Neurocomputing 149 (2015) 275–284

Contents lists available at ScienceDirect

Neurocomputing

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

# Class-specific soft voting based multiple extreme learning machines ensemble

Jingjing Cao [a,b], Sam Kwong [a,*], Ran Wang [a], Xiaodong Li [a], Ke Li [a], Xiangfei Kong [a]

[a] Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon, SAR 999077, Hong Kong
[b] School of Logistics Engineering, Wuhan University of Technology, Wuhan 430070, China

ARTICLE INFO

*Article history:*
Received 7 August 2013
Received in revised form
5 December 2013
Accepted 3 February 2014
Available online 16 September 2014

*Keywords:*
Extreme learning machine
Soft voting
Condition number
Sparse ensemble

ABSTRACT

Compared with conventional weighted voting methods, class-specific soft voting (CSSV) system has several advantages. On one hand, it not only deals with the soft class probability outputs but also refines the weights from classifiers to classes. On the other hand, the class-specific weights can be used to improve the combinative performance without increasing much computational load. This paper proposes two weight optimization based ensemble methods (CSSV-ELM and SpaCSSV-ELM) under the framework of CSSV scheme for multiple extreme learning machines (ELMs). The designed two models are in terms of accuracy and sparsity aspects, respectively. Firstly, CSSV-ELM takes advantage of the condition number of matrix, which reveals the stability of linear equation, to determine the weights of base ELM classifiers. This model can reduce the unreliability induced by randomly input parameters of a single ELM, and solve the ill-conditioned problem caused by linear system structure of ELM simultaneously. Secondly, sparse ensemble methods can lower memory requirement and speed up the classification process, but only for classifier-specific weight level. Therefore, a SpaCSSV-ELM method is proposed by transforming the weight optimization problem to a sparse coding problem, which uses the sparse representation technique for maintaining classification performance with less nonzero weight coefficients. Experiments are carried out on twenty UCI data sets and Finance event series data and the experimental results show the superior performance of the CSSV based ELM algorithms by comparing with the state-of-the-art algorithms.

© 2014 Elsevier B.V. All rights reserved.

## 1. Introduction

Extreme learning machine (ELM) becomes popular for solving classification problem due to its light computational requirements. It is an extension of the single-hidden layer feedforward networks (SLFNs). By making use of a least-square method, it analytically obtains the output weights of SLFNs [1]. Moreover, ELM emphasizes on achieving both the smallest norm of output weights and the least training error, which is different from conventional neural type of SLFNs. Essentially, ELM is originally designed by utilizing random computational nodes, which are independent of the training data. The process for tuning the hidden layer parameters is avoided, which significantly shortens the learning time. A great many of ELM based algorithms have been done in recent years [1–3].

However, since the input hidden nodes are randomly generated, it is easy to misclassify patterns that are close to the boundary [3,4]. In order to improve the classification performance, a number of real world applications based on ensemble learning have been done in previous research [2,5–7]. Different from designing a single classifier in traditional pattern recognition field, ensemble learning consists of a group of machine learning algorithms that aims at constructing multiple classifiers to form a hybrid predictive model. Generally speaking, the overall classification performance of ensemble classifier could be better than using a single classifier. The ensemble learning aims at a high accurate prediction at the expense of increased complexity. In multiple classifier system (MCS), the field of ensemble learning usually employs homogeneous base learners. In the past few decades, many ensemble techniques [8–10] are proposed to enhance the reliability of multiple models. Besides, ensemble methods are also successfully applied into applications from a wide range of fields [11–13] due to their remarkable capability in increasing the classification performance of a learning model.

