Neurocomputing 128 (2014) 31–41

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Parallelized extreme learning machine ensemble based on min–max modular network

Xiao-Lin Wang [a,b], Yang-Yang Chen [a,b], Hai Zhao [a,b], Bao-Liang Lu [a,b,*]

[a] Center for Brain-Like Computing and Machine Intelligence, Department of Computer Science and Engineering, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai 200240, China

[b] MOE-Microsoft Key Laboratory for Intelligent Computing and Intelligent Systems, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai 200240, China

ARTICLE INFO

Article history:
Received 30 September 2012
Received in revised form
28 January 2013
Accepted 3 February 2013
Available online 26 October 2013

Keywords:
Extreme learning machine
Min–max modular network
Big data
Ensemble method
Parallel learning

ABSTRACT

Extreme Learning Machine (ELM) as an emergent technology has shown its promising performance in many applications. This paper proposes a parallelized ELM ensemble based on the Min–Max Modular network (M^3-network) to meet the challenge of the so-called big data. The proposed M^3-ELM first decomposes classification problems into smaller subproblems, then trains an ELM for each subproblem, and in the end ensembles these ELMs with the M^3-network. Twelve data sets including both benchmarks and real-world applications are employed to test the proposed method. The experimental results show that M^3-ELM not only speeds up the training phrases by 1.6–4.6 times but also reduces the test errors by 0.37–19.51% compared with the normal ELM. The results also indicate that M^3-ELM possesses scalability on large-scale tasks and accuracy improvement on imbalanced tasks.

1. Introduction

Extreme Learning Machine (ELM) as an emergent technology has recently attracted the attention from more and more researchers. Compared with traditional machine learning methods such as back propagation [1] and support vector machine [2,3], it provides better generalization performance at a much faster learning speed and with least human intervening [4].

The essence of ELM is that, different from the common understanding of learning, the hidden layer of a single-hidden layer feedforward network needs not be tuned [5–7,4]. One of the typical implementations of ELM is to employ random computational nodes in the hidden layer, and then resolve the output weights using the least-square method.

Ensemble methods consist in using multiple models to obtain better predictive performance than that could be obtained from any of the constituent models [8,9]. This technique has been proved very effective in many applications [10–12]. The ensemble method of Min–Max Modular network (M^3-network) is proposed by Lu and Ito as an approach to complex pattern classification tasks [13]. It follows the principle of divide-and-conquer – dividing the whole task into smaller pieces and then solving them one by one [14,15].

Though ELM has been well researched as a singular classifier, yet ELM ensembles are less explored for pattern classification tasks. However, recently such a demand has been raised by the so-called big data. This term refers to a collection of data sets so large and complex that it becomes awkward to work with using on-hand database management tools [16]. Conventional machine learning methods work poorly on big data. Those big data datasets are usually too large to be fully loaded into the memories of computers though this procedure is required by most conventional methods. In addition, the running time of those methods on big data datasets is usually extremely long if a massively parallel computing system is not available.

In this paper, we propose a parallelized ELM ensemble method based on M^3-network, named M^3-ELM, to meet the challenge of big data. In addition to the high efficiency brought by parallelism, this method also solves imbalanced classification tasks well as its task decomposition can convert imbalanced ones into balanced ones. A wide range of classification tasks are employed to test the accuracy and scalability of the proposed method.

The rest of this paper is organized as follows. Related work is first reviewed in Section 2. Then ELM and M^3-network are introduced in Section 3 and Section 4, respectively. After that, the proposed M^3-ELM is described and analyzed in Section 5. The experimental results and analysis are presented in Section 6. In the end, we conclude the paper with a description of the future work in Section 7.

* Corresponding author at: Center for Brain-Like Computing and Machine Intelligence, Department of Computer Science and Engineering, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai 200240, China.
E-mail address: bllu@sjtu.edu.cn (B.-L. Lu).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.02.053

2. Related work

ELM ensembles have been less explored compared to the large volume of papers about ELM, and only a few related researches have been reported.

An integration of several ELMs is proposed by Sun et al. [17] to predict the future sales amount. Several ELM networks were connected in parallel and the average of the ELMs' outputs was used as the final predicted sales amount. The resulting ensemble has better generalization performance.

Heeswijk et al. [18] investigate the adaptive ensemble models of ELM on the application of one-step ahead prediction. The ensemble weights are tuned online to adapt to non-stationary time series. The empirical studies verify that the proposed method not only achieves a testing error comparable to LS-SVM on stationary time series, but also possesses adaptivity to non-stationary time series.

Lan et al. [19] propose an ensemble of online sequential ELMs (EOS-ELM). An EOS-ELM model comprises multiple online sequential ELMs (OS-ELM), and takes the average value of their outputs as the output. The intuition behind is that the nature of randomly generated parameters makes each OS-ELM network distinct, thus they may have different adaptive capacity to the new data. The experimental results show that EOS-ELM is more stable than the original OS-ELM in most problems.

Heeswijk et al. [20] propose a GPU-accelerated ELM ensemble for large scale regression applications. The proposed method first trains multiple ELMs independently through an efficient Leave-One-Out method (LOO) on the whole training sets, and then ensembles these ELMs through weighted voting. The LOO method accomplishes both model training and model structure selection (determining the number of hidden neurons) for each component ELM. In addition, the LOO error is employed to calibrate the ensemble weights.

Escandell-Montero et al. [21] propose an ensemble of ELMs using a regularized committee for regression problems. A regularized version of least squares fitting is employed to calculate the combining weights. The experiments on real-world regression data sets demonstrate that their method generally outperforms the conventional ELM.

This paper employs the M^3-network to ensemble ELM to improve its scalability on large-scale classification problems, which has not been addressed before. In addition, the ensemble methods in the related work are mainly combining the outputs of component ELMs linearly, while the proposed method leverages the M^3-network to derive the prediction instead.

3. ELM

ELM, proposed by Huang et al. [4–7,22–24], is a single-hidden layer feedforward network. The output function of ELM can be formulated as follows:

$$f(x)=\sum_{i=1}^{L}\beta_i g(x;w_i,b_i), \tag{1}$$

where L is the number of hidden nodes, β_i is the output weight vector that connects the i-th hidden node and output nodes, g is the activation function, w_i is the input weight vector that connects the i-th hidden node and input nodes, and b_i is the bias of the i-th hidden node.

Two kinds of hidden nodes are usually employed in ELM [4,25]. One is additive hidden nodes whose activation function can be formulated as

$$g(x;w_i,b_i)=g(x^T w_i+b_i). \tag{2}$$

The other is Radial Basis Function (RBF) hidden nodes whose activation function can be formulated as,

$$g(x;w_i,b_i)=g\left(\frac{\|x-w_i\|}{b_i}\right). \tag{3}$$

ELM adopts a training method that is quite different from ordinary single-hidden layer feedforward networks [5,6]. The input weights w_i and biases b_i are chosen randomly and the output weights β_i are calculated analytically. For a training set $(x_k,y_k)_{k=1}^{L}$, the training of ELM can be formulated as follows:

$$\arg\min_{\beta}(\|Y-H\beta\|^2), \tag{4}$$

where,

$$X=\begin{pmatrix}x_1^T\\ \vdots\\ x_N^T\end{pmatrix},\quad Y=\begin{pmatrix}y_1^T\\ \vdots\\ y_N^T\end{pmatrix},\quad \beta=\begin{pmatrix}\beta_1^T\\ \vdots\\ \beta_L^T\end{pmatrix},$$

$$H=\begin{pmatrix}g(x_1;w_1,\beta_1) & \cdots & g(x_1;w_L,\beta_L)\\ \vdots & \cdots & \vdots\\ g(x_N;w_1,\beta_1) & \cdots & g(x_N;w_L,\beta_L)\end{pmatrix}.$$

The parameter β is determined by

$$\hat{\beta}=H^{\dagger}Y=(H^TH)^{-1}H^TY \tag{5}$$

where $H^{\dagger}$ is the Moore–Penrose generalized inverse of H.

Regularized ELM is proposed to increase the stability of the trained model especially when the number of hidden neurons is large [26,4]. It achieves better generalization performance than the basic one in real-world applications [24,27]. The training of L2-regularized ELM can be formulated as follows:

$$\arg\min_{\beta}(C\|Y-H\beta\|^2+\|\beta\|^2), \tag{6}$$

where C is the trade-off between the training error and the regularization. The corresponding solution of β is

$$\hat{\beta}=\left(H^TH+\frac{I}{C}\right)^{-1}H^TY. \tag{7}$$

ELM achieves high classification accuracy in many applications. In theory, Huang et al. prove that ELM can approximate any continuous target function on any compact input sets [5,6].

