracy when comparing with Table 1 reflect the effect of generalization from the reduced database to the entire database. Again best results are provided by ELM with lower computational cost.

Accuracy versus time Figure 2 shows a plot of the ratio Accuracy/time for the experiments with feature dimension 10, intended to highlight the trade off between accuracy and time requirements. The greater this ratio, the better the algorithm. ELM appears on top of all algorithms in this plot. Second best is the logistic regression. Other algorithms show some improvement in the second experiment due to the well balanced sample, but have big drops in the last experiment. Worst results are obtained by the decision tree and the adaboost.

6 Conclusions

We have introduced a Trust prediction system based on reputation features obtained as the trust values of witness agents. The system has been demonstrated over a benchmark trust database extracted the trust assessment of an Online Review System. The computational experiments have shown that the ELM achieves the best accuracies, and accuracy versus computational time ratio. Also, we have found that the loss of accuracy when generalizing from a small database to the complete database is smaller in ELM.

Further work will be carried out within the SandS project where a social network of household appliances share knowledge about the use of appliances through the exchange of recipes of use. The system is intended to provide emergent social intelligence from the social interactions. Users can ask for a recipe giving a task (described in natural language). Recipes will be generated by system and users. At the end, users can give a feedback about the obtained recipe. Trust values would serve to moderate the influence of the users regarding the composition of new recipes from the past recommendations. Therefore, a trust prediction system similar to the one presented here would be of value for the SandS system.

Acknowledgments This research has been partially funded by EU through SandS project, grant agreement No. 317947.

References

1. B. Bhargava, L. Lilien, A. Rosenthal, M. Winslett, M. Sloman, T.S. Dillon, E. Chang, F.K. Hussain, W. Nejdl, D. Olmedilla, V. Kashyap, The pudding of trust. IEEE Intell. Syst. **19**(5), 74–88 (2004)
2. N. Shadbolt, A matter of trust. IEEE Intell. Syst. **17**(1), 2–3 (2002)
3. R.R. Hoffman, M. Johnson, J.M. Bradshaw, A. Underbrink, Trust in automation. IEEE Intell. Syst. **28**(1), 84–88 (2013)
4. M.S. Lund, B. Solhaug, K. Stlen, Evolution in relation to risk and trust management. Computer **43**(5), 49–55 (2010)
5. J. Ma, M.A. Orgun, Trust management and trust theory revision. IEEE Trans. Syst. Man Cybern. Part A Syst. Hum. **36**(3), 451–460 (2006)
6. P. Massa, P. Avesani, Trust metrics in recommender systems, in *Computing with Social Trust*, ed. by J. Golbeck (Springer, Berlin, 2009), pp. 259–285
7. P. Victor, C. Cornelis, M.D. Cock, A.M. Teredesai, Trust- and distrust-based recommendations for controversial reviews. IEEE Intell. Syst. **26**(1), 48–55 (2011)
8. K.S. Cook (ed.), *Trust in Society. Russell Sage Foundation Series on Trust*, vol. 2. (Russell Sage Foundation, New York, 2003)
9. S.I. Ahamed, M.M. Haque, Md.E. Hoque, F. Rahman, N. Talukder, Design, analysis, and deployment of omnipresent formal trust model (FTM) with trust bootstrapping for pervasive environments. J. Syst. Softw. **83**(2), 253–270 (2010) (Comput. Softw. Appl.)
10. J. Golbeck, Computing with trust: definition, properties, and algorithms, in *Securecomm and Workshops, 2006*, pp. 1–7, 28 Aug 2006–1 Sept 2006
11. J.-H. Cho, A. Swami, I.-R. Chen, A survey on trust management for mobile ad hoc networks. IEEE Commun. Surv. Tutorials **13**(4), 562–583 (2011)

12. X. Gai, Y. Li, Y. Chen, C. Shen, Formal definitions for trust in trusted computing, in *2010 7th International Conference on Ubiquitous Intelligence Computing and 7th International Conference on Autonomic Trusted Computing (UIC/ATC)*, pp. 305–310 (2010)
13. R.R. Hoffman, J.D. Lee, D.D. Woods, N. Shadbolt, J. Miller, J.M. Bradshaw. The dynamics of trust in cyberdomains. IEEE Intell. Syst. **24**(6), 5–11 (2009)
14. R.C. Arkin, P. Ulam, A.R. Wagner, Moral decision making in autonomous systems: enforcement, moral emotions, dignity, trust, and deception. Proc. IEEE **100**(3), 571–589 (2012)
15. A.B. Can, B. Bhargava. Sort: a self-organizing trust model for peer-to-peer systems. IEEE Trans. Dependable Secure Comput. **10**(1), 14–27 (2013)
16. X. Li, F. Zhou, X. Yang, Scalable feedback aggregating (SFA) overlay for large-scale P2P trust management. IEEE Trans. Parallel Distrib. Syst. **23**(10), 1944–1957 (2012)
17. M. Omar, Y. Challal, A. Bouabdallah, Certification-based trust models in mobile ad hoc networks: a survey and taxonomy. J. Network Comput. Appl. **35**(1), 268–286 (2012)
18. Y. Han, Z. Shen, C. Miao, C. Leung, D. Niyato. A survey of trust and reputation management systems in wireless communications. Proc. IEEE **98**(10), 1755–1772 (2010)
19. R. Akbani, T. Korkmaz, G.V. Raju, Emltrust: an enhanced machine learning based reputation system for MANETs. Ad Hoc Networks **10**(3), 435–457 (2012)
20. G.-B. Huang, Q.-Y. Zhu, C.-K. Siew. Extreme learning machine: a new learning scheme of feedforward neural networks, in *IEEE International Conference on Neural Networks—Conference Proceedings*, vol. 2, pp. 985–990 (2004). (Cited By (since 1996), 113)
21. G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications. Neurocomputing **70**, 489–501 (2006)
22. P. Massa, P. Avesani, Controversial users demand local trust metrics: an experimental study on epinions.com community, in *Proceedings of the National Conference on Artificial Intelligence*, vol. 1, pp. 121–126 (2005). (Cited By (since 1996), 36)
23. P. Massa, P. Avesani, Trust metrics on controversial users: balancing between tyranny of the majority and echo chambers. Int. J. Semant. Web Inf. Syst. **3**(1), 39–64 (2007). (Cited By (since 1996), 26)

Indoor Location Estimation Based on Local Magnetic Field via Hybrid Learning

Yansha Guo, Yiqiang Chen and Junfa Liu

Abstract In this chapter, the magnetic field samples over more than 2 months inside an office building presents a finding that there exist the relative stable measurements for a single location and the relative obvious difference in most of locations. Under this phenomenon, a hybrid learning method based on the local magnetic field measurements is proposed. (1) Kalman filter is firstly utilized to smooth the initial samples in order to obtain the stable data. (2) Classification programs by Extreme learning machine (ELM) is introduced to model the relationship between the measurements and physical locations, and then four potential positions can be chosen for a special measurement. (3) The optimal location is finally confirmed in view of those four selections by using K-nearest neighbor (KNN) algorithm. A series of experiments and comparisons with other five methods were implemented to validate the feasibility and superiority of this technique for improving the positioning accuracy.

Keywords Magnetic field · Location estimation · Hybrid learning · Kalman filter · Extreme learning machine (ELM) · K-nearest neighbor (KNN)

Y. Guo (✉)
School of Information Technology Engineering, Tianjin University of Technology and Education, Tianjin 300222, China
e-mail: guoyansha@ict.ac.cn

Y. Guo · Y. Chen · J. Liu
Institute of Computing Technology, Chinese Academy of Sciences, Beijing 10090, China
e-mail: yqchen@ict.ac.cn

J. Liu
e-mail: liujunfa@ict.ac.cn

Y. Guo
Taicang science and Technology Incubator Park, Taicang 215400, China
e-mail: guoyansha@ict.ac.cn

F. Sun et al. (eds.), *Extreme Learning Machines 2013: Algorithms and Applications*, Adaptation, Learning, and Optimization 16, DOI: 10.1007/978-3-319-04741-6_14,

1 Introduction

Indoor location estimation has received considerable attention with the gradual development and maturation of wireless communication technology. As a major element in the particular applications, it can be used to provide the navigation aids for the disabled, help a user conveniently obtain the specific merchandise information from shopping malls, quickly guide rescuers to rescue the trapped people in trouble, provide the exhibition information for visitors in time in museum. First and last, indoor positioning in all walks of life are directly or indirectly affecting people's works and lives.

Recent years, some systems on the grounds of signal propagation and fingerprint algorithm had been exploited for location estimation. **Ultrasonic** technology accomplished a location estimation through measuring the time-of-flight of transmitting and receiving ultrasonic signals so that obtain the distances between move terminal and reference nodes (well-known points); This way for indoor positioning is greatly influenced by multipath effects and non-line-of–sight (NLOS) propagation although the overall positioning accuracy of it is higher [1, 2]. Some scholars constructed a positioning system in view of the **infrared** emitters (e.g. three emitters) mounted in fixed known sites, and then the angle differences, which were directly used to determine a target location, between any two emitters can be measured by an incident angle sensor [3, 4]; Whereas it is limited since infrared is only suitable for short-distance communication and easily disturbed by the lights. Also, **Bluetooth** positioning using received signal strength (RSS) and triangulation methods was proposed, this technique established an mathematical model to analyze the relationship between RSS, which were gotten by utilizing a feature in new Bluetooth standard, and the distances, which were calculated by some methods (e.g. Least Square Estimation, Three-border and Centroid Method), based on triangulation algorithm between any two Bluetooth devices [5]; In order to improve the positioning accuracy, Liang Chen et al. combined the information from RSS measurements and prior motion model based on Bluetooth access points (APs), and speed detection was either brought to calibrate the location estimation [6]; The superiority of this technology is that Bluetooth module can be conveniently embedded into other devices, but the stability is poor because it might be influenced by noise and complicated spatial environment. **Radio frequency identification** (RFID) tagged objects is a common feature to be employed for realizing indoor positioning; Some algorithms designed for station estimation have been presented by using mobile RFID readers and landmarks which were active or passive tags with the known locations [7–10]; This technique has no communication ability and need to combine with other technologies although it can transmit longer distance and has stronger penetrability. Currently, **Wi-Fi**, which owns some potential features such as wide covering range, fast transmission speed and high reliability, as a new wireless communication means has been broadly applied in indoor positioning; Firstly, fingerprint database or map are constructed through collecting RSS values from different Wi-Fi APs in environments; And then machine learning methods are adopted to model the relation between labeled locations and

RSS map; The target station can be finally concluded by mapping the real-time RSS records with them in the database based on the training model [11–14].

As described above, each technology with its advantages and disadvantages had been used in different occasions. Whereas, one thing in common of them is that a local radio network or some additional devices are necessary to be prearranged and deployed for emitting and receiving signals, which subsequently were employed to estimate the locations. Thus, piles of work and big-cost are inevitable. Whether there is one or some resources with minimum spending that can be applied in positioning increasingly becomes an exploratory problem facing the scholars.

Magnetic field as a universal natural resource in earth started to be noted. Janne Haverinen et al. tried a series of location researches and experiments using particle filters and Monte Carlo Localization (MCL) in the case of preconditions such as each building having its unique static magnetic field and the anomalies of magnetic field in local area having sufficient variability. The effect in indoor positioning especially in guiding the robot was prominent [15, 16]. Evidence of the magnetic field data in an office building (working area in Institute of Computing Technology Chinese Academy of Science) showed that the local variability was relative stable for longer periods, i.e. amplitude range of measurements (X, Y, Z) in different times for a location was relative smaller, but the phenomenon that the observations of one or more locations, where may be neighboring or far from each other, were similar also existed and directly confused the location identifications.

In this article, the magnetic field measurements of the working area in Institute of Computing Technology Chinese Academy of Sciences from 9 to 21 o'clock of 46 days over 2 months are gathered here to experiment. And then Kalman filter, ELM classification and KNN methods are introduced in different stages, respectively. Section 2 detailedly describes the basic principles and processes of three methods. Experiments and the corresponding results based on local magnetic field are implemented in Sect. 3. Section 4 summarizes the whole work and prospects several potential research points.

2 Methodologies

2.1 Kalman Filter

Filtering as a signal processing and conversion course can be achieved by hardware or realized by software for removing or weakening the unwanted components and enhancing the needed components [17]. A good filtering algorithm should drop the noise from signals (e.g. electromagnetic signals) while preserving the useful information.