Numerous works have been proposed regarding to ensemble ELMs in recent years. In [14], Liang et al. proposed the online sequential extreme learning machine (OS-ELM), which shows better generalization behavior than the other sequential algorithms. Then in [5], Lan et al. extended OS-ELM to an ensemble version and improved the stability. In [6], Liu and Wang pointed out that ELM might be prone to overfit since it approximates the training data in

* Corresponding author.
*E-mail address:* cssamk@cityu.edu.hk (S. Kwong).

http://dx.doi.org/10.1016/j.neucom.2014.02.072
0925-2312/© 2014 Elsevier B.V. All rights reserved.

learning phase. To alleviate this problem, they presented an ensemble based ELM (EN-ELM) and embedded the cross-validation into the training process. Wang and Li [7] designed a dynamic Adaboost ensemble method by using multiple ELMs with fuzzy activation function to deal with large data sets. Different from this method, Zhai et al. [2] developed the sample entropy based dynamic ensemble to handle the instability and overfitting problems of ELM. Wang et al. [4] regarded the upper integral as a base classifier and constructed an upper integral network through the learning mechanism of ELM. van Heeswijk et al. [15] introduced an adaptive ensemble models of ELMs for time series prediction. The proposed algorithm aims at performing well on both nonstationary and stationary time series. Besides, Heeswijk et al. [16] also proposed the GPU-accelerated and parallelized ELM ensemble to perform regression on large data sets.

As aforementioned, the randomly selected parameters in ELM will lead to unstable training accuracy. Therefore, Cao et al. [3] designed a voting based extreme learning machine (V-ELM) by employing ELM as base classifier under the framework of majority voting. Despite the demonstrated reliability and stability, the fact that different base classifiers have different classification performances has not been considered in their work. In ensemble learning techniques, the combinative linear classifier adopts weighted voting instead of simple voting when the accuracies of base classifiers are unequal [17]. In view of this reason, weighted fusion methods can be utilized to evaluate the confidence degree of each base classifier. The weighted voting methods mainly include the weighted majority vote schemes [18] and weighting methods [19,20] with classifier-specific weights. In previous research, the minimum square error (MSE) based method [21] was proposed as a class-specific optimal weighting approach used for linear combination of multiple classifiers. It is easy to be implemented but the weights are not optimized. Recently, Zhang and Zhou [22] proposed three new weighted combination approaches that was inspired by the idea of sparse ensembles. They employed linear programming (LP) algorithm to select classifiers and tuned their weights simultaneously. This approach made use of optimization tool to get satisfactory results. However, the weights they used are also defined on the classifier-specific level. One cannot delicately tune or optimize the weight assignment distribution on this level. Therefore, in this work, we apply the class-specific weight based soft voting for ELM classifiers (CSSV-ELM) by optimizing a class-specific weight based model. Further, since the latter steps of ELM can be regarded as solving a linear equation problem, it may suffer from the ill-conditioned problem. Thus, the condition number of the inverse of the weight matrix between hidden nodes and output nodes could be considered as part of the constraints in the optimization model. Particularly, this model takes into account both the best-worst weighted voting measure and the condition number simultaneously.

To improve computational efficiency and increase test speed, another interesting problem is how to construct sparse ensemble for class-specific soft voting scheme. Concretely, sparse ensemble aims at finding a sparse weight vector to sparsely represent the outputs of multiple classifiers. In classifier-specific weight, the sparse ensemble concept is equivalent to ensemble pruning, which aims at selecting an optimal sub-ensemble (a subset of classifiers) from a weight vector. However, in class-specific soft voting scheme, a class-based weight matrix should be determined. The pruned ensemble methods [9,22–24] are not well suited to obtain the sparse ensemble with class weight matrix. Compared with these conventional pruning methods, sparse representation based methods have been popular recently due to its flexibility to construct various optimization models based on diverse problems. The other advantage is that once the model is built by selecting an appropriate over-complete dictionary, the corresponding solution algorithm is well prepared.