4. M^3-network

This section introduces the ensemble method of M^3-network which is used in Section 5 to scale ELMs to big data. M^3-network is proposed by Lu and Ito to solve complex classification problems in 1997 [13,14]. Its work flow consists of task decomposition, training component classifiers and module combination.

4.1. Task decomposition

Let $\mathcal{X}$ denote the training samples for a K-class classification problem:

$$\mathcal{X}=\cup_{i=1}^{K}\mathcal{X}_i=\cup_{i=1}^{K}\{x_l^i\}_{l=1}^{\mathcal{L}_i} \tag{8}$$

where $\mathcal{X}_i$ is the training sample set of the class $\mathcal{C}_i$, x_l^i is the l-th sample, and $\mathcal{L}_i$ is the number of samples in the class $\mathcal{C}_i$.

This K-class classification problem can be divided into $K(K-1)/2$ smaller binary subproblems through the one-versus-one (OVO) strategy as follows:

$$\mathcal{T}_{ij}=\{(x_l^i,+1)\}_{l=1}^{\mathcal{L}_i}\cup\{(x_l^j,-1)\}_{l=1}^{\mathcal{L}_j}\quad \text{for } i,\ j=1,\ldots,K \text{ and } i\neq j \tag{9}$$

where the classes $\mathcal{C}_i$ and $\mathcal{C}_j$ are taken as positive and negative

classes, respectively. Alternatively, a K-class classification problem can also be divided into K binary subproblems through the one-versus-all (OVA) strategy as follows:

$$\mathcal{T}_i = \{(x_l^i, +1)\}_{l=1}^{\mathcal{L}_i} \cup (\cup_{j=1, j\neq i}^{K} \{(x_l^j, -1)\}_{l=1}^{\mathcal{L}_j})$$
$$\text{for } i = 1, \ldots, K \tag{10}$$

Note that multi-labeled classification problems can only employ the OVA strategy. The reason is that a sample can have more than one class labels in multi-labeled problems [28], and a sample might belong to both the positive and negative classes in some binary tasks derived by OVO.

These binary subproblems defined by Eqs. (9) and (10) can be further divided. Assume that the sample set $\mathcal{X}_i$ is partitioned into $\mathcal{N}_i$ subsets in the form

$$\mathcal{X}_i^{(\mu)} = \{x_l^{(i,\mu)}\}_{l=1}^{\mathcal{L}_i^{(\mu)}}$$
$$\text{for } i = 1, \ldots, K \quad \text{and} \quad \mu = 1, \ldots, \mathcal{N}_i, \tag{11}$$

where $\mathcal{X}_i^{(\mu)}$ is the a-th subset of $\mathcal{X}_i$, $x_l^{(i,\mu)}$ is the l-th sample, $\mathcal{L}_i^{(\mu)}$ is the number of the samples, and $\cup_{\mu=1}^{\mathcal{N}i} \mathcal{X}_i^{(\mu)} = \mathcal{X}_i$.

After the partition of the sample sets defined by Eq. (11) , each binary subproblem $\mathcal{T}_{ij}$ is divided as follows:

$$\mathcal{T}_{ij}^{(\mu,\nu)} = \{(x_l^{(i,\mu)}, +1)\}_{l=1}^{\mathcal{L}_i^{(\mu)}} \cup \{(x_l^{(j,\nu)}, -1)\}_{l=1}^{\mathcal{L}_j^{(\nu)}}$$
$$\text{for } i, j = 1, \ldots, K, \text{ and } i \neq j$$
$$\mu = 1, \ldots, \mathcal{N}_i, \quad \nu = 1, \ldots, \mathcal{N}_j, \tag{12}$$

where the sample sets $\mathcal{X}_i^{(\mu)}$ and $\mathcal{X}_j^{(\nu)}$ are taken as positive and negative sets, respectively.

A wide range of decomposition strategies have been proposed, including random strategy, hyperplane strategy, machine learning based strategy and knowledge-based strategy. The random strategy randomly divides a training set into several subsets [13] (see Fig. 1b). This strategy is straightforward and easy to apply, but it does not make use of any statistical properties or the prior knowledge of the training data. The hyperplane strategy utilizes the geometry information of the training samples in the feature space [29]. It employs a group of parallel hyperplanes to divide the sample sets (see Fig. 1c). The machine learning based strategy adopts conventional machine learning methods into decomposition methods, such as clustering-based decomposition [30] (see Fig. 1d) and perceptron-based decompositions [31]. The knowledge-based strategy leverages the prior knowledge about the training samples. For example, to recognize gender from facial images, the training set is decomposed through the age of the persons [32]; to classify patent documents, the training set is decomposed through the published date and the subclasses [33,34]; to predict the protein subcellular location, the training set is decomposed through the gene ontology [35].

4.2. Training component classifiers

After task decomposition, learning the binary subproblems defined by Eq. (12) is independent and non-communicating tasks. Therefore, they can be solved simultaneously. Assume that the classifier learned from the subproblem $\mathcal{T}_{ij}^{(\mu,\nu)}$ is noted as $M_{ij}^{(\mu,\nu)}$.

The subproblems can be learned through identical classifiers or different kinds of classifiers. In previous researches, various classifiers such as multilayer neural networks [14], k-nearest neighbor algorithm [36] and support vector machines [37] have been employed.

4.3. Module combination

After all the binary subproblems defined by Eq. (12) have been learned, the trained classifiers are integrated through the M^3-network (Fig. 2). The principles behind are the minimization principle and the maximization principle [14].

Theorem 4.1 (*Minimization Principle*). *The classifiers* $M_{ij}^{(\mu_0,\nu)}(\nu = 1, \ldots, \mathcal{N}_j)$, *which are trained by the data sets that have the same positive training samples* $\mathcal{X}_i^{(\mu_0)}$ *as defined by Eq. (11), should be integrated through minimization.*

Proof. The resultant network can be proved to correctly classify all the training samples as follows. Suppose that the training sample x is positive, that is, $x \in \mathcal{X}_i^{(\mu_0)}$. As all the base classifiers have learned x as a positive sample, they all output high scores. Thus

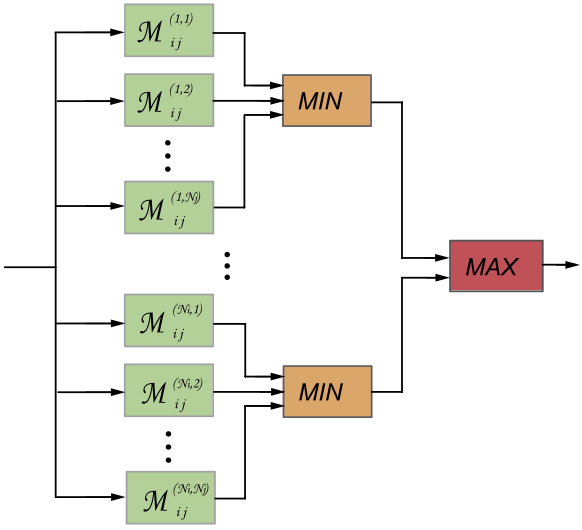


Fig. 2. Module combination of M^3 on the binary problem $\mathcal{T}_{ij}$. The MIN and MAX units represents the *minimization* function (output the minimum from the inputs) and the *maximization* function (output the maximum from the inputs), respectively.