Kalman filter, which generally used in a linear system, can provide state estimators with the minimum variance of estimation error. During estimating a system state based on the measurements of concrete position, two requirements, the expected

estimators are came anywhere near the true state and an estimator resulting in the smallest possible error variance can be obtained, become the criteria that the calculated estimators should satisfy [17]. Kalman filter implements the state estimations mainly by executing a recursive procedure that predicts the current state only based on the last estimation (so need not to track and save the historical data) and optimally updates the predicted estimator based on the real observations in order to acquire a new estimator with higher accuracy. Accordingly, how to predict and update the estimators constitutes three stages of this filtering method.

Step 1: State prediction. This process is responsible for gaining the priori estimator and error covariance of the current state. Equations (1) and (2) are introduced here to estimate the current state $X(k|k-1)(X \in R^n)$ and corresponding covariance $P(k|k-1)$ on the basis of the previous best estimator, $X(k-1|k-1)$ and $P(k-1|k-1)$, of a special system, where k is assumed the present time and $k-1$ means the last moment.

$$X(k|k-1) = A \times X(k-1|k-1) + B \times U(k-1) \tag{1}$$

$$P(k|k-1) = A \times P(k-1|k-1) \times A' + Q \tag{2}$$

$U(k-1) \in R^l$, which may be equal to 0 if there is no control variable, denotes an optional control variable in $k-1$ time and Q expresses the process noise covariance. A as a driving function or process noise is a $n \times n$ matrix that relates the state from $k-1$ to k time, and B is a $n \times l$ matrix that connects $U(k-1)$ to the state. In this chapter, these parameters are set as constants [18].

Step 2: Measurement update. This process incorporates the priori prediction into the real observations to calculate optimal posteriori estimation $X(k|k)$ for the current state. In this stage, the first task is to calculate Kalman gain $K(k)$ by Eq. (3). And then, the posteriori estimator and corresponding covariance can be generated by fusing Kalman gain $K(k)$, real measurement $Z(k) \in R^m$, priori estimator and covariance in k time based on Eqs. (4) and (5), respectively.

$$K(k) = P(k|k-1) \times H'/(H \times P(k|k-1) \times H' + R) \tag{3}$$

$$X(k|k) = X(k|k-1) + K(k) \times (Z(k) - H \times X(k|k-1)) \tag{4}$$

$$P(k|k) = (I - K(k) \times H) \times P(k|k-1) \tag{5}$$

where H is a $m \times n$ matrix that links the state and measurement $Z(k)$, and R that denotes the original measurement noise covariance is analogous to Q [18].

Step 3: Recursive process by repeating Step 1 and Step 2. In view of the above descriptions, Step 1 makes a preparation for Step 2, which is the extension and improvement for the former again [17–20].

2.2 Extreme Learning Machine

Extreme Learning Machine (ELM), which is a new learning algorithm for single-hidden layer feedforward neural network (SLFN) proposed by G.B.Huang and has been widely used in a number of fields in recent years, has been proved that there are some superiorities to gradient-based learning algorithms like back-propagation (BP). (1) Compare to several hours or even more time taken in training neural network even for simple applications by utilizing traditional learning algorithms, the learning process of ELM can be finished in seconds. (2) ELM owns better performance during learning course than classic SLFN in most cases. (3) ELM is simple but can also solve the common problems, such as local minima, overtraining, overfitting and instability, existed in conventional gradient-based learning algorithms. (4) Many nondifferentiable activation functions can be adopted in ELM unlike classic SLFN methods which only employ differentiable activation functions [21, 22].

ELM uses a finite number of input-output samples, $(x_i, t_i) \in R^n \times R^m$, to train and construct a model, where x_i is a $n \times 1$ input vector and t_i is a $m \times 1$ target or output vector that can be expressed by Eq. (6) with $\overline{N}$ hidden nodes and activation function $g(x)$ to approximate N samples with zero error.

$$f_{\overline{N}}(x_j) = \sum_{i=1}^{\overline{N}} \beta_i g(w_i \cdot x_j + b_i) = t_j \quad j = 1, 2, \ldots, N \tag{6}$$

In Eq. (6), $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ denotes output weight vector connecting the ith hidden node and output nodes. $w_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ is input weight vector connecting the ith hidden node and input nodes, b_i expresses a threshold or hidden layer bias of the ith hidden node, and $w_i \cdot x_j$ shows the inner product of w_i and x_j. Equation (6) can also be simply given as

$$H\beta = T \tag{7}$$

$$H(a_1, \ldots, a_{\overline{N}}, b_1, \ldots, b_{\overline{N}}, x_1, \ldots x_N) = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_{\overline{N}} \cdot x_1 + b_{\overline{N}}) \\ \vdots & \cdots & \vdots \\ g(w_1 \cdot x_N + b_1) & \cdots & g(w_{\overline{N}} \cdot x_N + b_{\overline{N}}) \end{bmatrix}_{N \times \overline{N}} \tag{8}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\overline{N}}^T \end{bmatrix}_{\overline{N} \times m} \quad \text{and} \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m} \tag{9}$$

H is said to be hidden layer output matrix of neural network, the ith column of H is the ith hidden node output under input samples $x_1, \ldots x_N$ and the jth row of H denotes the output vector of hidden layer under one input sample x_j [21, 23].

In light of Eqs. (6) or (7), β as a weight matrix connecting the hidden nodes and output nodes is of great importance in whole learning process. Apparently, β should be the product of the inverse of H and T like following Eq. (10). Here the inverse $H^{\dagger}$ adopted the Moore-Penrose generalized inverse of H according to [21].

$$\beta = H^{\dagger}T \tag{10}$$

To sum up, give a set of training samples $(x_i, t_i) \in R^n \times R^m, i = 1, 2, \ldots, N$, select a suitable activation function $g(x)$ (e.g. sigmoidal function, radial basis, sine) and define a number of hidden nodes, the learning procedure of ELM algorithm can be executed in the following steps [21, 24]:

Step 1: Randomly distribute the input weight w_i and bias b_i of the hidden nodes, $i = 1, 2, \ldots, N$.

Step 2: Compute the hidden layer output matrix H according to Eq. (8).

Step 3: Compute the output weight β based on Eq. (10).

2.3 K-nearest Neighbor

K-nearest Neighbor (KNN) is an easy classification algorithm, which assumes that each class contains a plurality of samples and each sample has a unique mark to indicate its category. The principle of KNN is to select out K sample data that are nearest to a specific sample with unknown category by calculating the similarities between all samples with known category (or training data) and this specific sample (or testing data). Generally, the similarity can be obtained using Euclidean distance by Eq. (11) [25]. Ultimately, this unclassified sample belongs to the category that most of the K samples are in [26].

$$d = \sqrt{\sum_{i=1}^{L} (f_{traini} - f_{testi})^2} \tag{11}$$

where L is the number of researched features such as X, Y, Z; f_{traini} and f_{testi} denote a particular feature value of training data and testing data, respectively.

Obviously, the data including training samples with known classification marks and testing samples with unknown categories are preliminary requirements for KNN, and then the subsequent classification process are implemented as following:

Step 1: Calculate the distances on the basis of Eq. (11) between each testing sample and training data, then choose K training samples with the shortest distance or maximum similarity.

Step 2: Summarize the categories of K potential samples gotten in Step 1, and confirm an optimal class that more ones of K samples are belonged to, then the testing sample is necessary also in this class [25–27].

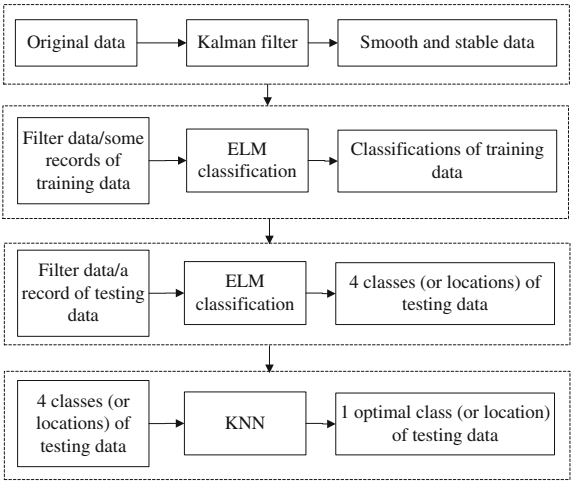

Fig. 1 Overall design of the proposed method or procedure in this chapter

2.4 Proposed Method or Procedure for Location Estimation

Generally, the solution of an issue depends on multi-method to exploit their cooperation by exerting respective advantages in various aspects. In this article, a hybrid learning method or procedure for location estimation is proposed by utilizing three algorithms to fully mine the useful information in different stages as shown in Fig. 1. Firstly, Kalman filter is functioned on the original magnetic field data for smoothing the fluctuation in order to attain a relative stable situation. Secondly, classification learning based on the filtering data by ELM is brought into effect according to Eqs. (7)–(10). Thirdly, four underlying classes that a particular testing data may belong are selected out in light of the results in second step. Fourthly, an optimal class for this testing data is finally ensured by applying KNN algorithm.

3 Experiments

3.1 Data and Experiment Conditions

Eight neighboring stations (Fig. 2 showed the environment where magnetic field data were collected) of an office building in Institute of Computing Technology Chinese Academy of Sciences were selected as the experiment and test sites. This environment is a typical workplace with computers, printers, desks, chairs and some other things. All these items may cause the magnetic field to fluctuate in any station at some time. The maximum distance between the neighboring stations is about 2.4 m and the minimum distance between them is about 1 m, and the covering area of the

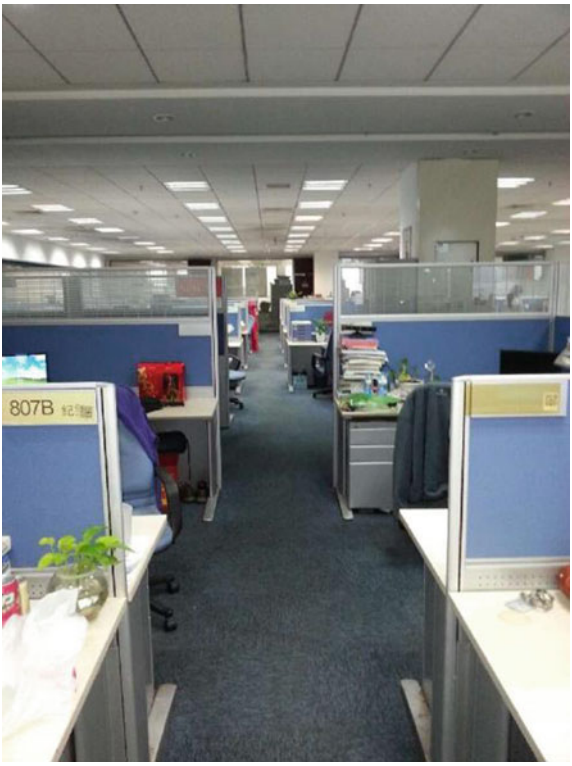

Fig. 2 Experiment environment tested in this chapter

experiment environment is about 6 m^2. The magnetic field data were collected only with the help of a magnetometer in smart-phone (HTC).

Each sampling started on an hour and got 25 records lasting about five seconds for every station. The magnetic field measurements, including three indicators (X, Y, Z), were acquired from 9 a.m. to 21 p.m. in weekdays and from 11 a.m. to 16 p.m. at weekend over 73 days. However, only the data of 46 days can be used in this article. Thereinto, the data in previous 37 days contained in training process and the others as testing data were to be concerned with the certification. And then, the other four indicators (H: total amount in horizontal direction, F: total amount, D: geomagnetic declination, I: geomagnetic inclination) were derived from X, Y and Z according to Eqs. (12) and (13) [28].

$$H = \sqrt{X^2 + Y^2} \qquad F = \sqrt{X^2 + Y^2 + Z^2} \tag{12}$$

$$D = \arctan(Y/X) \qquad I = \arctan(Z/H) \tag{13}$$

All algorithms for data processing and experiments were run in such computer conditions: (1) CPU: Intel (R) Core(TM)2 Duo CPU; (2) Memory: 2G; (3) Analysis software: Matlab R2009a.

3.2 Experiment Procedure

Three courses were projected here to realize the flow of Fig. 1 by utilizing the proposed method depicted in Sect. 2.