Furthermore, sparse representation has shown strong relationship to classification and face recognition [25–28]. The main technique for sparse representation is sparse coding and its variants has been successfully used in face recognition. Therefore, the sparse coding (SC) techniques are applied so as to represent class weight coefficients sparsely for multiple ELMs in this paper. The problem of SC origins from sparse representations of signals and the goal is to find a linear decomposition of a signal with a few atoms of a over-complete dictionary [28]. However, the objective function for optimizing the class-specific based weights does not naturally fit to the sparse coding condition since the "dictionary matrix" is not over-complete. In face recognition field, a common way for solving this problem is to map the high-dimensional data to low-dimensional spaces by using feature extraction techniques. Thus, in this work, we apply an iterative optimization algorithm for adapting the feature extraction projection matrix $\boldsymbol{P}$ and the weight coefficients $\boldsymbol{\alpha}_k$ for each class $k$ simultaneously to exploit more robust and efficient classifier. We named the proposed model as sparse based class-specific soft voting ELM (SpaCSSV-ELM). On one hand, for fixed $\boldsymbol{\alpha}_k$, the update of $\boldsymbol{P}$ is to exploit more appropriate transformation for original conditional probability outputs of base classifiers. On the other hand, the weight $\boldsymbol{\alpha}_k$ can be refined when $\boldsymbol{P}$ is fixed. In this way, the learned $\boldsymbol{P}$ and $\boldsymbol{\alpha}_k$ can improve the robustness of sparse representation of the proposed model.

In this work, two new optimization methods based on class-specific weight are proposed for multiple ELMs with three contributions:

- The first contribution is that a convex optimization model (CSSV-ELM) is designed based on the class-specific soft voting scheme.
- The second contribution lies in constructing the constraints of optimization model (CSSV-ELM). Besides the constraint $\sum_{t=1}^{T}\alpha_t^j = 1$ and $\alpha_t^j \geq 0$ for each class, the weight constraint for each classifier is formed by combining the worst-best weighted voting and the condition number of ELM, which guarantees the importance and stability of each component ELM.
- The third contribution focuses on learning the sparse weight vector based on class-specific soft voting method with ELM as base classifier.

Overall, under the framework of class-specific weight based soft voting, two models that are related to ELM characteristic and sparse ensemble aspects, are designed to improve performance in terms of accuracy and sparsity. The rest of this paper is organized as follows. Section 2 briefly reviews ELM and some background on ensemble learning. Section 3 describes the proposed CSSV-ELM and SpaCSSV-ELM algorithms. Experimental results are shown in Section 4. Finally, Section 5 gives a conclusion.

## 2. Preliminary on extreme learning machine and related knowledge

### 2.1. Extreme learning machine

For a classification problem, we typically have a $d$-dimensional training data set with patterns that belong to one of $m$ classes each. In this paper, let the data set denoted as $\boldsymbol{z_n} = (\boldsymbol{x_n}, \boldsymbol{y_n})$, $n = 1, 2, \ldots, N$, where $\boldsymbol{x_n} \in \mathbf{R}^d, \boldsymbol{y_n} \in \mathbf{R}^m$. In neural network field, the task for supervised learning is transformed to minimize a regression cost function $\|\widehat{\mathbf{Y}} - \mathbf{Y}\|$, where $\mathbf{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N)$ is the target output matrix, and $\widehat{\mathbf{Y}} = (\widehat{\boldsymbol{y}}_1, \widehat{\boldsymbol{y}}_2, \ldots, \widehat{\boldsymbol{y}}_N)$ is the output of network with $L$ hidden nodes:

$$\widehat{\boldsymbol{y}}_n = \sum_{i=1}^{L} \boldsymbol{\beta}_i g(\boldsymbol{w}_i \cdot \boldsymbol{x}_n + b_i), \tag{1}$$

where $\mathbf{w}_i \in \mathbf{R}^d$ and $b_i \in R (i = 1, 2, \ldots, L)$ are the weight vector and threshold of the $i$th hidden node. $\boldsymbol{\beta}_i$ is the weight vector connecting to the $i$th hidden node and the output nodes, and $g(\mathbf{w}_i \cdot \mathbf{x}_n + b_i)$ is the activation function of additive nodes. Two types of activation functions: Radial basis function (RBF) and Sigmoid function are utilized in this paper and defined as follows:

RBF function:

$$g(\boldsymbol{w}_i \cdot \boldsymbol{x}_n + b_i) = \exp(-b_i \| \boldsymbol{x}_n - \boldsymbol{w}_i \|^2) \tag{2}$$

Sigmoid function:

$$g(\boldsymbol{w}_i \cdot \boldsymbol{x}_n + b_i) = \frac{1}{1 + \exp(-(\boldsymbol{w}_i \cdot \boldsymbol{x}_n + b_i))} \tag{3}$$