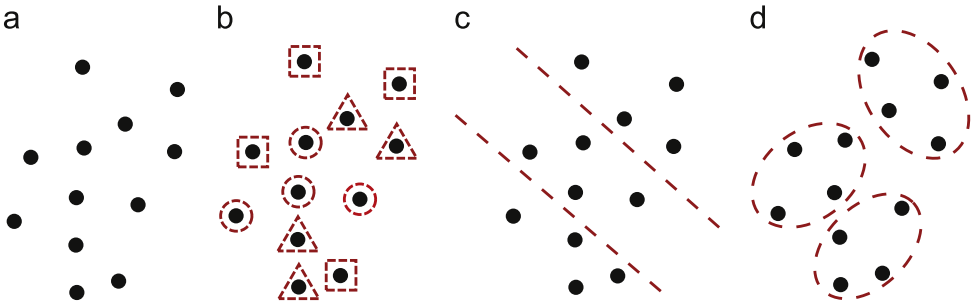


Fig. 1. Illustration of M^3 task decomposition strategies. Here 12 samples at the same class are divided into three subsets. (a) Original sample set; (b) random strategy (the shapes represent different subsets); (c) hyperplane strategy (the dotted lines represent the hyperplanes); (d) clustering-based strategy (the dotted ellipses represent the clusters).

the *minimization* also outputs a high score, which makes the network predict positive. Conversely, suppose that the training sample x is negative. Then there exists ν_0 $(1 \le \nu_0 \le \mathcal{N}_j)$, $x \in \mathcal{X}_j^{(\nu_0)}$. As the base classifier $M_{ij}^{(\mu_0,\nu_0)}$ has learned x as negative, it outputs a low score. Thus the *minimization* also outputs a low score, which makes the network predict negative. □

Theorem 4.2 (*Maximization Principle*). *The classifiers* $M_{ij}^{(\mu,\nu_0)}(\mu = 1,\ldots,\mathcal{N}_i)$, *which are trained by the data sets that have the same negative training samples* $\mathcal{X}_j^{(\nu_0)}$ *as defined by Eq. (11), should be integrated by maximization.*

proof. The resultant network can be proved to correctly classify all the training samples, similar to the proof of the Minimization Principle. For each positive training sample $x \in \mathcal{X}_i^{(\mu_0)}(1 \le \mu_0 \le \mathcal{N}_i)$, the base classifier $M_{ij}^{(\mu_0,\nu_0)}$ has learned it as positive, so it outputs a high score. Thus the *maximization* outputs a high score. Conversely, for each negative training sample $x \in \mathcal{X}_j^{(\nu_0)}$, all the base classifiers have learned it as negative, so they all output low scores. Thus the *maximization* outputs a low score. □

The module combination for the task decomposition defined by Eq. (12) can be formulated as

$$g_{ij}(x) = \max_{\mu=1}^{N_i} \min_{\nu=1}^{N_j} h_{ij}^{(\mu,\nu)}(x) \tag{13}$$

where x is a sample, $g_{ij}(x)$ is the discriminant function of the binary problem $\mathcal{T}_{ij}$, and $h_{ij}^{(\mu,\nu)}$ is the output of the module $M_{ij}^{(\mu,\nu)}$.

In addition to the above standard combination strategy, several other strategies have also been proposed. The symmetric strategy avoids the predictive bias of the standard combination strategy [38]. Fig. 3 illustrates the two combination strategies. Suppose the sample sets of the positive and negative classes are both decomposed into 5 subsets, which yields 25 modules; their outputs on a test sample are presented (1 for positive and 0 for negative). The standard combination strategy first performs *minimization* along each row, then performs *maximization* on the results of each rows (Fig. 3a). Its output can be formulated as

$$Output = \begin{cases} 1 & \text{if there exists a row of all 1} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

The symmetric combination strategy starts from the top left cell, steps right if a cell is 1 and step down otherwise recursively (Fig. 3a). The output is 1 if it ends in the left boundary, or 0 if it ends in the bottom boundary. Its output can be formulated as

$$Output = \begin{cases} 1 & \text{if there exists a row of all 1} \\ 0 & \text{if there exists a column of all 0} \\ 1 \text{ or } 0(\text{not ensured}) & \text{otherwise} \end{cases} \tag{15}$$

According to Eqs. (14) and (15), the standard strategy has a bias towards 1 while the symmetric strategy has no such a bias. Another advantage of the symmetric strategy is lower computational complexity, as it calls at most $\mathcal{N}_i + \mathcal{N}_j$ modules for each test sample. In addition to the symmetric strategy, an meta-learning based strategy which employs a decision tree algorithm to combine the modules has been explored recently [39].

5. M^3-ELM

Big data problems are usually extremely large thus the conventional ELM might become intractable. First, the training data sets are possibly too large to be fully loaded into the memories of the computation nodes in a parallel computation environment, thus the ELM models cannot be successfully learned. Second, the training procedure of ELM is not natively parallel, so it is quite time-consuming. M^3-network is a general and parallelized machine learning framework which can employ any kind of classifiers. Therefore, this paper incorporates ELM into the framework of M^3-network.

Fig. 4 illustrates the working flow of M^3-ELM. The original problem $\mathcal{T}$ is a binary problem where the circles and rectangles represent the positive and negative samples respectively, and the dotted line represents the oracle discriminant plane. The first step is to employ M^3 decomposition strategies to decompose this problem. Both the positive and negative samples are divided into two subsets, thus four subproblems are formed noted as $\mathcal{T}_{00}$–$\mathcal{T}_{11}$. The second step is to utilize ELM to learn each subproblem. The trained ELM models are noted as E_{00}–E_{11}. The third step is to employ the M^3 module combination strategy to combine the learned ELM models. The resultant network first applies MIN to (E_{00}, E_{01}) and (E_{10}, E_{11}), and then applies MAX to the outputs of the MIN units. This network is just the solution to the original problem $\mathcal{T}$.

A potential benefit of applying M^3-ELM to big data problems is that the prior knowledge and domain knowledge about the problems can be utilized. The previous researches on M^3-network have employed the prior knowledge and domain knowledge to partition the sample sets. Experimental results show that this method not only reduces the overall training time costs, but also raises the classification accuracies. Those researches include employing the information of public date and subclass to decompose the set of patent documents for patent classification [40,34], employing gene ontology to decompose the set of protein samples for protein subcellular localization [35], and employing the age information to decompose the set of facial images for gender recognition [32].

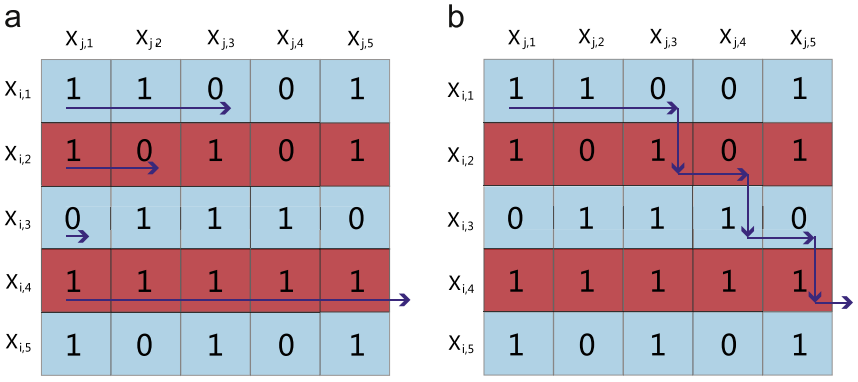


Fig. 3. Illustration of module combination strategies: (a) standard combination strategy; (b) symmetric combination strategy.

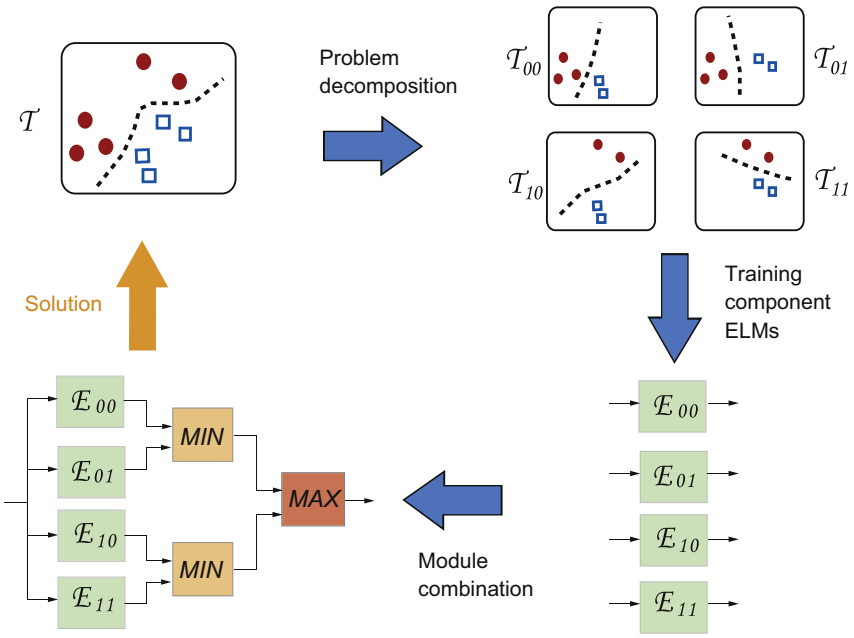


Fig. 4. Working flow of M^3-ELM. First decompose the original problem $\mathcal{T}$ into subproblems; second train a component ELM for each subproblem; third combine the ELMs through a M^3-network. The result will be a solution for the original problem.