3.2.1 Data Preprocess

In order to attain the comparatively stable data, the stacking, a total of 325 records with seven indicators (X, Y, Z, H, F, D, I), of 1, 2, 3... 25 records of original magnetic field measurements in each station at a time point were constructed to be preprocessed by Kalman filter based on the steps in Sect. 2.1. Next, Four and two schemes for training and testing data in view of the filtering results were considered, respectively. As to the training data, (1) the mean of 125 filtering data in every station at each time point (eight stations, 37 days, 13 or 6 h in each day); (2) integrated data by supplementing (1) with the daily mean of eight stations (37 records for each station); (3) the last five ones of 125 filtering data in every station at each time point (eight stations, 37 days, 13 or 6 h in each day, 5 records per hour); (4) integrated data by supplementing (3) with the daily mean of eight stations (37 records for each station), the above four kinds of filtering data were prepared for training phase. Two types, the mean and the last one of 125 filtering data of a given testing sample, corresponding to the training data were applied to validate the classification model conceived in training phase by ELM algorithm. Note that the forementioned 125 filtering data were intercepted from 201 to 325 records so that the relative smooth values can be acquired. Figure 3 displayed the filtering results of three dimensions (X, Y, Z) in a station at some time point, and the steady trends of filtering records from 201 to 325 were also clearly appeared.

Figure 4 portrayed the variation characteristics of magnetic filed (X, Y, Z) in part of stations from 9 a.m. to 21 p.m.. In legend of subplot (1) (subplot (2)–(3) had the same legend), the letter S following a number denoted the station label (2, 5, 7) and the last number indicated how many days after the first measurements acquisition. It was evident that the magnetic field of each station was comparatively stable in three dimensions for long time although a certain degree of undulation, which may be the comprehensive effects of periodical changes and magnetic disturbance, existed in different hours. However, another issue can also be observed, that is station 5 and 7 owning the similar measurements especially in Y direction. The mutual influences caused by this similarity between any two stations would be the main reason for lower positioning accuracy, which had been validated in latter experiments. Whereas, how to distinguish these stations by using decomposition method and magnetic field features was beyond the scope of this chapter and will be explored in future study.

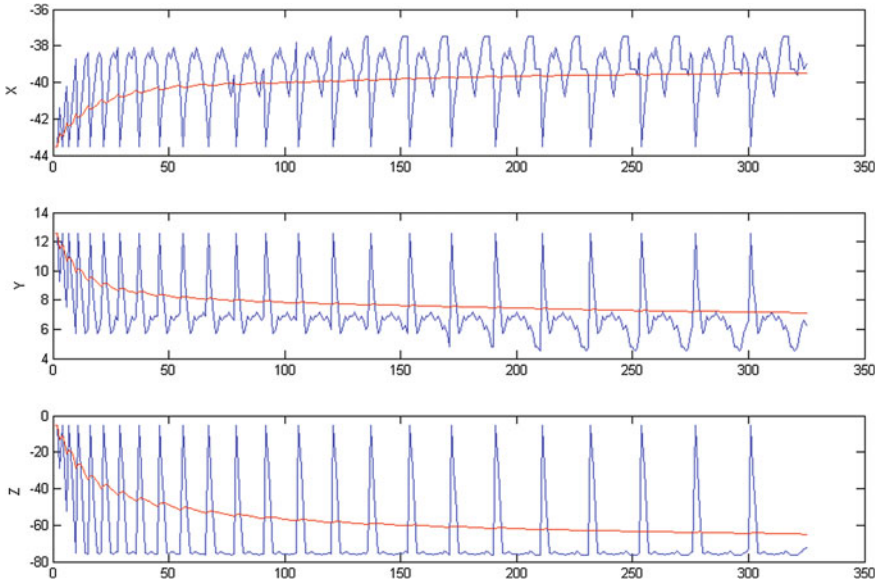

Fig. 3 Kalman filtering results in a station at some time

3.2.2 Classification Learning

Based on the prepared training and testing data, three groups (data with three indicators (X, Y, Z), data with five indicators (X, Y, Z, H, F) and data with seven indicators (X, Y, Z, H, F, D, I) of which were designed to be as the initial data. ELM algorithm began to execute Step 2 and Step 3 in Fig. 1. Noise was reduced by principal component analysis (PCA) firstly in order to make the energy to be rearranged in different dimensions of ELM data S (i.e. the initial filtering data preparing for ELM learning), including all the training data and a group of special testing data. In Table 1, S1 was the transformed ELM data, and which would be as the inputs to directly involved in ELM classification learning.

During the process of ELM learning, sigmoidal was selected as the activation function, the default number of hidden node was set as 10 and the number of output node was automatically assigned as the classification count that detected from training data. The input weights and biases of hidden nodes were randomly generated based on the number of hidden node and input samples. Where the randomness in this stage simplified the calculation and saved much time comparing with other neural network systems. According to these defined parameters and Eq. (8), output matrix H_{train} and the corresponding Moore-Penrose generalized inverse ($H^{\dagger}_{train}$) on the basis of training data can be gained. And then, the output weight β, which was the key for successive estimation of testing data, equaled to the product of $H^{\dagger}_{train}$ and the classification vector in training data. In the same manner, the output matrix H_{test} based on the testing sample was calculated so that the output vector for this particular

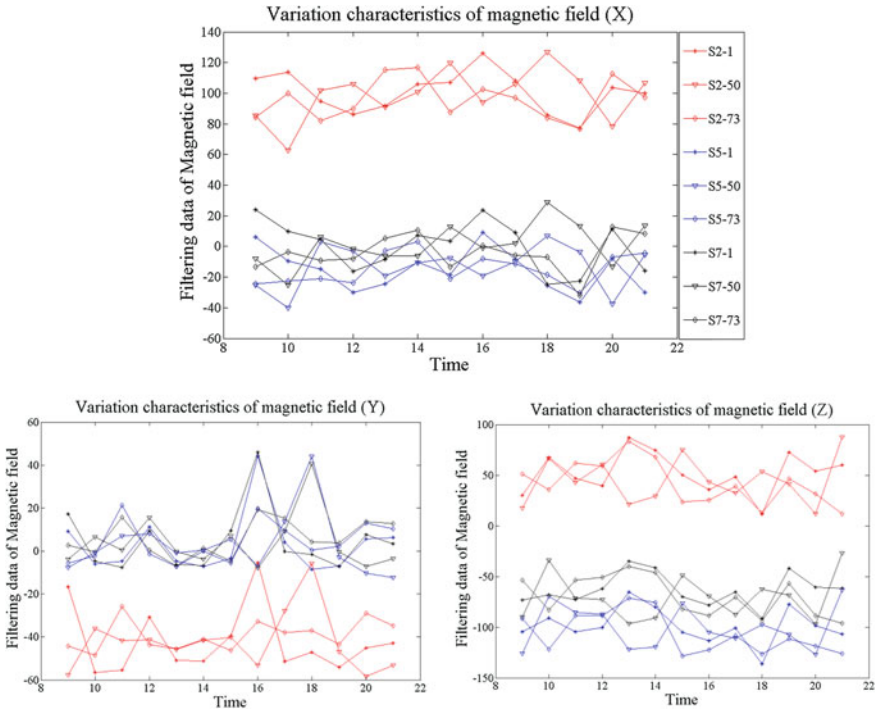

Fig. 4 Variation characteristics of magnetic field in three stations (2, 5, 7)

Table 1 Matlab codes for noise reduction by PCA

Procedure of PCA for noise reduction
S = S − repmat(mean(S), size(S, 1), 1);
C = cov(S);
[P, Lambda] = eig(C);
S1 = S*P;

testing sample would be acquired by Eq. (7). Normally, this testing sample should be endowed the class (or location) with maximum value in output vector. Here, four potential classes (or locations) with the first four maximum values in output vector were distilled to be used for reckoning the optimal location, where may be lived in these classes with larger probability.

Table 2 listed the accuracies to recognize exact locations for a group of testing data by using only one maximum ('1 max') and the first four maximum ('4 max') of output vector. Where the results acquired based on filtering data and original measurements were showed in left and right three columns, respectively. In Table 2, the digital label i (i = 1, 2, 3, 4) in train (i) means the type of training data described in the foregoing Sect. 3.2, and the digital label j (j = 1, 2) in test (j) denotes that the testing data were the mean or the last one of 125 records based on filtering or

Table 2 Accuracies to estimate exact locations for a group of testing data based on two ways

Data_Filter	1 max	4 max	Data_origin	1 max	4 max
F_ELM_train(1)_test(1)$_3$	0.627	0.950	**O_ELM_train(1)_test(1)$_3$**	0.596	0.927
F_ELM_train(2)_test(1)$_3$	0.637	0.948	**O_ELM_train(2)_test(1)$_3$**	0.590	0.938
F_ELM_train(3)_test(1)$_3$	0.619	0.951	**O_ELM_train(3)_test(1)$_3$**	0.605	0.921
F_ELM_train(4)_test(1)$_3$	0.646	0.938	**O_ELM_train(4)_test(1)$_3$**	0.584	0.928
F_ELM_train(1)_test(2)$_3$	0.646	0.936	**O_ELM_train(1)_test(2)$_3$**	0.595	0.957
F_ELM_train(2)_test(2)$_3$	0.651	0.930	**O_ELM_train(2)_test(2)$_3$**	0.582	0.945
F_ELM_train(3)_test(2)$_3$	0.617	0.941	**O_ELM_train(3)_test(2)$_3$**	0.587	0.939
F_ELM_train(4)_test(2)$_3$	0.639	0.951	**O_ELM_train(4)_test(2)$_3$**	0.593	0.950
F_ELM_train(1)_test(1)$_5$	0.721	0.974	**O_ELM_train(1)_test(1)$_5$**	0.659	0.976
F_ELM_train(2)_test(1)$_5$	0.726	0.971	**O_ELM_train(2)_test(1)$_5$**	0.695	0.974
F_ELM_train(3)_test(1)$_5$	0.721	0.980	**O_ELM_train(3)_test(1)$_5$**	0.665	0.974
F_ELM_train(4)_test(1)$_5$	0.698	0.971	**O_ELM_train(4)_test(1)$_5$**	0.662	0.971
F_ELM_train(1)_test(2)$_5$	0.713	0.968	**O_ELM_train(1)_test(2)$_5$**	0.655	0.970
F_ELM_train(2)_test(2)$_5$	0.706	0.971	**O_ELM_train(2)_test(2)$_5$**	0.639	0.973
F_ELM_train(3)_test(2)$_5$	0.688	0.976	**O_ELM_train(3)_test(2)$_5$**	0.674	0.966
F_ELM_train(4)_test(2)$_5$	0.715	0.979	**O_ELM_train(4)_test(2)$_5$**	0.678	0.977
F_ELM_train(1)_test(1)$_7$	0.668	0.976	**O_ELM_train(1)_test(1)$_7$**	0.666	0.979
F_ELM_train(2)_test(1)$_7$	0.718	0.980	**O_ELM_train(2)_test(1)$_7$**	0.659	0.973
F_ELM_train(3)_test(1)$_7$	0.683	0.971	**O_ELM_train(3)_test(1)$_7$**	0.633	0.957
F_ELM_train(4)_test(1)$_7$	0.712	0.956	**O_ELM_train(4)_test(1)$_7$**	0.662	0.965
F_ELM_train(1)_test(2)$_7$	0.700	0.973	**O_ELM_train(1)_test(2)$_7$**	0.659	0.973
F_ELM_train(2)_test(2)$_7$	0.686	0.966	**O_ELM_train(2)_test(2)$_7$**	0.601	0.960
F_ELM_train(3)_test(2)$_7$	0.697	0.977	**O_ELM_train(3)_test(2)$_7$**	0.671	0.966
F_ELM_train(4)_test(2)$_7$	0.657	0.979	**O_ELM_train(4)_test(2)$_7$**	0.668	0.971

original data. The subscript 3, 5 and 7 expressed the related indictors employed in ELM learning course.

Apparently, the classes with four maximum values from ELM learning results can rightly identify a location for the special testing sample in most cases in despite of datum type. However, the results based on filtering data had better performance than original measurements when only one maximum value was used to estimate a location, and the results with five indicators were superior to other situations again. Therefore, there may be a higher accuracy for indoor positioning if a suitable method can be functioned on these results.

Table 3 Comparing the positioning accuracies obtained by six methods

Data type	KNN	PCA-KNN	ELM_KNN	Kalman_KNN	Kalman_PCA-KNN	Kalman_ELM_KNN
F_ELM_train(1)_test(1)$_3$	0.183	**0.782**	**0.771**	0.182	**0.806**	***0.843***
F_ELM_train(2)_test(1)$_3$	0.197	**0.788**	**0.774**	0.182	**0.811**	***0.831***
F_ELM_train(3)_test(1)$_3$	0.125	0.742	0.744	0.455	0.803	0.725
F_ELM_train(4)_test(1)$_3$	0.125	0.748	0.761	0.165	**0.808**	0.748
F_ELM_train(1)_test(2)$_3$	0.111	0.755	0.747	0.215	0.800	***0.819***
F_ELM_train(2)_test(2)$_3$	0.113	0.758	0.745	0.215	0.803	***0.799***
F_ELM_train(3)_test(2)$_3$	0.226	0.755	0.753	**0.399**	0.800	0.733
F_ELM_train(4)_test(2)$_3$	0.226	0.759	0.761	0.215	0.802	0.771
F_ELM_train(1)_test(1)$_5$	0.184	**0.790**	**0.790**	0.182	0.797	***0.852***
F_ELM_train(2)_test(1)$_5$	0.184	**0.796**	**0.788**	0.182	0.798	***0.843***
F_ELM_train(3)_test(1)$_5$	0.194	0.738	0.765	0.443	0.800	0.766
F_ELM_train(4)_test(1)$_5$	0.194	0.741	0.750	0.276	0.798	0.783
F_ELM_train(1)_test(2)$_5$	0.116	0.752	0.753	0.215	**0.806**	***0.825***
F_ELM_train(2)_test(2)$_5$	0.117	0.755	0.762	0.215	**0.806**	***0.835***
F_ELM_train(3)_test(2)$_5$	0.232	0.753	0.777	**0.399**	0.799	0.771
F_ELM_train(4)_test(2)$_5$	0.232	0.762	0.761	0.215	0.802	0.782
F_ELM_train(1)_test(1)$_7$	0.195	**0.787**	**0.761**	0.304	0.797	***0.834***
F_ELM_train(2)_test(1)$_7$	0.195	**0.794**	0.756	0.304	0.797	***0.835***
F_ELM_train(3)_test(1)$_7$	0.125	0.744	0.742	0.212	0.803	0.725
F_ELM_train(4)_test(1)$_7$	0.125	0.750	0.742	0.391	**0.805**	0.747
F_ELM_train(1)_test(2)$_7$	0.113	0.764	0.752	0.209	0.803	***0.817***
F_ELM_train(2)_test(2)$_7$	0.113	0.767	0.750	0.209	**0.805**	***0.811***
F_ELM_train(3)_test(2)$_7$	**0.348**	0.761	0.755	**0.396**	0.803	0.742
F_ELM_train(4)_test(2)$_7$	**0.348**	0.767	**0.759**	0.209	**0.805**	0.748

3.2.3 Location Estimation

Finally, KNN algorithm introduced in Sect. 2.3 was applied to figure out the optimal location in view of the above results (four potential classes or locations) for a special testing sample. Equation (11) was brought to calculate the distances between this testing sample and the specified training data with four selected classes. Because there were many records with the same class in training data, the summations of these distances based on all features were then calculated and the class with minimum summation was ensured to be the optimal location. Note that: (1) the number of

training data for each station was the same, so the minimum summation and minimum mean had a consistent effect. (2) K (in KNN) equaled to 1 here and 4 was the potential sample size, whereafter Step 1 depicted in Sect. 2.3 was implemented to select out K (or 1) sample, which had been the optimal value, and Step 2 no longer enforced.

The comparisons of six methods, i.e. KNN, PCA-KNN (the function of PCA is to reduce noise like the usage in classification learning part), ELM_KNN, Kalman_KNN, Kalman_PCA_KNN and Kalman_ELM_KNN (proposed method in this chapter), based on eight kinds of data with three groups of indicators used in Table 2 were displayed in Table 3. Thereinto, the former three methods utilized the original magnetic field measurements, which were not preprocessed by Karman filter, while the filtering data were employed in the other methods. Each output obtained by every method expressed the mean positioning accuracy for all the testing data (9 days from 9 a.m. to 21 p.m. in weekday and from 11 a.m. to 16 p.m. in weekend).

As shown in Table 3, the positioning accuracies gained by KNN and Kalman_KNN were relative stable but much lower. Although PCA_KNN and ELM_KNN can effectively recognize the locations for testing data to some extent, the performance of them were obviously inferior to Kalman_PCA_KNN and Kalman_ELM_KNN in much of the times, the phenomenon of which presented that filtering preprocess for original magnetic field measurements, i.e. the comparatively smooth and stable data, can efficiently improve the subsequent positioning precision. The results of Kalman_PCA_KNN were consistent with regard to all types of data with three groups of indicators, whereas Kalman_ELM_KNN had the highest accuracy when the accumulated data (the mean of 125 filtering data in each station at every time) of eight stations in 37 days were taken in training process, and the capability of ELM learning with five indicators were especially prominent (marked by bold italic).

Figure 5 sketched the positioning performances by using four methods (except KNN and Kalman_KNN) at each station, which also unfurl the superiority of the proposed method (Kalman_ELM_KNN). In Fig. 5, a distinct characteristic existed in first three methods (subplot (1)–(6)) was that the positioning accuracies gained on account of eight kinds of learning data basically remained steady at each station while the larger differences aroused from diverse training data (eight kinds of learning data) by Kalman_ELM_KNN emerged in part of stations such as station1, 5 and 7. Nevertheless, it was found that Kalman_ELM_KNN can present the excellent performance for identifying locations in most stations when the mean of filtering data as the training data, i.e. (1, 1), (2, 1), (1, 2) and (2, 2) ((i, j) in legend was corresponding to the datum type in Table 3), were built into ELM learning, and which were evidently embodied in subplot (8) (5 indicator of Kalman_ELM_KNN). All of these were accordant with the results in Table 3.

3.3 Discussion

Kalman filter, ELM learning and KNN algorithms were adopted in different stages to fully excavate and apply the considerable information for indoor positioning. This

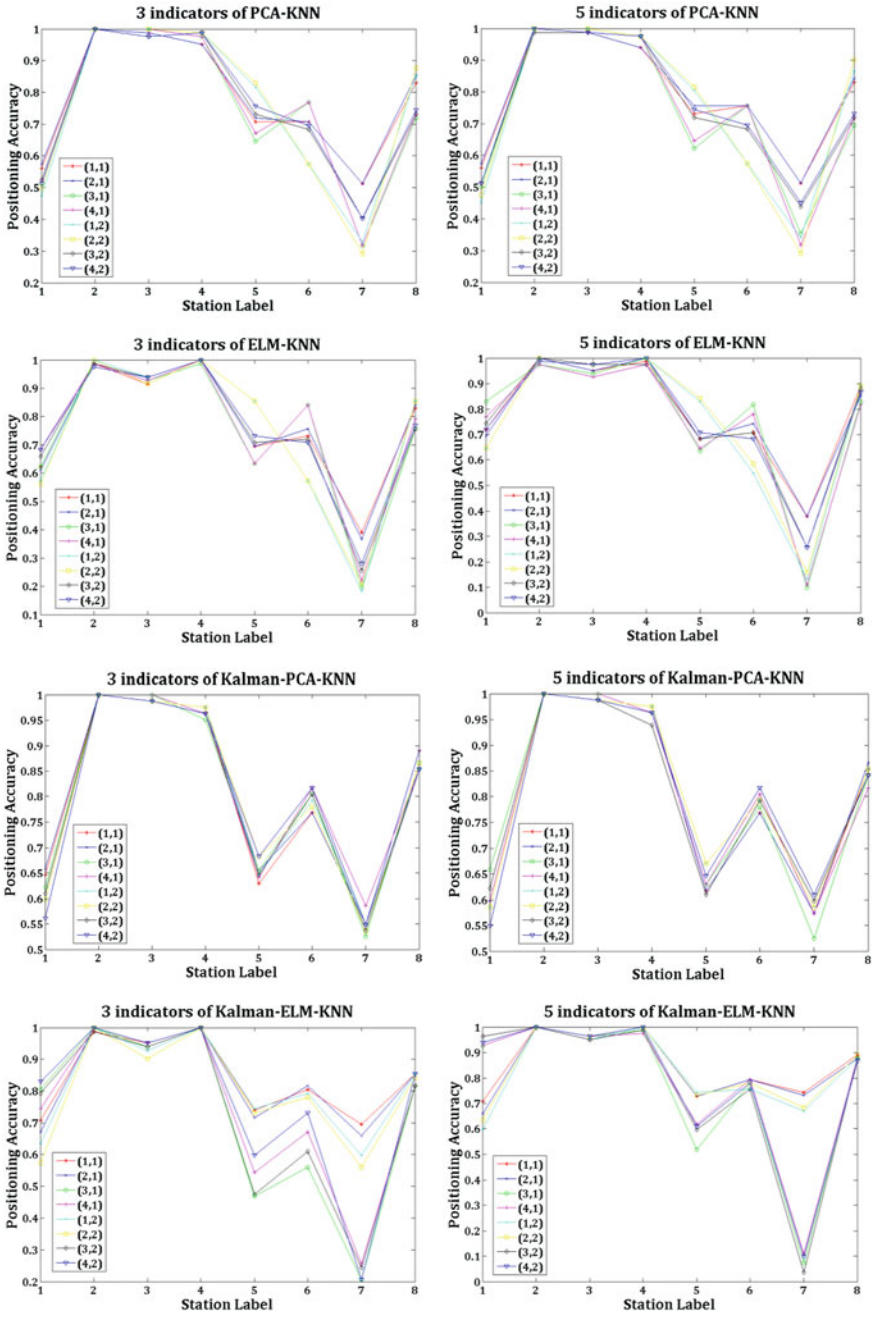

Fig. 5 Comparisons of positioning accuracy at each station using four methods based on eight kinds of learning data with two groups of indicators

proposed idea were validated by a series of experiments based on eight kinds of learning data with three groups of indicators, and compared with other five methods displayed in Table 3 and Fig. 5. Also the whole processes only recur to a smart-phone with magnetometer but no additional radio network or devices so that it can be simply extended to some other applications.

As shown in Tables 2 and 3, the stable data as inputs preprocessed by Kalman filter for estimating the locations were more effective than original magnetic field measurements in most instances. The precisions of single KNN and Kalman_KNN, which should be integrated with other algorithms, were the worst. And the similar capability of PCA_KNN and ELM_KNN revealed that noise process by PCA and learning enhancement by ELM can efficiently exert the functions of KNN. Kalman_PCA_KNN enforced the procedures of PCA_KNN based on the filtering data and had an obvious improvement, which likewise demonstrated the validity of Kalman filter. The positioning accuracies of compared five methods in view of eight kinds of learning data with three groups of indicators were relative stable, in other words, these methods were insensitive to any formats of data and indicators. Kalman_ELM_KNN gave prominence to its higher precision when the cumulative data (mean of 125 filtering data in each station at one time point) of eight stations were taken in the learning process, and the performances with five indicators (X, Y, Z, H, F) in experiments were the best.

The analogous outcomes also emerged in Fig. 5 that presented the positioning accuracies of each station by using four methods based on eight kinds of learning data with two groups of indicators. Thereinto, the precisions of station 2, 3 and 4 were comparatively higher and up to more than 90 %, which illuminated that these stations owned unique measurements and clearly distinguished from other stations. Station 1, 5 and 7, especially station 7 (maximum: 51 % in subplot (2), 38 % in subplot (4), 61 % in subplot (6) and 74 % in subplot (8)), had more difficult for right recognition. The accuracy difference on station 6 and 8 by PCA_KNN, ELM_KNN and Kalman_PCA_KNN were not big, whereas the precisions gained by ELM_KNN in station 1 and 5 were slightly better than the results by PCA_KNN and Kalman_PCA_KNN. The performance of Kalman_ELM_KNN with the mean data and five indicators in subplot (8) were notable in majority of stations except station 1.

4 Conclusions

Evidence had been demonstrated that the magnetic field, produced by the Earth's uppermost lithosphere, with relative larger difference between any two stations and relative stability for each station in a long time had a potential ability to support the accurate indoor positioning [15, 16]. However, the experiments reported in this chapter proved that it is possible to utilize the local magnetic field for indoor positioning even in complex environments such as an office building, some stations of where may own the similar measurements.

The positioning technique provided in this chapter didn't require to deploy the additional equipments such as a radio network in advance, only some algorithms (Kalman filter, ELM learning and KNN) were united to preprocess, analyze and compute based on the measurements for location recognition. (1) Kalman filter was functioned on the primitive values (eight stations, 46 days) and the last 125 filtered records of each station at a time point were selected so that the smooth data can be utilized to continue the subsequent steps. (2) The filtered data obtained in (1) were separated into two parts: training and testing data, which were then inputted to ELM learning procedure and four potential locations were accordingly fixed for a particular testing sample. (3) KNN method was finally introduced to process the four intermediate results obtained in (2) for picking out the optimal location. The experiments based on eight kinds of learning data with three groups of indicators tested and verified the feasibility of this proposed method, the superiority of which was also apparent by comparing with other five methods. Although these data was only from the part of working area in an office building, this technique can be extended to other districts for indoor positioning, too.