5.1. Complexity analysis

The training procedure of ELM is formulated by Eq. (5), whose complexity can be expressed as

$$\begin{aligned} Q_{\text{train}}^{\text{ELM}} &= Q_H + Q_{H^\dagger} + Q_{H^\dagger Y} \\ &= O(NML) + O(2NL^2 + L^3) + O(NL) \\ &= O(L(M+2L+1)N + L^3) \end{aligned} \tag{16}$$

where Q_H, $Q_{H^\dagger}$ and $Q_{H^\dagger Y}$ are the complexity of producing H, the Moore–Penrose generalized inverse and the matrix multiplication respectively; N, L and M are the numbers of training samples, hidden neurons and features respectively. The test procedure of ELM is formulated by Eq. (1), whose complexity will be

$$Q_{\text{test}}^{\text{ELM}} = O(ML) \tag{17}$$

M^3-ELM is a parallel method, and each component ELM is independent thus they are trained in parallel. The time cost of training M^3-ELM is equal to the longest time of training a component ELM, so the training time complexity is

$$Q_{\text{train}}^{\text{M}^3-\text{ELM}} = O(L(M+2L+1)\tfrac{N}{\eta} + L^3) \tag{18}$$

where the number of training samples for each component ELM is assumed η times smaller than the whole training set. Therefore, the speedup will be between 1 and η, affected by L. If L is small, M^3-ELM can achieve a speed up close to η. Conversely, If L is very large, L^3 will dominate the time cost of training, and the speed up will be close to 1.

The test procedure of M^3-ELM consists of calling component ELMs and the min–max network, whose time complexity is

$$\begin{aligned} Q_{test}^{M^3-ELM} &= Q_{test}^{ELM} + Q_{min} + Q_{max} \\ &= O(ML) + O(1) + O(1) \\ &\approx O(ML). \end{aligned} \tag{19}$$

Note that the time complexity of *minimization* and *maximization* is 1 in parallel computation environment. Therefore, the test complexity of M^3-ELM is approximately equal to that of ELM.

5.2. Scalability analysis

In real-world applications, M^3-ELM can improve the scalability of the conventional ELM so as to provide us with an efficient approach to dealing with big data problems.

M^3-ELM can manipulate the size of the subproblems to fit into the memory capacity of the computation nodes in a parallel computation environment. The normal training procedure of ELM models commonly requires all the samples to be loaded into the memory. However, big data problems are usually as large as several tera or peta bytes, thus they are impossible to be fully loaded. In contrast, M^3-ELM can decompose the sample set of each class into proper size to meet the memory limits.

M^3-ELM can also manipulate the number of subproblems to fit into the number of computation nodes. The number of resultant subproblems produced by the M^3-ELM decomposition strategy is predictable. Suppose the positive and negative sample sets of a binary classification problem are divided into N_i and N_j subsets respectively, then the number of resultant subproblems will be $N_i \times N_j$.

6. Experiments

The proposed M^3-ELM is investigated through experiments in this section. The experimental settings are first described. Then the accuracies and training time costs of the conventional ELM and M^3-ELM are compared. In the end, the effects of the ELM related parameters are analyzed.

6.1. Experimental setting

6.1.1. Experimental data sets

The experimental data sets consist of a synthetic one, four relatively large benchmarks from UCI Machine Learning Repository [41], and a text categorization task derived from International Patent Classification (IPC). In this paper, we handle binary problems. Multi-class problems can be converted to binary ones through the one-versus-all or one-versus-one framework. Table 1 presents their sizes.

Table 1
Experimental data sets.

Problem	# Train	# Test	# Positive	# Negative	# Features
Two-spiral	194	40,000	20,097	20,097	2
Adult	30,162	15,060	11,208	34,014	22
Bank	30,141	15,070	5289	39,922	16
Chess	2131	1065	1669	1527	36
Internet Ads.	2186	1093	459	2820	1558
Patent(balanced)	20,000	10,000	15,000	15,000	5000
Patent(1:3[a])	20,000	10,000	7500	22,500	5000
Patent(1:9[a])	20,000	10,000	3000	27,000	5000
Patent(30,000[b])	30,000	10,000	20,000	20,000	5000
Patent(40,000[b])	40,000	10,000	25,000	25,000	5000
Patent(50,000[b])	50,000	10,000	30,000	30,000	5000
Patent(60,000[b])	60,000	10,000	35,000	35,000	5000

[a] The ratio of the positive samples versus the negative ones.
[b] The size of training sets.

The synthetic data set of two-spiral problem has been chosen as a benchmark in many machine learning studies [42,14,43]. This 2-D data set enables the working mechanism of M^3-ELM to be visualized. Fig. 6(a) presents the ground truth separating boundary of this problem. The training set of this data set is downloaded from ⟨http://www.cs.swarthmore.edu/~meeden/cs81/s06/lab02.html⟩. The test set is made by picking 200×200 uniformly distributed grid points from the region of $[0,1]\times[0,1]$ and assigning them the label of the nearest training sample.

The four UCI benchmarks used in this paper are Adult [44], Bank Marketing [45], Chess (King-Rook vs. King-Knight) [46] and Internet Advertisements [47]. They are selected from the UCI repository because they are relatively large. These data sets except the last one provide no train/test split, so the whole samples are randomly divided into training and test sets according to a widely adopted ratio of 2:1.

Patent classification is a real-world application of text categorization [48–50]. IPC is a hierarchical patent classification taxonomy created under the Strasbourg Agreement in 1971 and then administered by World Intellectual Property Organization. When patent application documents are submitted to patent offices, they are categorized into IPC classes to facilitate issuing and retrieval.

The experimental data sets are made from the Japanese patent corpus released by the project of NII Test Collection for IR Systems (NTCIR).[1] This corpus is publicly available for academic research. The experimental data sets are built by randomly picking up patent documents from the sections G (Physics) and H (Electricity) of IPC. According to our previous study, these two classes are quite difficult to distinguish [33]. The data sets with the different ratios between positive and negative samples are built to test the effects on imbalanced problems. The data sets with large number of training samples are built to test the scalability.

The Japanese texts in the data set are tokenized and stemmed by ChaSen [51,52],[2] and then function words are removed. Then the feature selection criterion of χ^2 is used to pick out the top-5000 most useful words [53,33]. In the end, the remaining words are indexed by Term Frequency-Inverse Document Frequency (TF-IDF) [53,54]. The following TFIDF formula is chosen as it achieves slightly higher classification accuracy than other variants according to our experiments:

$$\mathrm{TFIDF}(t,d)=n(t,d)\log\frac{|T|}{n_T(t)}$$

where t denotes a term, d denotes a document, T denotes the training corpus; $n(t,d)$ denotes the times that t occurs in d, namely term frequency; $n_T(t)$ denotes the number of documents where t occurs, named document frequency. The L2-norm of representation vectors is unified to 1.

6.1.2. Parameter settings

In this paper, the L2-regularized ELM is taken as the baseline method and the component classifier of M^3-ELM. The L2-regularized ELM has two parameters – the trade-off between training error and regularization noted as C, and the number of hidden neurons noted as L. The M^3-ELM has five parameters – C and L for component ELMs, the decomposition strategy $\mathcal{D}$, the number of decomposed subproblems and the combination strategy $\mathcal{C}$. This paper adopts 5-fold cross-validation on the training set and greedy algorithm to decide these parameters (except the number of the decomposed subproblems). The algorithm for M^3-ELM is presented in Fig. 5; the algorithm for ELM is a simplification of it without tuning the decomposition and combination strategies.

The knowledge-based decomposition strategy of M^3-ELM depends on the practical applications, and here it is performed only on the patent classification problems. Real-word big data problems usually have some extra information such as when, where and how the data is collected. The knowledge-based decomposition strategy is to put the samples of similar extra attributes into the same subset. For the patent classification task, each sample has subclass labels [55,49], thus the knowledge-based decomposition strategy picks up the samples that belong to the same subclass to form subsets [33,40].