With more and more extensive applications of location-based service (LBS), the research of which had been given considerable concerns. In this setting and based on the findings in this chapter, there exist important aspects of future research as to the local magnetic field in environments, including the followings:

(1) How to select out the representative sites, the measurements of where will be used for interpolating unknown places so that a continuous indoor positioning can be implemented, in environments through analysis and comparison methods.
(2) How to set the interpolation distances according to a particular environment in order to obtain the accurate indoor locations.
(3) How to ulteriorly improve the positioning precision by efficiently differentiating those sites, which may be the adjacent or non-adjacent locations, with similar measurements through decomposition method and magnetic field features.

Acknowledgments This study was supported by the National Natural Science Foundation of China (41201410, 61173066 and 61005045). The authors also thank Mr. Xinlong Jiang for debugging the ELM programs.

References

1. A. Sanchez, S. Elvira, A.D. Castro et al., in Low Cost Indoor Ultrasonic Positioning Implemented in FPGA. The 35th Annual Conference of the IEEE Industrial Electronics Society (IECON), pp. 2709–2714 (2009)
2. E. Garcia, J.J. Garcia, A. Hernandez et al., in Ultrasonic Positioning System by using UWB Techniques. Proceedings of the 14th IEEE International Conference on Emerging Technologies and Factory Automation ETFA'09, pp. 1734–1737 (2009)
3. C. Lee, Y. Chang, G.H. Park et al., in Indoor Positioning System Based on Incident Angles of Infrared Emitters. The 30th Annual Conference of the IEEE Industrial Electronics Society, pp. 2218–2222 (2004)

4. S. Mori, K. Hida, K. Sawada, in Advanced Car Positioning Method Using Infrared Beacon. The 8th International Conference on ITS Telecommunications, pp. 45–50 (2008)
5. Y.P. Wang, X. Yang, Y.T. Zhao et al., in Bluetooth Positioning using RSSI and Triangulation. IEEE 10th Consumer Communications and Networking Conference (CCNC), pp. 837–842 (2008)
6. L. Chen, H. Kuusniemi, Y.W. Chen et al., in Information Filter with Speed Detection for Indoor Bluetooth Positioning. International Conference on Localization and GNSS (ICL-GNSS 2011), pp. 47–52 (2011)
7. S.S. Saab, Z.S. Nakad, A standalone RFID indoor positioning system using passive tags. IEEE Trans. Industr. Electron. **58**(5), 1961–1970 (2011)
8. K.S. Bok, Y.H. Park, J.I. Pee et al., in Location Acquisition Method Based on RFID in Indoor Environments. The 2011 International Conference on Multimedia, Computer Graphics and Broadcasting (MulGraB), vol. 263, pp. 310–318 (2011)
9. A. Lim, K. Zhang, in A Robust RFID-Based Method for Precise Indoor Positioning. The 19th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE), pp. 1189–1199 (2006)
10. Y.W. Ma, C.F. Lai, J.M. Hsu et al., RFID-based positioning system for telematics location-aware applications. Wireless Pers. Commun. **59**, 95–108 (2011)
11. R.W. Ouyang, A.K.S. Wong, C.T. Lea, Indoor location estimation with reduced calibration exploiting unlabeled data via hybrid generative/discriminative learning. IEEE Trans. Mob. Comput. **11**(11), 1613–1625 (2012)
12. F. Lassabe, P. Canalda, P. Chatonnay et al., Indoor Wi-Fi positioning: techniques and systems. Ann. Telecommun. **64**, 651–664 (2009)
13. Q. Yang, S.J. Pan, V.W. Zheng, Estimating location using Wi-Fi. IEEE Intell. Syst. **23**(1), 8–13 (2008)
14. G.V. Zaruba, M. Huber, F.A. Kanmangar, Indoor location tracking using RSSI readings from a single Wi-Fi access point. Wireless Netw. **13**, 221–235 (2007)
15. J. Haverinen, D.A. Kemppainen, in A Global Self-Localization Technique Utilizing Local Anomalies of the Ambient Magnetic Field. IEEE international conference on robotics and automation, pp. 3142–3147 (2009)
16. J. Haverinen, A. Kemppainen, in A Geomagnetic Field Based on Positioning Technique for Underground Mines. International Symposium on Robotic and Sensors Environments, pp. 7–12 (2011)
17. D. Simon, in Kalman Filtering. Embedded Systems Programming, pp. 72–79 (2001)
18. G. Welch, G. Bishop, An Introduction to the Kalman Filter. http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf, pp. 1–16 (2006)
19. A. Constantinos, B.A. Moshe, N.K. Haris, Nonlinear Kalman filtering algorithms for on-line calibration of dynamic traffic assignment models. IEEE Trans. Intell. Transp. Syst. **8**(4), 661–670 (2007)
20. D. Simon, Kalman filtering with state constraints: a survey of linear and nonlinear algorithms. IET Control Theory Appl. **4**(8), 1303–1318 (2010)
21. G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications. Neurocomputing **70**, 489–501 (2006)
22. G.B. Huang, L. Chen, Enhanced random search based incremental extreme learning machine. Neurocomputing **71**, 3460–3468 (2008)
23. N.Y. Liang, G.B. Huang, P.N. Saratchandran et al., A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans. Netw. **17**(6), 1411–1423 (2006)
24. J.F. Liu, Y.Q. Chen, M.J. Liu et al., SELM: semi-supervised ELM with application in sparse calibrated location estimation. Neurocomputing **74**, 2566–2572 (2011)
25. B. Altintas, T. Serif, in Indoor Location Detection with a RSS-Based Short Term Memory Technique (KNN-STM). IEEE Percom Workshop on Pervasive Wireless Networking, pp. 794–798 (2012)
26. K-Nearest Neighbor algorithm, http://baike.baidu.com/view/1485833.htm

27. D.A. Stanley, R. Min, Z.N. Yuan et al., in A Deep Non-linear Feature Mapping for Larger-Margin KNN Classification. IEEE International Conference on Data Mining, pp. 357–366 (2009)
28. W.Y. Xu, *Geomagnetism* (Seismological Press, Beijing, 2003)

A Novel Scene Based Robust Video Watermarking Scheme in DWT Domain Using Extreme Learning Machine

Charu Agarwal, Anurag Mishra, Arpita Sharma and Girija Chetty

Abstract In this chapter, we present a novel fast and robust watermarking scheme for three different standard video in RGB uncompressed AVI format in DWT domain using a newly developed SLFN commonly known as Extreme Learning Machine (ELM). The embedding is carried out by using scene detection. The LL4 sub-band coefficients of frames constitute the dataset to train the ELM in millisecond time. The output of the ELM is used to embed a binary watermark in the video frames using a pre-specified formula. The resultant video exhibits good visual quality. Five different video processing attacks are executed over signed video. The extracted watermarks from the signed and attacked video yield high normalized correlation (NC) values and low Bit Error Rate (BER) values. This indicates successful watermark recovery and the embedding scheme is found to be robust against these common attacks. It is concluded that the proposed watermarking scheme produces best results due to optimized embedding facilitated by fast training of the ELM. The proposed scheme is found to be suitable for developing real time video watermarking applications due to its low time complexity.

C. Agarwal (✉)
Department of Computer Science, University of Delhi,
Delhi 110007, India
e-mail: agarwalcharu2@rediffmail.com

A. Mishra
Department of Electronics, Deendayal Upadhyay College, University of Delhi,
Delhi 110015, India
e-mail: anurag_cse2003@yahoo.com

A. Sharma
Department of Computer Science, Deendayal Upadhyay College, University of Delhi,
Delhi 110015, India
e-mail: arpt1_mishra1@yahoo.com

G. Chetty
Faculty of Information Sciences and Engineering, University of Canberra,
Canberra, Australia
e-mail: Girija.Chetty@canberra.edu.au

F. Sun et al. (eds.), *Extreme Learning Machines 2013: Algorithms and Applications*, Adaptation, Learning, and Optimization 16, DOI: 10.1007/978-3-319-04741-6_15,

Keywords Extreme learning machine · Video watermarking · Uncompressed RGB AVI format · Scene detection

1 Introduction

Copyright issues related to digital content lead to its authentication which is critically required for distribution of the content into safe hands. Digital watermarking of multimedia content is one such technique which has gained tremendous importance for last more than one decade. This application requires fast execution of embedding and extraction processes so as to enable the application to finish on a real time scale. Besides this, it is expected to be robust against common signal processing attacks without losing perceptible quality of the of the host signal. These twin requirements are often found to be conflicting to each other. Due to this reason, the problem of watermarking is perceived as an optimization problem. Many researchers have dealt with the issue of optimization of these processes in images. However, not much research work is done to develop fast and optimized embedding and extraction schemes for video using soft computing tools.

Hartung et al. [1] published one of the pioneering works for watermarking of compressed and uncompressed video. They embed an encrypted pseudo-noise signal as watermark within the MPEG-2 encoded video to obtain an invisible, statistically unobtrusive and robust scheme. Their scheme is also found to work for other hybrid transform coding schemes like MPEG-1, MPEG-4, H.261 and H.263. For processing of their frames, they have used DCT method within their algorithm. They have also delved upon the issue of time complexity of their embedding algorithm vis-à-vis other methods. Biswas et al. [2] have presented a new compressed video watermarking procedure which embeds several binary images as watermarks decomposed and obtained from a single watermark image into various scenes of the subject MPEG-2 encoded video sequence. Their scheme is found to be substantially more effective and robust against spatial attacks such as scaling, rotation, frame averaging and filtering besides temporal attacks like frame dropping and temporal shifting. Rajab et al. [3] and Faragallah [4] have recently used SVD technique to implement video watermarking. Rajab et al. [3] embed their watermark in the SVD transformed video in diagonal wise and block wise fashion. They evaluate the performance with respect to imperceptibility, robustness and data payload or capacity. They argue that the diagonal-wise algorithm achieves better robustness while the block-wise algorithm gives higher pay load rate. Faragallah [4] presented an efficient, robust and imperceptible video watermarking technique based on SVD decomposition performed in DWT domain. In this chapter, two levels of high frequency band HH and middle frequency band LH are SVD transformed and the watermark are hidden into them. Their proposed algorithm is tested in the presence of image and video processing attacks and their experimental results prove that this method survives these attacks. They attribute these positive results to the amalgamation of DWT and SVD transforms they use in their work. Wu et al. [5] have very recently proposed a flexible H.264/AVC

compressed video watermarking scheme using particle swarm optimization (PSO) based dither modulation. The technique proposed by them is found to be robust against commonly employed watermarking attacks. In order to consider watermark imperceptibility within the video, the authors have used swarm optimization. They claim that the imperceptibility is enhanced by using this optimization method. El' Arbi et al. [6] have delved upon the issue of video watermarking based on neural networks. They have employed a back propagation neural network (BPNN) to implement a video watermarking scheme based on multi resolution motion estimation. They said that their embedding algorithm is robust against common video processing attacks. However, they have not touched upon the issue of time complexity. It is a well known fact that the BPNN while propagating back are often found to get trapped into local minima and therefore its training time span is found to be large. On the contrary, any practical video processing such as watermarking should be efficient in terms of time complexity issues [1]. Chen et al. [7] presented a compressed video watermarking algorithm based on synergetic neural network in IWT domain. They use pattern recognition method of synergetic neural network during watermark extraction. They claim that their algorithm results in fine performance of robustness and speediness. A novel digital video watermarking scheme based on 3D-DWT and Artificial Neural Network is proposed by Li et al. [8]. In this case, a 3D-DWT was performed on each selected video shots and then the watermark is embedded in the LL sub-band wavelet coefficients. Their scheme shows strong robustness against common video processing attacks. The frame coefficients are selected adaptively to embed the watermark and to ensure perceptual invisibility. The embedding intensity was adaptively controlled using statistical characteristics such as mean and standard deviation. Their scheme implements a blind extraction process. Isac et al. [9] presented a compact review of image and video watermarking techniques using neural networks. Leelavathy et al. [10] presented a scene based raw video watermarking in Discrete Multi-wavelet domain. They also use Quantization Index Modulation (QIM) to implement their embedding algorithm. They claim that by using QIM, the watermark is embedded into selected multi-wavelet coefficients by quantizing them. They generate scrambled watermarks using a set of secret keys and each watermark is embedded in each motionless scene of the video. They claim that their scheme is robust against frame dropping, frame averaging, swapping and statistical analysis attacks.