$\mathbb{C}\leftarrow\{2^{-18},2^{-17},\ldots,2^{18}\}$ ▷ Candidate C
$\mathbb{L}\leftarrow\{2^{1},\ldots,2^{10}\}$ ▷ Candidate L
$\mathbb{D}\leftarrow\{random,\ hyperplane,\ cluster,\ knowledge\}$ ▷ Candi. decomposition
$\mathbb{B}\leftarrow\{standard,\ symmetric\}$ ▷ Candidate combination
$C\leftarrow2^{0},\ L\leftarrow2^{5},\mathcal{D}\leftarrow random,\mathcal{B}\leftarrow standard$ ▷ Initial values
repeat
  $C\leftarrow argmax_{C'\in\mathbb{C}}\ f_{CV}(C',L,\mathcal{D},\mathcal{B})$ ▷ Avg. acc. of 20 times cross-vali.
  $L\leftarrow argmax_{L'\in\mathbb{L}}\ f_{CV}(C,L',\mathcal{D},\mathcal{B})$
  $\mathcal{D}\leftarrow argmax_{\mathcal{D}'\in\mathbb{D}}\ f_{CV}(C,L,\mathcal{D}',\mathcal{B})$
  $\mathcal{B}\leftarrow argmax_{\mathcal{B}'\in\mathbb{B}}\ f_{CV}(C,L,\mathcal{D},\mathcal{B}')$
until No values are changed

Fig. 5. Tune the parameters of M^3-ELM through cross-validation and greedy search. Each cross-validation is performed 20 times given the random nature of ELM.

Table 2
Parameters of ELM and M^3-ELM.

Data set	ELM		M^3-ELM				
	C	L	C	L	Split	Combine	# Modules
Two-spirals	2^{10}	2^{10}	2^{10}	2^{10}	Hyperplane	Standard	3×3
Adult	2^{-5}	2^{10}	2^{-5}	2^{10}	Cluster	Standard	1×3
Bank marketing	2^{-5}	2^{10}	2^{-4}	2^{10}	Cluster	Symmetric	1×7
Chess	2^{0}	2^{10}	2^{0}	2^{10}	Hyperplane	Standard	2×2
Internet Ads.	2^{-3}	2^{10}	2^{-4}	2^{10}	Hyperplane	Standard	1×6
Patent(balanced)	2^{-7}	2^{10}	2^{-5}	2^{10}	Knowledge	Symmetric	2×2
Patent(1:3)	2^{-4}	2^{10}	2^{-4}	2^{10}	Knowledge	Standard	1×3
Patent(1:9)	2^{0}	2^{10}	2^{-5}	2^{10}	Knowledge	Standard	1×9
Patent(30,000)	2^{-6}	2^{10}	2^{-5}	2^{10}	Knowledge	Standard	3×3
Patent(40,000)	2^{-6}	2^{10}	2^{-5}	2^{10}	Knowledge	Standard	4×4
Patent(50,000)	2^{-6}	2^{10}	2^{-6}	2^{10}	Knowledge	Standard	5×5
Patent(60,000)	–[a]	–	2^{-4}	2^{10}	Knowledge	Standard	6×6

[a] Run out of memory.

[1] http://research.nii.ac.jp/ntcir/index-en.html
[2] http://chasen.naist.jp/hiki/ChaSen/

Table 3
Classification accuracies and training time costs on experimental data sets.

Data set	Accuracy		Training time (s)		Comparison	
	ELM	M^3-ELM	ELM	M^3-ELM	Error red. (%)	Speed up.
Two-spirals	0.9380 ± 0.0016[a]	**0.9394** ± 0.0012	0.56	**0.35**	2.26	1.6
Adult	0.8357 ± 0.0007	**0.8363** ± 0.0009	18.43	**9.94**	0.37	1.9
Bank marketing	0.8963 ± 0.0006	**0.8986** ± 0.0008	17.95	**6.13**	2.22	2.9
Chess	0.9918 ± 0.0024	**0.9934** ± 0.0023	5.15	**1.22**	19.51	4.2
Internet ads.	0.9758 ± 0.0021	**0.9766** ± 0.0014	2.57	**1.20**	3.31	2.1
Patent(balanced)	0.8397 ± 0.0037	**0.8540** ± 0.0028	33.42	**17.03**	8.92	2.0
Patent(1:3)	0.8640 ± 0.0029	**0.8767** ± 0.0019	33.10	**17.18**	9.34	1.9
Patent(1:9)	0.9211 ± 0.0015	**0.9315** ± 0.0013	33.15	**8.31**	13.18	4.0
Patent(30,000)	0.8460 ± 0.0017	**0.8694** ± 0.0014	49.78	**16.80**	15.19	3.0
Patent(40,000)	0.8464 ± 0.0025	**0.8715** ± 0.0018	65.82	**17.52**	16.34	3.8
Patent(50,000)	0.8467 ± 0.0026	**0.8759** ± 0.0021	97.68	**21.35**	19.05	4.6
Patent(60,000)	–[b]	**0.8794** ± 0.0022	–	**18.36**	–	–

[a] *Average* ± *standard derivation*. Each experiment is performed 10 times.
[b] Run out of memory.

The number of the decomposed subproblems for M^3-ELM in real-world applications should satisfy the following three points:

- The size of each subproblem should be small enough to be loaded into the memory of each computation node in a parallel computation environment.
- The size of each subproblem should not be too small either, which will make the number of subproblems too large.
- The positive and negative samples in each subproblem should be balanced in number so as to achieve high classification accuracy.

Therefore this paper sets the number of decomposed subproblems by first computing the ratio between the positive and negative training samples, e.g. 1:3.4; then rounding the numbers into integers which leads to 1:3 for this example; after that trying them as the number of decomposed subproblems, that is, decompose the positive samples into one subset (which is equivalent to do nothing) and decompose the negative samples into three subsets. If the size of the decomposed subproblems is larger than the memory capacity of the computational nodes, increasingly multiply the numbers, that is, trying the 2×6, 3×9, etc. The experiments of this paper are simulated on a six-cored computer with 16 G memory, thus we assume that each computation node has 2.5 G memory. The decomposition of the two-spiral problem is an exception, whose number of decomposed subproblems is set 3×3 to demonstrate the effect of M^3-ELM.

Table 2 presents the parameter setting of ELM and M^3-ELM. The optimal *C*s verify for different data sets, while the optimal *C*s of the two methods are equal or close. Large *L*s are adopted on all the data sets for both methods. As for the settings related to M^3-ELM, the hyperplane decomposition strategy performs best on Two-spirals, Chess and Internet advertisements, while the cluster strategy performs best on Adult and Bank marketing. The knowledge-based strategy outperforms the others on the patent data sets. The symmetric combination strategy is adopted on Bank marketing and Patent(balanced), while the standard combination strategy is adopted on the rest of the data sets.

6.1.3. Other settings

The experiments are carried out in MATLAB 7.10.0 environment running in an AMD six-cored 3.00-GHZ CPU with 16-GB RAM. The MATLAB code for ELM is downloaded from [24].[3] The code for M^3-ELM is implemented also in the language of MATLAB by us.

[3] http://www.ntu.edu.sg/home/egbhuang/

6.2. Performance comparison

The classification accuracy and training time costs of ELM and M^3-ELM are compared. Each experiment is performed 50 times given the random nature of ELM. Table 3 presents the results. The measurement of error reduction is computed as follows [56]:

$$ErrorReduction = 1-\frac{1-Acc_{new}}{1-Acc_{old}} \tag{20}$$

where Acc_{old} and Acc_{new} are the old and new accuracies, respectively. The experimental results show that M^3-ELM outperforms ELM on all the data sets. It reduces the test error by 0.37–19.51% and speeds up the training phrases by 1.6–4.6 times.

According to the analysis in Section 5.1, the speed up of training is roughly proportional to the size ratio of the original problems against the subproblems, noted as η at Eq. (18). For example, the data set of Patent(30,000) is decomposed into 3×3 subproblems, that is, both the positive and negative samples are split into 3 subsets and then be paired into 9 binary subproblems. Therefore each subproblem is 3 times smaller than the original problem, so the speed up should be close to 3. Most results presented in Table 3 are proportional to η while those of Two-spirals, Chess and Patent(1:9) slightly violate the relation. This is because the actually measured speed up is affected by the number of hidden neurons, and the working characteristic of the computation system.