In this chapter, we successfully embed a binary image as a watermark into all frames of three different RGB uncompressed AVI video by using DWT- ELM watermarking scheme. We extend the preliminary work we have proposed previously in [11]. The ELM training is particularly important in this case as it optimizes the watermark embedding to produce best results in minimum time. For this purpose, first, the video is decomposed into non overlapping frames which led to detection of scenes. The scene based RGB frames thus obtained are converted into YCbCr color space. A 4-level DWT of luminance component (Y) of all video frames is computed. The LL4 sub-band coefficients are used to develop a data set which is fed to a newly developed fast neural network known as extreme learning machine

(ELM). The training of the ELM is completed within few milliseconds. The output of this machine is used to embed the coefficients of a binary image as watermark into LL4 sub-band coefficients using a pre-specified formula. The signed video sequences are found to be completely imperceptible after watermark embedding as indicated by high PSNR values. The extraction of the watermarks from these frames yield high normalized correlation (NC) and low bit error-rate (BER) values which indicate successful watermark recovery. The signed video frames are also examined for robustness by executing five different video processing attacks. The attacks used in the present work are: (1) Scaling (20, 40, 60, 80 and 100 %), (2) Gaussian Noise (with mean = 0 and variance 0.001, 0.01, 0.03 and 0.05), (3) JPEG (compression ratio = 5, 25, 50, 75 and 90 %), (4) Frame dropping (10, 30, 50, 70 and 90 %), and (5) Frame Averaging (5, 10, 15 and 20 %). Watermarks are extracted from the attacked frames as well. In this case, the experimental results indicate that the proposed watermarking scheme is robust against the selected video processing attacks. All these processes are carried out in few seconds. On the other hand, the ELM training is carried out in millisecond time. It is concluded that the proposed ELM based fast embedding and extraction scheme is suitable for real time applications which is one of the most important considerations for multimedia processing.

The chapter is organized as follows. Section 2 gives a brief theoretical description of ELM algorithm. Section 3 describes the proposed embedding and extraction algorithm. Section 4 delves upon the results obtained in this simulation and its discussion. Finally, Sect. 5 presents the conclusion followed by list of references.

2 Extreme Learning Machine

The Extreme Learning Machine [12–16] is based on a Single hidden Layer Feed forward Neural Network (SLFN) architecture. This differs from the conventional training algorithms such as Back Propagation (BP) algorithms which may face difficulties in manual tuning control parameters and local minima. On the contrary, training of ELM is very fast, it has a good accuracy and offers a solution in the form of system of linear equations. For a given network architecture, ELM does not have any control parameters like stopping criteria, learning rate, learning epochs etc., and thus, the implementation of this network is very simple. In this algorithm, the input weights and hidden layer biases are randomly chosen which are based on some continuous probability distribution function. We choose uniform probability distribution in our simulation. The output weights are then analytically calculated using a simple generalized inverse method known as Moore-Penrose generalized pseudo inverse [15].

2.1 Mathematics of ELM Model

Given a series of training samples $(x_i, y_i)_{i=1,2,...,N}$ and $\hat{N}$ the number of hidden neurons where $x_i = (x_{i1}, ..., x_{in}) \in \Re^n$ and $y_i = (y_{i1}, ..., y_{im}) \in \Re^m$, the actual outputs of the single-hidden-layer feed forward neural network (SLFN) with activation function $g(x)$ for these N training data is mathematically modeled as

$$\sum_{k=1}^{\hat{N}} \beta_k g(< w_k, x_i > + b_k) = o_i, \forall i = 1, \ldots, N \tag{1}$$

where $w_k = (w_{k1}, ..., w_{kn})$ is a weight vector connecting the kth hidden neuron, $\beta_k = (\beta_{k1}, ..., \beta_{km})$ is the weight vector connecting the kth hidden neuron and output neurons and b_k is the threshold bias of the kth hidden neuron. The weight vectors w_k are randomly chosen. The term $\langle w_k, x_i \rangle$ denotes the inner product of the vectors w_k and x_i and g is the activation function.

The above N equations can be written as

$$H\beta = 0 \tag{2}$$

and in practical applications $\hat{N}$ is usually much less than the number N of training samples and $H\beta \neq Y$, where

$$H = \begin{bmatrix} g(< w_1, x_1 > + b_1) \ldots g(< w_{\hat{N}}, x_1 > + b_{\hat{N}}) \\ \cdots \\ \cdots \\ \cdots \\ g(< w_1, x_N > + b_1) \ldots g(< w_{\hat{N}}, x_N > + b_{\hat{N}}) \end{bmatrix}_{N \times \hat{N}}$$

$$\beta = \begin{bmatrix} \beta_1 \\ \cdot \\ \cdot \\ \cdot \\ \beta_N \end{bmatrix}_{\hat{N} \times m} \quad 0 = \begin{bmatrix} 0_1 \\ \cdot \\ \cdot \\ \cdot \\ 0_N \end{bmatrix}_{N \times m} \quad Y = \begin{bmatrix} y_1 \\ \cdot \\ \cdot \\ \cdot \\ y_N \end{bmatrix}_{N \times m} \tag{3}$$

The matrix H is called the hidden layer output matrix. For fixed input weights $w_k = (w_{k1}, ..., w_{kn})$ and hidden layer biases b_k, we get the least-squares solution $\hat{\beta}$ of the linear system of equation $H\beta = Y$ with minimum norm of output weights β, which gives a good generalization performance. The resulting $\hat{\beta}$ is given by $\hat{\beta} = H^+Y$ where matrix H^+ is the Moore-Penrose generalized inverse of matrix H [15]. The above algorithm may be summarized as follows:

2.2 The ELM Algorithm

Given a training set
$S = \{(x_i, y_i) \in \Re^{m+n},\ y_i \in \Re^m\}_{i=1}^{N} \sum$, for activation function $g(x)$ and the number of hidden neurons $\hat{N}$;
Step 1: For $k = 1, ..., \hat{N}$ randomly assign the input weight vector $w_k \in \Re^n$ and bias $b_k \in \Re$.
Step 2: Determine the hidden layer output matrix H.
Step 3: Calculate H^+.
Step 4: Calculate the output weights matrix $\widehat{\beta}$ by $\hat{\beta} = H^+T$.

Many activation functions can be used for ELM computation. In the present case, Sigmoid activation function is used to train the ELM.

2.3 Computing the Moore-Penrose Generalized Inverse of a Matrix

Definition 1.1: A matrix G of order $\hat{N} \times N$ is the Moore-Penrose generalized inverse of real matrix A of order if $N \times \hat{N} AGA = A$, $GAG = G$ and AG, GA are symmetric matrices.

Several methods, for example orthogonal projection, orthogonalization method, iterative methods and singular value decomposition (SVD) methods exist to calculate the Moore-Penrose generalized inverse of a real matrix. In ELM algorithm, the SVD method is used to calculate the Moore-Penrose generalized inverse of H. Unlike other learning methods, ELM is very well suited for both differential and non-differential activation functions. As stated above, in the present work, computations are done using "Sigmoid" activation function for $\hat{N} = 20$. $\widehat{\beta}$ is a column vector and is used to embed the binary watermark coefficient into luminance component (Y) of the video frame by using a pre specified formula. This is described in detail in Sect. 3.

3 Proposed DWT-ELM Based Video Watermarking Scheme

Figure 1 depicts the block diagram of the proposed video watermark embedding scheme.

The host video is first divided into non-overlapping frames of size M × N. Secondly, the scene detection algorithm gives the number of scenes available in the given video comprising of these non-overlapping frames. Let T is the total number of such frames and k is the total number of available scenes. The watermark (W) used in this work is a binary image of size (x, y) which depends on the size of original video frame (M × N), total number of DWT levels employed and the number of available scenes (k).

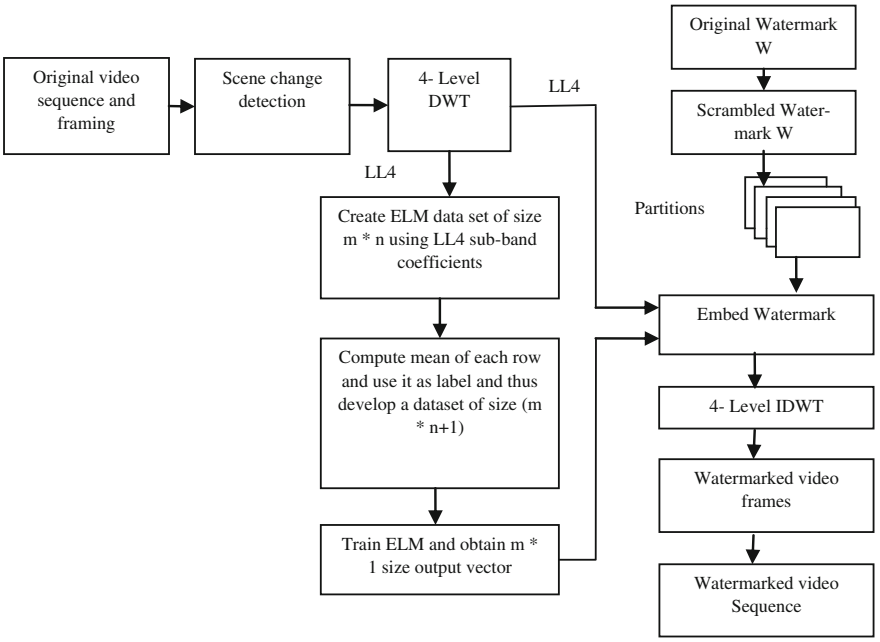

Fig. 1 Block diagram of watermark embedding scheme

3.1 Scene Change Detection

In the proposed scheme, we use histogram difference method for scene change detection and is given by Listing 1.

Listing 1: Scene Change Detection Algorithm.

Step 1. Calculate the histogram of the red component of all the frames
Step 2. Calculate the total difference of the whole histogram using the formula given by Eq. (4)

$$D(z, z+1) = \sum_{z=1}^{T} |A_z(r) - A_{z+1}(r)| \tag{4}$$

where $A_z(r)$ is the histogram value for the red component r in the zth frame
Step 3. If $D(z, z+1) > threshold$ a scene change is detected, where threshold is empirically determined.

3.2 Embedding the Watermark

The watermark embedding algorithm is given in Listing 2.

Listing 2: Algorithm for Watermark Embedding.

Step 1. Apply scene change detection algorithm (Listing 1) to detect the available scenes (k) from the original video frames
Step 2. Convert every RGB frame F_i (i = 1, 2, 3…T) of each scene k to YCbCr format
Step 3. Obtain scrambled watermark W_p by performing pseudorandom permutation on original watermark W

$$W_p = Permute(W)$$

Step 4. Decompose the permuted watermark W_p into k watermark sub-images such as, $W_p^1, W_p^2, \ldots W_p^k$ where a specific watermark is used to modify the frames of the corresponding scene
Step 5. Apply 4—level DWT using Haar filter to the luminance (Y) component of every ith frame of each scene k of the host video to obtain the $LL4_i^k$ sub-band coefficients of size m × n
Step 6. Compute the output column vector using ELM as follows:

1. Consider an initial data set of size (m × n) using $LL4_i^k$ sub-band coefficients and calculate row wise the arithmetic mean of the coefficients of all rows
2. For each row, use the mean value as label and arrange them in the first column of the data set. Thus obtain a final data set of size m × (n + 1)
3. Train the ELM in regression mode using this data set and obtain an output column vector (E_i^k) of size m × 1. This column vector is further used to embed the watermark in the $LL4_i^k$ sub-band coefficients.

Step 7. Embed the binary watermark sub-image (W^k) into the $LL4_i^k$ sub-band coefficients of every ith video frame of each scene k using the formula given by Eq. (5)

$$wLL4_i^k(q,r) = LL4_i^k(q,r) + (E_i^k(q) * W_p^k(q,r)) \tag{5}$$

where q = 1, 2…m and r = 1, 2…n
Step 8. After embedding the watermark in every ith frame of each scene k of the host video, apply 4—level inverse DWT to every ith signed frame of each scene k of the host video to obtain the watermarked luminance component of the ith frame. Convert every ith frame of each scene k of the signed video back to the RGB format to obtain the watermarked video.

The embedded frames are further examined for its perceptible quality by computing PSNR individually and taking an average PSNR of all frames put together. Equations (6) and (7) respectively give mathematical formulae for PSNR and AVG_PSNR. The computed results are presented and discussed in detail in Sect. 4.

$$PSNR = 10log_{10}(\frac{255^2}{MSE}) \tag{6}$$

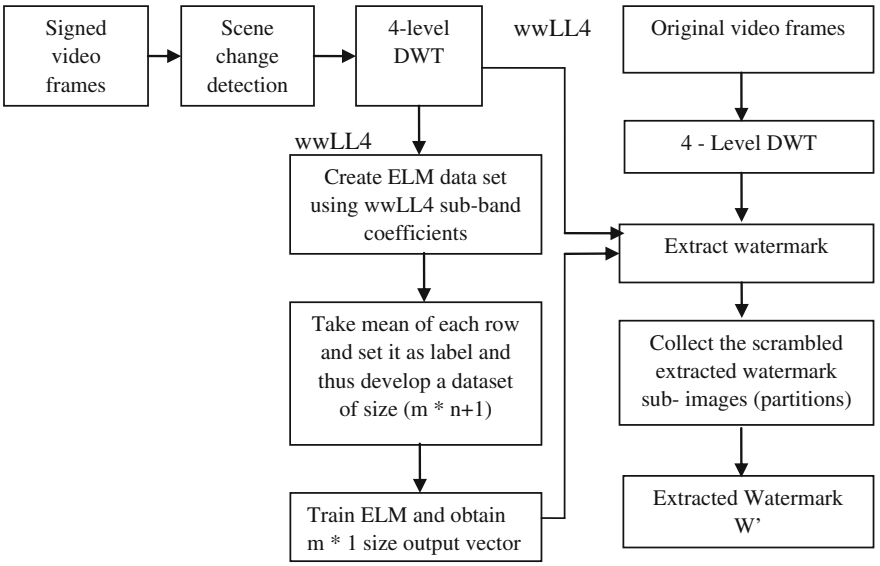

Fig. 2 Block diagram of watermark extraction scheme

$$\text{AVG\_PSNR} = \frac{\sum_{i=1}^{T} \text{PSNR}}{T} \tag{7}$$

where T is the total number of frames in the video sequence.

3.3 Extracting the Watermark from Signed Video

Figure 2 depicts the proposed video watermark extraction scheme. For this purpose, $NC(W, W')$ normalized correlation and bit error rate $BER(W, W')$ parameters are computed. W and W′ are respectively original and recovered watermarks. These two parameters are given by Eqs. (8) and (9) respectively.

$$NC(W, W') = \frac{\sum_{i=1}^{x} \sum_{j=1}^{y} [W(i, j).W'(i, j)]}{\sum_{i=1}^{x} \sum_{j=1}^{y} [W(i, j)]^2} \tag{8}$$

$$BER(W, W') = \frac{1}{xy} \sum_{j=1}^{xy} |W'(j) - W(j)| \tag{9}$$

The extraction algorithm is given in Listing 2.

Listing 2: Algorithm for Watermark Extraction

Step 1. Apply scene change detection algorithm (Listing 1) to detect the available scenes (k) from the signed video frames
Step 2. Convert every ith RGB frame of signed video $F_i'(i = 1, 2, 3...T)$ and original video of each scene k to YCbCr format
Step 3. Apply 4—level DWT to the luminance (Y) component of every ith frame of each scene k of signed video and original video to obtain the $wwLL4_i^k$ and $LL4_i^k$ sub-bands of size m × n
Step 4. Compute the output column vector using ELM as follows:

1. Consider an initial data set of size (m × n) using $wwLL4_i^k$ sub-band coefficients and row wise calculate the arithmetic mean of the coefficients of all rows
2. For each row, use the mean value as label and arrange them in the first column of the data set. Thus obtain a final data set of size m × (n + 1)
3. Train the ELM in regression mode using this data set and obtain an output column vector (wE_i^k) of size m × 1. This column vector is further used to embed the watermark in the $wwLL4_i^k$ sub-band coefficients.

Step 5. Extract the watermark sub-image from every ith frame of each scene k using Eq. (10).

$$wW_p^k(q, r) = \frac{wwLL4_i^k(q, r) - LL4_i^k(q, r)}{wE_i^k(q)} \tag{10}$$

where q = 1, 2…m and r = 1, 2…n
Step 6. Compute average extracted binary watermark sub-images wW^k for every scene k from i extracted scrambled watermark sub-images wW_p^k obtained from every ith frame of each scene k of the signed video. Construct the extracted binary watermark image W' from the extracted k binary watermark sub-images wW^k.

These signed video frames are also examined for robustness by executing five different video processing attacks. These are: (1) Scaling (20, 40, 60, 80 and 100 %), (2) Gaussian Noise (with mean=0 and variance 0.001, 0.01, 0.03 and 0.05), (3) JPEG (compression ratio=5, 25, 50, 75 and 90 %), (4) Frame dropping (10, 30, 50, 70 and 90 %), and (5) Frame Averaging (5, 10, 15 and 20 %). The watermarks are subsequently recovered from attacked frames and get matched with the original ones. For this purpose, normalized correlation $NC(W, W')$ and bit error rate $BER(W, W')$ parameters are computed. W and W' respectively being the original and recovered watermarks. The results are compiled and discussed in Sect. 4.

4 Experimental Results and Discussion

The performance of the proposed watermarking scheme is evaluated on three standard CIF (352 × 288) video sequences namely News, Silent and Hall_Monitor in RGB uncompressed AVI format having frame rate of 30 fps and each consisting of 300

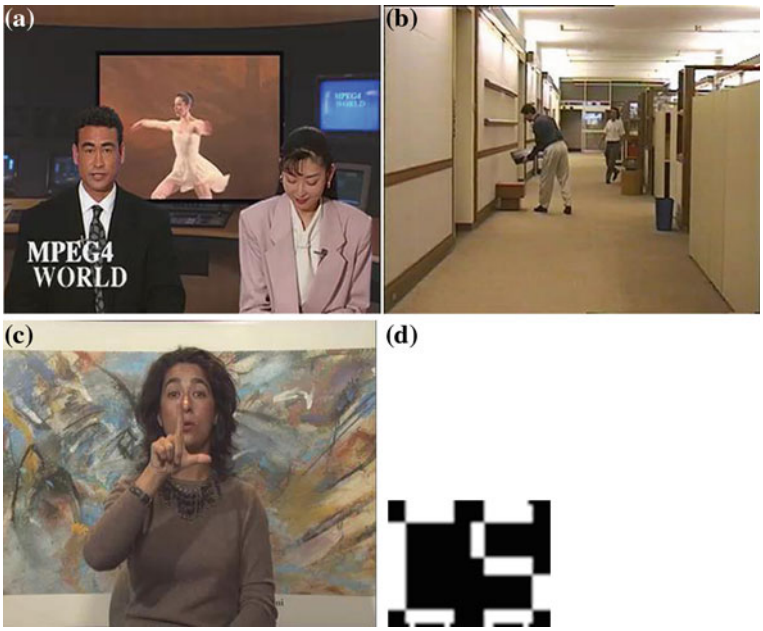

Fig. 3 **a**–**c** 100th original video frame of video sequence News, Hall_Monitor and Silent respectively and **d** Original watermark

Fig. 4 100th signed video frame of video sequence **a** News (43.1621 dB), **b** Hall_Monitor (43.2017 dB) and **c** Silent (43.045 dB)

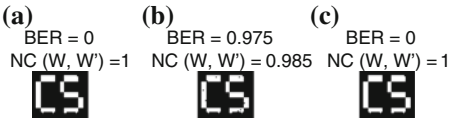

Fig. 5 **a**–**c** Extracted watermarks from the signed video sequences **a** News, **b** Hall_Monitor and **c** Silent with BER (%) and NC (W, W′) on top

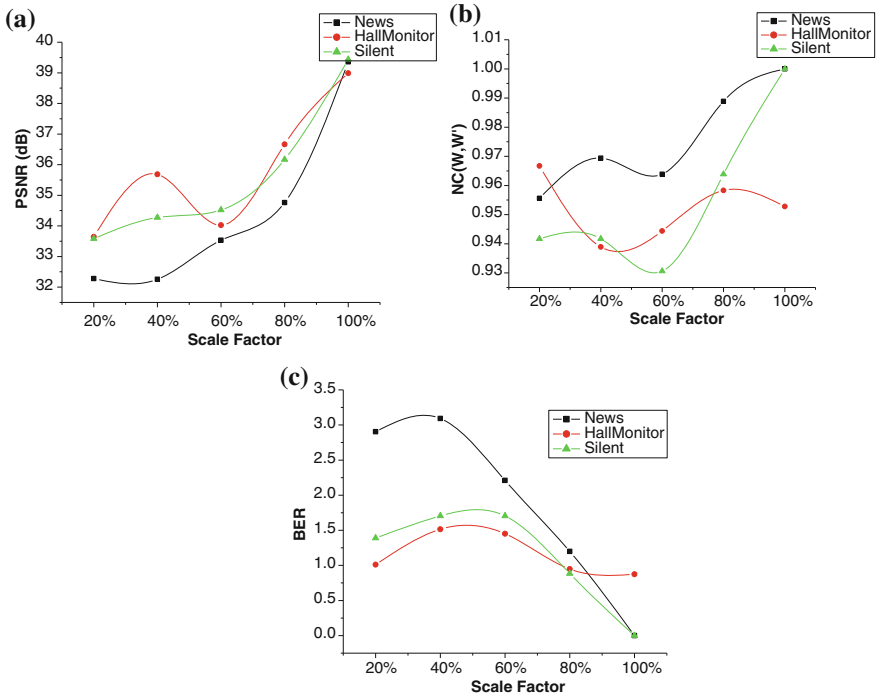

Fig. 6 **a**–**c** Plot of PSNR, NC (W, W′) and BER (%) w.r.t scaling factor

frames. A binary watermark of size 44 × 36 is embedded in all frames of these videos by using DWT-ELM scheme. Figure 3a–c depicts the 100th original frame of the video sequences News, Hall_Monitor and Silent respectively and Fig. 3d depicts the original binary watermark. Figure 4a–c depicts the signed frames respectively obtained from Fig. 3a–c. Figure 5a–c depicts the binary watermarks extracted from the three video sequences.

The average PSNR in our simulation is 43.1621, 43.2017 and 43.045 dB respectively for News, Hall_Monitor and Silent sequences. We further report high computed values of normalized cross correlation NC (W, W′) for all three video sequences. The computed NC (W, W′) values in our work are 1.0, 0.985 and 1.0 for these three video respectively. We obtain BER (%) values as 0.0, 0.975 and 0.0 respectively for the three video sequences. These results indicate that the proposed watermarking

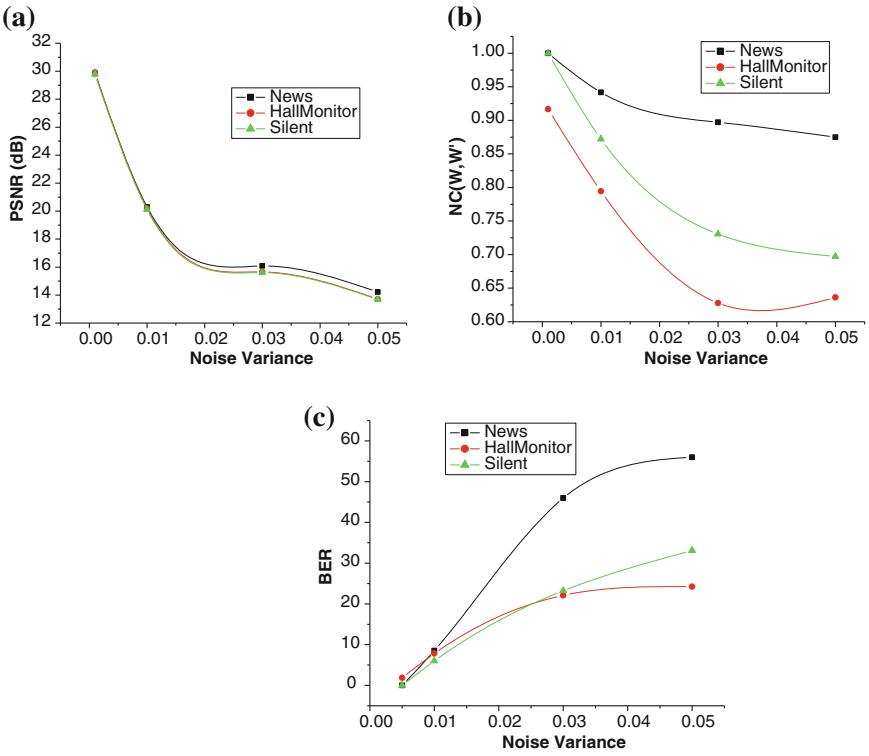

Fig. 7 **a–c** Plot of PSNR, NC (W, W′) and BER (%) w.r.t. Gaussian noise density

scheme is capable of maintaining the visual quality of all frames along with a successful watermark recovery. This is particularly true in this work due to optimized embedding facilitated by ELM training.

To examine the robustness of the proposed watermarking scheme, five different video processing attacks are carried out on the signed video sequences. PSNR, NC (W, W′) and BER (%) are calculated with respect to variation in respective attack parameter and plots are shown in Figs. 6, 7, 8, 9 and 10.