The two-spiral problem is on a 2-D space, so it can be visualized. Fig. 6(a) presents the ground truth, and (b)–(f) presents the separating boundaries of the learned ELM and M^3-ELM models. The result that M^3-ELM models are more accurate than the ELM model can be visually verified as the area of (c), (d) and (f) matching the ground truth (a) is larger than that of (b). However, the separating boundaries of M^3-ELM have a few spikes.

In addition, Fig. 7 illustrates the derivation of the M^3-ELM model that adopts the hyperplane decomposition and standard combination strategies. The separating boundary is derived through first applying 3 MIN units to 9 component ELMs and then applying an MAX unit to the results of the MIN units.

M^3-ELM has an advantage on imbalanced problems as indicated by the results on the three patent data sets that have different positive-negative ratios. As mentioned before, M^3-ELM can transform imbalanced problems into balanced ones through the task decomposition. Table 3 shows that, as the imbalance between the numbers of positive and negative samples grows

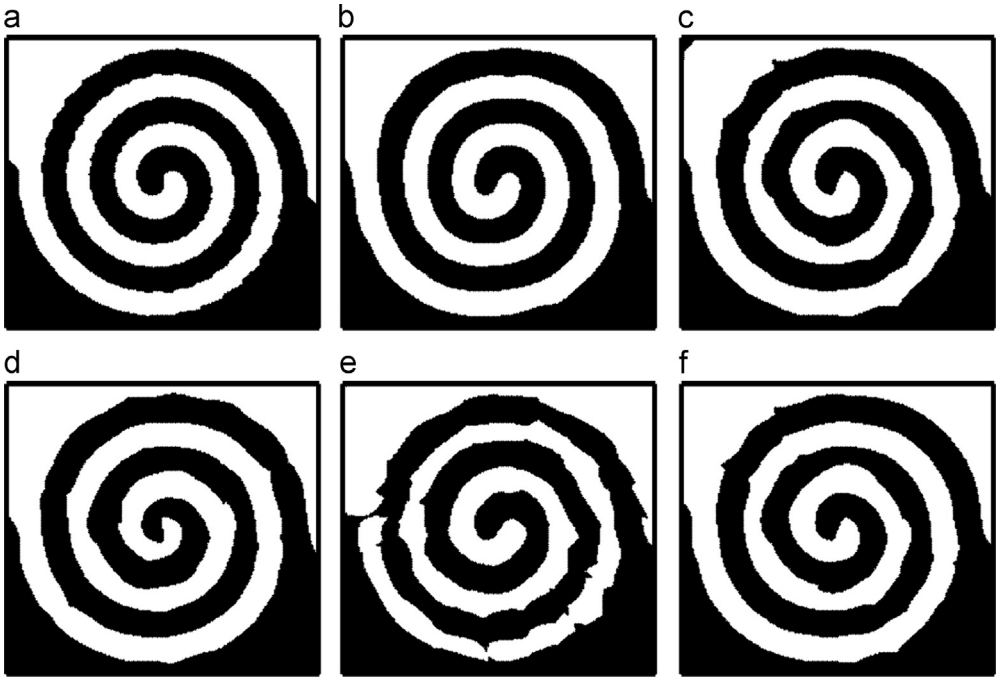


Fig. 6. Separating boundaries of the two-spiral problem: (a) ground truth; (b) ELM model (acc.: 0.9375); (c)–(e) M^3-ELM models of standard combination with hyperplane decomposition (acc.: 0.9397), clustering decomposition (acc.: 0.9383) and random decomposition (acc.: 0.9262), respectively; (f) M^3-ELM model of symmetric combination with hyperplane decomposition (acc.: 0.9396).

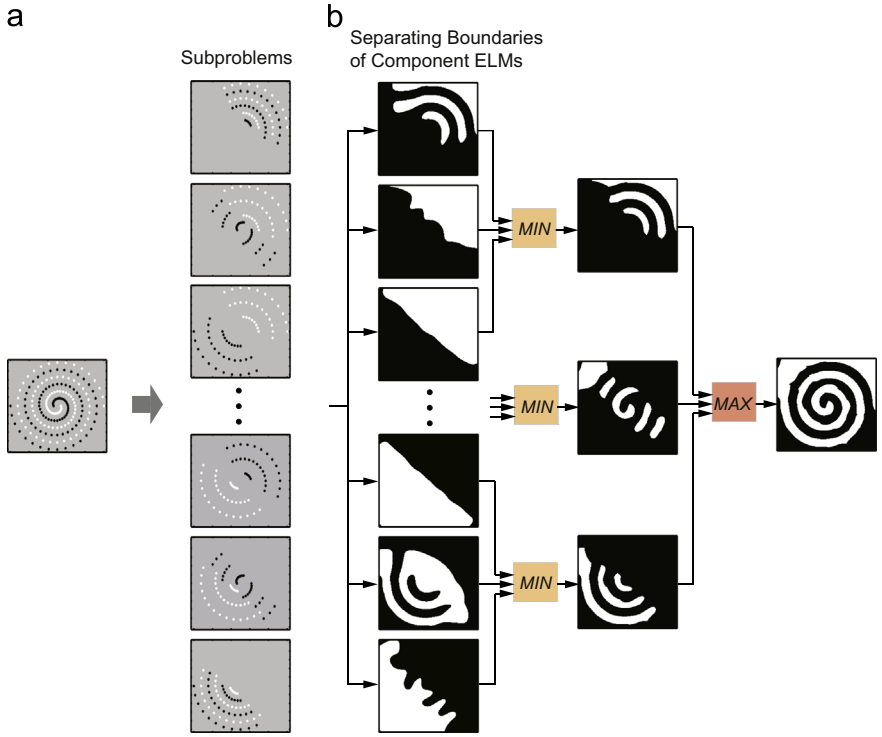


Fig. 7. Derivation of M^3-ELM model on the two-spiral problem. (a) Decompose the problem into 9 subproblems by the hyperplane decomposition strategy; (b) Learn one ELM for each subproblem and combine them by the M^3-network. The white color of the separating boundaries represents positive output while the black color represents negative output. Thus MIN shrinks white regions while MAX expands them.

from 1:1 to 1:9, the error reduction against the conventional ELM correspondingly rises from 8.92% to 13.18%.

M^3-ELM also has an advantage on large-scale data sets. Fig. 8 compares the classification accuracies and training time costs of the two methods on the Patent data sets. ELM runs out of memory when the number of training samples sets reaches 60,000. On the contrary, the memory requirement of M^3-ELM is manageable because it decomposes large classification problems into smaller

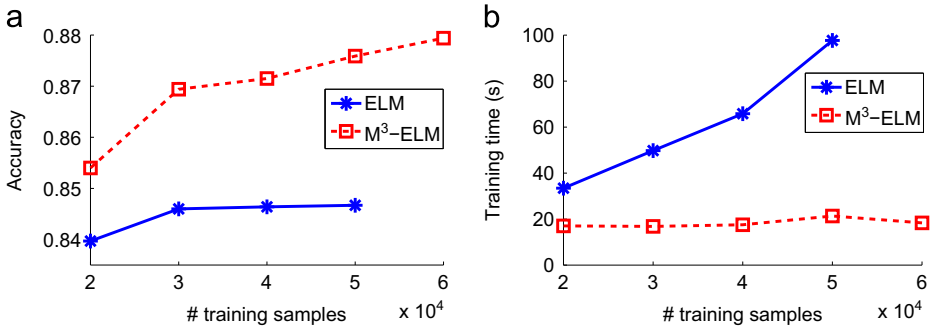


Fig. 8. Comparison of ELM and M^3-ELM on Patent data sets. (a) Classification accuracy; (b) training time costs. ELM runs out of memory under 6.0×10^4 training samples.

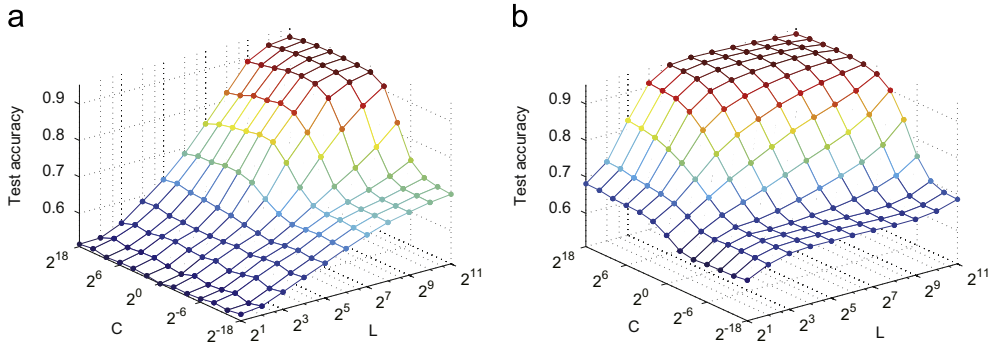


Fig. 9. Classification accuracies with different numbers of hidden neurons L and the trade-off between training error and regularization C on two-spiral data set: (a) ELM; (b) M^3-ELM with hyperplane decomposition and standard combination.

ones. The training time of M^3-ELM is determined by the longest time to train one component ELM.[4] Though the Patent data sets grow larger, yet the size of the subproblems is unchanged. Therefore the training time of M^3-ELM is almost of no growth.

6.3. Parameter analysis

ELM and M^3-ELM have two common parameters – the trade-off between training error and regularization noted as C, and the number of hidden neurons noted as L. This subsection investigates their relation with the classification accuracy. Similar to [24], Fig. 9 presents the test accuracies of different Cs and Ls on the two-spiral task. The experiments for each tuple (C, L) are performed 20 times, and the averaged accuracies are taken as the results.

The accuracies of ELM and M^3-ELM under different Cs are similar. They both can achieve high classification accuracy when Cs are larger than 2^0. Their accuracies both gradually drop when Cs decrease from 2^0 to 2^{-7}, and hit the bottom when Cs are smaller than 2^{-7}.

However, the accuracies of ELM and M^3-ELM under different Ls are different. M^3-ELM can achieve high classification accuracy using smaller Ls. It reaches the accuracy plateau when L exceeds 2^5 while ELM does not reach that until L exceeds 2^9. The reason behind is that each subproblem of M^3-ELM has fewer samples than the original problem, thus they are easier to learn. As a result, fewer hidden neurons are required.

7. Conclusions

In this paper, a parallelized ELM ensemble based on M^3-network is explored. The proposed M^3-ELM first decomposes classification problems into smaller subproblems, and then trains a component ELM for each subproblem, and in the end ensembles these ELMs with the M^3-network.

M^3-ELM improves the scalability of the conventional ELM to better solve big data problems. It can efficiently leverage the power of parallel computation environments through decomposing the original classification problems into subproblems. Both the size and the number of these subproblems are manageable thus they can fit the memory capacity and the number of the computation nodes in the parallel computation environments. In addition, the task decomposition can convert imbalanced problems into balanced ones.

The experimental results show that M^3-ELM not only reduces the training time costs but also raises the classification accuracy compared with the conventional ELM. The accuracy improvement becomes larger when the imbalance between the positive and negative samples increases. In addition, the analysis on parameter settings indicates that M^3-ELM needs less hidden nodes to achieve high classification accuracy than the conventional ELM does.

The future work of this paper is two-fold. First, we plan to release a high-performance toolkit of M^3-ELM. Our current implement is based on the MATLAB package of ELM. The new toolkit will be built on a new C/C++ package of ELM that will be released by Huang et al. soon. Second, we will test M^3-ELM in a wide range of large-scale tasks that our lab is currently working on, including text categorization, facial image-based gender classification and so on.

Acknowledgments

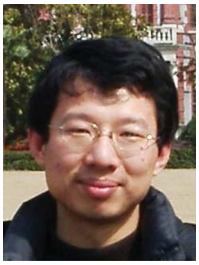
Bao-Liang Lu is supported by the National Basic Research Program of China (Grant nos. 2013CB329401 and 2009CB320901), the National Natural Science Foundation of China (Grant no. 61272248), and the Science and Technology Commission of Shanghai Municipality (Grant no. 13511500200). Hai Zhao is supported by

[4] Recent parallel computer system such as MapReduce and GPU can provide thousands of computation nodes, so here we assume that there are enough computation nodes and component ELMs are trained in parallel.

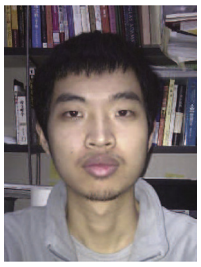
the National Natural Science Foundation of China (Grant nos. 60903119 and 61170114).