(a) Scaling: In this case, the video frames are scaled up to different sizes of the signed frame using bicubic interpolation method and reverted back. These sizes are 20, 40, 60, 80 and 100 %. Figure 6a–c respectively show the plot of PSNR, NC (W, W′) and BER (%) w.r.t. different scaling size.
(b) Gaussian Noise: This noise is added to signed frames by taking mean = 0 and variance = 0.001, 0.01, 0.03 and 0.05. Figure 7a–c shows the plot of PSNR, NC (W, W′) and BER (%) w.r.t. noise variance.
(c) JPEG Compression: As the host video is available in RGB uncompressed AVI format, it is subject to JPEG compression also. Figure 8a–c show plot of PSNR,

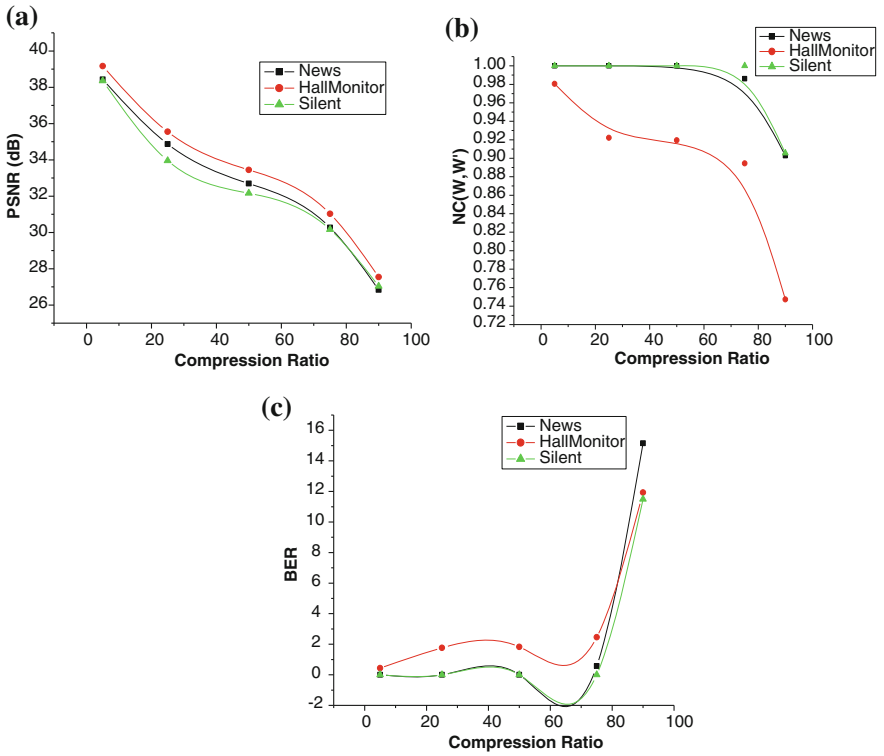

Fig. 8 **a–c** Plot of PSNR, NC (W, W′) and BER (%) w.r.t. JPEG compression ratio

NC (W, W′) and BER (%) w.r.t. variation in compression ratio (5, 25, 50, 75 and 90 %).

(d) Frame Dropping: For a video sequence having a large number of frames, dropping of a few redundant frames of a scene is considered as a natural video processing attack. It can be executed by removing a fraction of total frames from the video sequence. In this simulation, the percentage of dropped frames of a scene varies as 10, 30, 50, 70 and 90 %. Figure 9a–c show the plot of PSNR, NC (W, W′) and BER (%) w.r.t. the percentage of dropped frames.

(e) Frame Averaging: This is a very common video processing attack. In this case, the current frame is replaced by the average of two neighboring frames. In the present work, a variable percentage of averaged frames is taken into account. Figure 10a–c show the plot of PSNR, NC (W, W′) and BER (%) w.r.t. the percentage of averaged frames.

Note that the plot of PSNR and NC (W, W′) is found to be similar in case of all these attacks. For any efficient watermarking scheme, the visual quality of signed image / video frame and robustness are expected to be high. In this work, the results are clearly indicative of good optimization of visual quality and robustness obtained

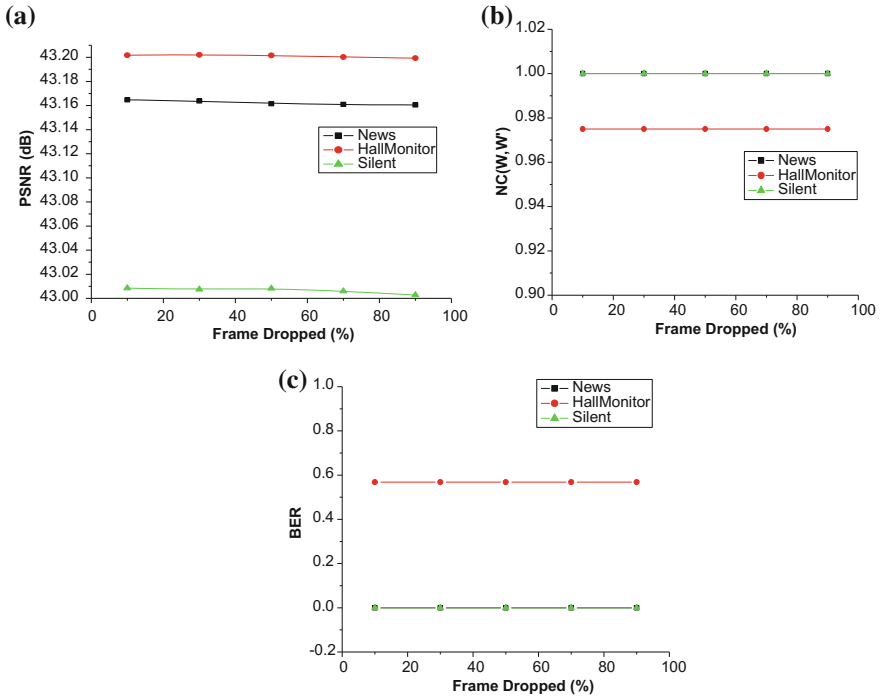

Fig. 9 **a–c** Plot of PSNR, NC (W, W′) and BER (%) w.r.t. number of dropped frames

Table 1 Time (seconds) consumed by different processes of the proposed scheme

For all 300 frames	News	Silent	Hall monitor
ELM training time	0.2365	0.2846	0.2969
Embedding time	32.9809	33.2819	33.3906
Extraction time	19.2317	20.0017	20.0156

by using ELM algorithm with minimum time complexity. The BER (%) is expected to show an inverse behavior with respect to NC (W, W′). Figures 6c–10c indicate the same.

To analyze the issue of time complexity of the proposed watermarking scheme, we take into account ELM training time, embedding and extraction time for 300 frames for all three video sequences. Table 1 compiles these results. Note that the ELM gets trained in millisecond time for all 300 frames. Similarly, embedding and extraction for all 300 frames are carried out in seconds. It is important to mention that the embedding time constitutes decomposition of video into frames, scene detection and actual embedding of the watermark. Similarly, extraction time is computed by taking into account decomposition of video into frames, scene detection and actual extraction of watermark.

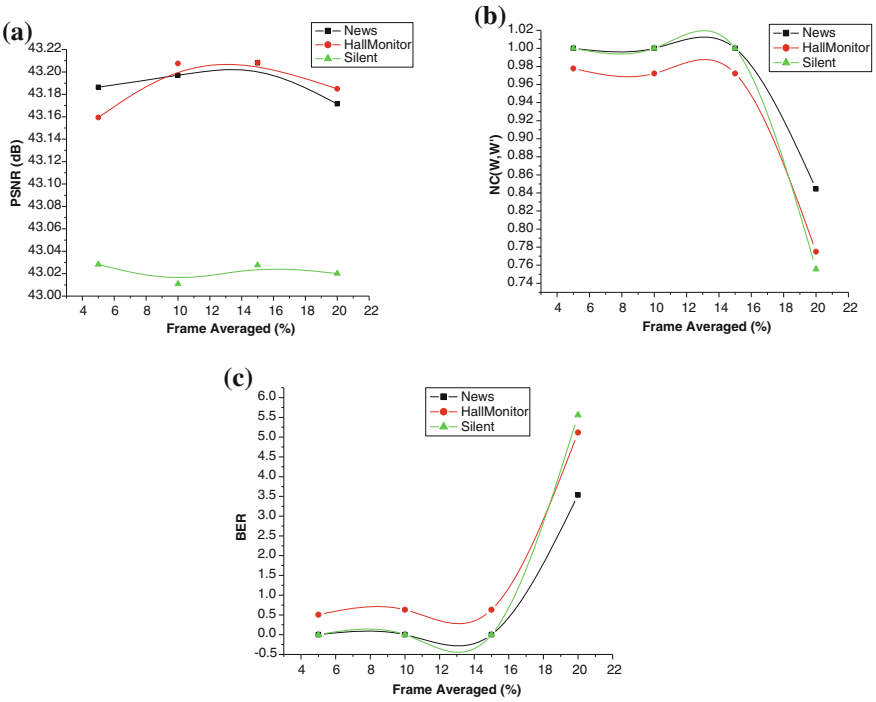

Fig. 10 **a–c** Plot of PSNR, NC (W, W′) and BER (%) w.r.t. number of averaged frames

The training of ELM is an integral part of both embedding and extraction procedures. We therefore conclude that the embedding and extraction using DWT-ELM is capable to implement real time video watermarking application.

5 Conclusions

We successfully demonstrate a novel scene based fast and robust video watermarking scheme for three standard video in RGB uncompressed AVI format. This scheme is implemented in DWT domain using a newly developed fast neural network known as Extreme Learning Machine (ELM). The fast training of this machine is suitable for optimized video watermarking on a real time scale. The perceptible quality of the video is good as indicated by high PSNR values. Watermark recovery is successful as indicated by high normalized cross correlation values and low bit error rate values between embedded and extracted watermarks. The robustness of the proposed scheme is examined by carrying out five different video processing attacks. This scheme is found to be robust against selected attacks. It is concluded that the proposed scheme produces best results due to optimized embedding facilitated by training of ELM in minimum time and overall the algorithm is suitable for developing real time video watermarking applications.

References

1. F. Hartung, B. Girod, Watermarking of uncompressed and compressed video. Sig. Process. **66**(3), 283–301 (1998)
2. S. Biswas, E.M. Petriu, An adaptive compressed MPEG-2 video watermarking scheme. IEEE Trans. Instrum. Measur. **54**(5), 1853–1861 (2005)
3. L. Rajab, T.A. Khatib, A.A. Haj, Video watermarking algorithms using the SVD transform. Eur. J. Sci. Res. **30**(3), 389–401 (2009)
4. O.S. Fargallah, Efficient video watermarking based on singular value decomposition in the discrete wavelet transform domain. AEU Int. J. Electron. Commun. **67**(3), 189–196 (2013)
5. C.H. Wu, Y. Zheng, W.H. Ip, C.Y. Chan, K.L. Yung, Z.M. Lu, A Flexible n H.264/AVC compressed video watermarking scheme using particle swarm optimization based dither modulation. Int. J. Electron. Commun. **65**, 27–36 (2011)
6. M. El'Arbi, C.B. Amar, H. Nicolas, Video watermarking based on neural networks, in *IEEE International Conference on Multimedia and Expo*, pp. 1577–1580, (2006)
7. Y.-Q. Chen, L.-H. Pen, Streaming media watermarking algorithm based on synergetic neural network, in *International conference on Wavelet Analysis and Pattern Recognition*, pp. 271–275, 2008
8. X. Li, R. Wang, A video watermarking scheme based on 3D-DWT and neural network, in *9th IEEE International Symposium on Multimedia*, pp. 110–114, 2007
9. B. Isac, V. Santhi, A study on digital image and video watermarking schemes using neural networks. Int. J. Comput. Appl. **12**9, 1–6 (2011)
10. N. Leelavathy, E.V. Prasad S. Srinivas Kumar, A scene based video watermarking in discrete multiwavelet domain. Int. J. Multi. Sci. Eng. **3**7, 12–16 (2012)
11. A. Mishra, A. Goel, R. Singh, G. Chetty, L. Singh, A novel image watermarking scheme using extreme learning machine, in *IEEE World Congress on Computational Intelligence*, pp. 1–6, 2012
12. M.-B. Lin, G.-B. Huang, P. Saratchandran, N. Sudararajan, Fully complex extreme learning machine. Neurocomputing **68**, 306–314 (2005)
13. G.-B. Huang, Q.-Y. Zhu, C.K. Siew, Extreme learning machine. Neurocomputing **70**, 489–501 (2006)
14. G.-B. Huang, Q.-Y. Zhu C.K Siew, Real-time learning capability of neural networks. IEEE Trans. Neural Netw. **17**4, 863–878 (2006)
15. G.-B. Huang, The Matlab code for ELM is available on http://www.ntu.edu.sg
16. D. Serre, *Matrices: Theory and Applications* (Springer, 2002)