References

[1] S.J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice hall, 2010.
[2] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (3) (1999) 293–300.
[3] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[4] G.B. Huang, D.H. Wang, Y. Lan, Extreme learning machines: a survey, Int. J. Mach. Learn. Cybern. 2 (2) (2011) 107–122.
[5] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of IEEE International Joint Conference on Neural Networks, vol. 2, 2004, pp. 985–990.
[6] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1) (2006) 489–501.
[7] G.B. Huang, L. Chen, C.K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Netw. 17 (4) (2006) 879–892.
[8] R. Maclin, D. Opitz, Popular ensemble methods: an empirical study, J. Artif. Intell. Res. 11 (1999) 169–198.
[9] Z.H. Zhou, J. Wu, W. Tang, Ensembling neural networks: many could be better than all, Artif. Intell. 137 (1) (2002) 239–263.
[10] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[11] R.E. Schapire, Y. Freund, P. Bartlett, W.S. Lee, Boosting the margin: a new explanation for the effectiveness of voting methods, Ann. Stat. 26 (5) (1998) 1651–1686.
[12] G. Rätsch, T. Onoda, K.R. Müller, Soft margins for AdaBoost, Mach. Learn. 42 (3) (2001) 287–320.
[13] B.L. Lu, M. Ito, Task decomposition based on class relations: a modular neural network architecture for pattern classification, in: Lecture Notes in Computer Science: Biological and Artificial Computation: From Neuroscience to Technology, vol. 1240, 1997, pp. 330–339.
[14] B.L. Lu, M. Ito, Task decomposition and module combination based on class relations: a modular neural network for pattern classification, IEEE Trans. Neural Netw. 10 (5) (1999) 1244–1256.
[15] B.L. Lu, X.L. Wang, Y. Yang, H. Zhao, Learning from imbalanced data sets with a min–max modular support vector machine, Front. Electr. Electr. Eng. China 6 (1) (2011) 56–71.
[16] T. White, Hadoop: The Definitive Guide, O'Reilly Media, 2009, May.
[17] Z.L. Sun, T.M. Choi, K.F. Au, Y. Yu, Sales forecasting using extreme learning machine with applications in fashion retailing, Decis. Support Syst. 46 (1) (2008) 411–419.
[18] M. van Heeswijk, Y. Miche, T. Lindh-Knuutila, P. Hilbers, T. Honkela, E. Oja, A. Lendasse, Adaptive ensemble models of extreme learning machines for time series prediction, in: Lecture Notes in Computer Science: Artificial Neural Networks, vol. 5769, 2009, pp. 305–314.
[19] Y. Lan, Y.C. Soh, G.B. Huang, Ensemble of online sequential extreme learning machine, Neurocomputing 72 (13–15) (2009) 3391–3395.
[20] M. van Heeswijk, Y. Miche, E. Oja, A. Lendasse, GPU-accelerated and parallelized elm ensembles for large-scale regression, Neurocomputing 74 (16) (2011) 2430–2437.
[21] P. Escandell-Montero, J.M. Martınez-Martınez, E. Soria-Olivas, J. Guimerá-Tomás, M. Martınez-Sober, A.J. Serrano-López, Regularized committee of extreme learning machines for regression problems, in: Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Special Session on Machine Ensembles, 2012, pp. 251–256.
[22] G.B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (16–18) (2007) 3056–3062.
[23] G.B. Huang, L. Chen, Enhanced random search based incremental extreme learning machine, Neurocomputing 71 (16) (2008) 3460–3468.
[24] G.B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multi-class classification, IEEE Trans. Syst. Man Cybern. Part B 42 (2) (2012) 513–529.
[25] L.C. Shi, B.L. Lu, Eeg-based vigilance estimation using extreme learning machines, Neurocomputing 102 (2013) 135–143.
[26] W. Deng, Q. Zheng, L. Chen, Regularized extreme learning machine, in: Proceedings of IEEE Symposium on Computational Intelligence and Data Mining pp. 389–395, IEEE, 2009.
[27] W.B. Zheng, Y.T. Qian, H.J. Lu, Text categorization based on regularization extreme learning machine, Neural Comput. Appl. (2012) 1–10.
[28] G. Tsoumakas, I. Katakis, Multi-label classification: an overview, Int. J. Data Warehous. Min. 3 (3) (2007) 1–13.
[29] K.A. Wang, H. Zhao, B.L. Lu, Task decomposition using geometric relation for min–max modular SVMS, in: Lecture Notes in Computer Science: Advances in Neural Networks, vol. 3611, 2005, pp. 431–442.
[30] Y.M. Wen, B.L. Lu, H. Zhao, Equal clustering makes min–max modular support vector machine more efficient, in: Proceedings of 12th International Conference on Neural Information Processing, 2005, pp. 77–82.
[31] C. Chong, B.L. Lu, Partition of sample space with perceptrons, Comput. Simul. 25 (2) (2008) 96–99.
[32] H.C. Lian, B.L. Lu, E. Takikawa, S. Hosoi, Gender recognition using a min–max modular support vector machine, in: Lecture Notes in Computer Science: Advances in Natural Computation, vol. 3611, 2005, pp. 438–441.
[33] B.L. Lu, X.L. Wang, M. Utiyama, Incorporating prior knowledge into learning by dividing training data, Front. Comput. Sci. China 3 (1) (2009) 109–122.
[34] C. Ma, B.L. Lu, M. Utiyama, Incorporating prior knowledge into task decomposition for large-scale patent classification, in: Lecture Notes in Computer Science: Advances in Neural Networks, 2009, pp. 784–793.
[35] Y. Yang, B.L. Lu, Protein subcellular multi-localization prediction using a min-max modular support vector machine, J. Neural Syst. 20 (01) (2010) 13–28.
[36] H. Zhao, B.L. Lu, A modular k-nearest neighbor classification method for massively parallel text categorization, in: Proceedings of International Symposium on Computational and Information Science, 2004, pp. 867–872.
[37] B.L. Lu, K.A. Wang, Y.M. Wen, Comparison of parallel and cascade methods for training support vector machines on large-scale problems, in: Proceedings of International Conference on Machine Learning and Cybernetic, 2004, pp. 3056–3061.
[38] H. Zhao, B.L. Lu, Improvement on response performance of min-max modular classifier by symmetric module selection, in: Lecture Notes in Computer Science: Advances in Neural Networks, vol. 3611, 2005, pp. 39–44.
[39] Q. Kong, H. Zhao, B.L. Lu, Adaptive ensemble learning strategy using an assistant classifier for large-scale imbalanced patent categorization, in: Lecture Notes in Computer Science: Neural Information Processing. Theory and Algorithms, vol. 6443, 2010, pp. 601–608.
[40] X.L. Chu, C. Ma, J. Li, B.L. Lu, M. Utiyama, H. Isahara, Large-scale patent classification with min–max modular support vector machines, in: Proceedings of IEEE International Joint Conference on Neural Networks, 2008, pp. 3973–3980.
[41] C.L. Blake, C.J. Merz, UCI Repository of Machine Learning Databases, Department of Information and Computer Science, University of California, 1998, ⟨http://www.ics.uci.edu/∼mlearn/MLRepository.html⟩.
[42] K. Lang, M. Witbrock, Learning to tell two spirals apart, in: Proceedings of 1988 Connectionist Models Summer School, 1988, pp. 52–59.
[43] S.K. Chalup, L. Wiklendt, Variations of the two-spiral task, Connect. Sci. 19 (2) (2007) 183–199.
[44] R. Kohavi, Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, vol. 7, 1996, pp. 202–207.
[45] S. Moro, R. Laureano, P. Cortez, Using data mining for bank direct marketing: an application of the CRISP-DM methodology, in: Proceedings of the European Simulation and Modelling Conference, Eurosis, 2011, pp. 117–121.
[46] J.R. Quinlan, Learning efficient classification procedures and their application to chess end games, Mach. Learn.: Artif. Intell. Approach 1 (1983) 463–482.
[47] N. Kushmerick, Learning to remove internet advertisements, in: Proceedings of the Third Annual Conference on Autonomous Agents, ACM, 1999, pp. 175–181.
[48] L.S. Larkey, A patent search and classification system, in: Proceedings of the Fourth ACM Conference on Digital Libraries, ACM, 1999, pp. 179–187.
[49] C.J. Fall, A. Törcsvári, K. Benzineb, G. Karetka, Automated categorization in the international patent classification, ACM SIGIR Forum 37 (1) (2003) 10–25.
[50] A. Fujii, M. Iwayama, N. Kando, Introduction to the special issue on patent processing, Inf. Process. Manage. 43 (5) (2007) 1149–1153.
[51] Y. Matsumoto, A. Kitauchi, T. Yamashita, Y. Hirano, H. Matsuda, K. Takaoka, M. Asahara, Morphological Analysis System Chasen Version 2.2. 1 Manual. Nara Institute of Science and Technology, 2000.
[52] G. Jin, Q. Kong, J. Zhang, X.L. Wang, C. Hui, H. Zhao, B.L. Lu, Multiple strategies for NTCIR-08 patent mining at BCMI, in: Proceedings of the 8th NTCIR Workshop Meeting on Evaluation of Information Access Technologies, 2010, pp. 303–308.
[53] F. Sebastiani, Machine learning in automated text categorization, ACM Comput. Surv. 34 (1) (2002) 1–47.
[54] D.D. Lewis, Y. Yang, T.G. Rose, F. Li, Rcv1: a new benchmark collection for text categorization research, J. Mach. Learn. Res. 5 (2004) 361–397.
[55] C.J. Fall, K. Benzineb, Literature Survey: Issues to be Considered in the Automatic Classification of Patents ⟨http://www.wipo.int/ipc/itos4ipc/ITSupport_and_download_area/Documentation/wipo-categorizationsurvey.pdf⟩, World Intellectual Property Organization Web, 2002.
[56] K.M. Ali, M.J. Pazzani, Error reduction through learning multiple descriptions, Mach. Learn. 24 (3) (1996) 173–202.



Xiao-Lin Wang received his B.S. of computer science in 2004 from Shanghai Jiao Tong University, China. Since then he has been working for his Ph.D. at the center of Brain-like Computing and Machine Intelligence in the same university. He is the key member of the lab participating at recent evaluations PASCAL LSHTC1–3 challenge and NTCIR8–9 Patent Mining task. His research interests include machine learning, natural language processing and information retrieval.



Yang-Yang Chen received his B.S. of computer science in 2011 from Shanghai Jiao Tong University, China. Since then he has been working for M.S. at the center of Brain-like Computing and Machine Intelligence in the same university. His research interests include machine learning, artificial intelligence and information retrieval.

Hai Zhao is currently an Associate Professor in Shanghai Jiao Tong University. He obtained his B.E. in sensor and instrument, and M.Phil. in Control Theory and Engineering from Yanshan University and Ph.D. in Computer Science from Shanghai Jiao Tong University. He was a visiting researcher in the natural language computing group of Microsoft Research Asia. He was a Postdoctoral Research Fellow at City University of Hong Kong from 2006 to 2009. His research interests include machine learning, natural language processing, and data mining and artificial intelligence.

Bao-Liang Lu received the B.S. degree in Instrument and Control Engineering from Qingdao University of Science and Technology, China, in 1982, the M. S. degree in Computer Science and Engineering from Northwestern Polytechnical University, Xi'an, China, in 1989, and the Dr. Eng. degree in electrical engineering from Kyoto University, Kyoto, Japan, in 1994. After that, he became a Frontier Researcher and later a Research Scientist at the RIKEN Brain Science Institute, Wako, Japan, from 1999. He has been a full professor at the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China, since 2002, and an adjunct professor of the Laboratory for Computational Biology, Shanghai Center for Systems Biomedicine since 2005. His research interests include brain-like computing, neural networks, machine learning, pattern recognition, computer vision, brain-computer interface, natural language processing, and computational biology. He was the past President of the Asia Pacific Neural Network Assembly (APNNA) and the general Chair of the 18th International Conference on Neural Information Processing (ICONIP2011). He is an Associate Editor of the Neural Networks and a senior member of the IEEE.