Equivalently, a compact format of Eq. (1) can be written as follows:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{Y}, \tag{4}$$

where $\mathbf{H} = \{g(\mathbf{w}_i \cdot \mathbf{x}_n + b_i)\}_{N \times L}$ denotes the hidden layer output matrix of neural network, where $n = 1, \ldots, N$, $i = 1, \ldots, L$. With regard to $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_L)$, its element $\boldsymbol{\beta}_i$, $i = 1, \ldots, L$ represents the weight vector connecting the $i$th hidden node and the output nodes. The unique smallest norm least squares solution of the above linear system is

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{Y} \tag{5}$$

where $\mathbf{H}^{\dagger}$ is the Moore–Penrose generalized inverse [29] of output matrix $\mathbf{H}$. Theoretically, Huang et al. [1] proposed the interpolation theorem and proved that the hidden layer parameters can be randomly generated if the activation function $g$ is infinitely differentiable in any interval. Furthermore, they also showed the universal approximation theorem [30] and proved that SLFNs with randomly generated additive or RBF nodes can universally approximate any continuous target functions over any compact subset $X \in \mathbf{R}^d$.

### 2.2. Voting, weighted voting and soft voting

In this section, we mainly introduce some basic concepts of simple voting, weighted voting and soft voting methods.

Suppose we have $T$ base classifiers $h_t$, which are trained with different variant settings. Let $f_t^{(j)}(\boldsymbol{x}_n)$ represents the output of the $t$th classifier for input $\boldsymbol{x}_n$ that belong to the $j$th class. The weighted linear combination of these $T$ base classifiers can be written as follows:

$$f_{com}^{(j)}(\boldsymbol{x}_n) = \boldsymbol{\alpha}^{\mathcal{T}} \boldsymbol{f}^{(j)}(\boldsymbol{x}_n), \tag{6}$$

where $\boldsymbol{f}^{(j)}(\boldsymbol{x}_n) = (f_1^{(j)}(\boldsymbol{x}_n), \ldots, f_T^{(j)}(\boldsymbol{x}_n))^{\mathcal{T}}$ and $\mathcal{T}$ denotes the transpose of vector or matrix. $\boldsymbol{\alpha}$ represents the weight vector of linear combination and it is independent to the class label $j$.

Thus, to determine the class label of a testing example $\boldsymbol{x}_p$ in voting system, the decision strategy is given by the rule (7). Further, the voting algorithm can be considered as a special case of the weighted majority voting with equal weights for each classifier, that is, $\boldsymbol{\alpha}^{(j)} = (1/T, \ldots, 1/T)^{\mathcal{T}}$.

$$\textit{Decide } \boldsymbol{y}_p \in c_j \quad \textit{if } f_{com}^{(j)}(\boldsymbol{x}_p) = \max_k f_{com}^{(k)}(\boldsymbol{x}_p) \tag{7}$$

Simple voting and weighted voting methods focus on dealing with the base classifier with crisp label (nominal) outputs.

• *Simple voting*: The simple voting is also called as plurality voting and it has been utilized for ELM [3]. The simple voting can be described as follows:

$$f_{com}^{(j)}(\boldsymbol{x}) = \sum_{t=1}^{T} f_t^{(j)}(\boldsymbol{x}_n), \quad f_t^{(j)}(\boldsymbol{x}_n) \in \{0, 1\},$$

where $f_t^{(j)}(\boldsymbol{x}_n) = 1$ if $h_t$ predicts $j$ as the class label and zero otherwise.

• *Weighted voting*: Simple voting method is suitable for the case that all base classifiers are with equal performance. Whereas in real practice, the base classifiers perform differently which requires the different weights to be assigned [17]. Thus, the weighted voting has been designed as follows to assign larger weights for the stronger classifier:

$$f_{com}^{(j)}(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t f_t^{(j)}(\boldsymbol{x}_n), \quad f_t^{(j)}(\boldsymbol{x}_n) \in \{0, 1\},$$

where $f_t^{(j)}(\boldsymbol{x}_n) = 1$ if $h_t$ predicts $j$ as the class label and zero otherwise.

Soft voting system is commonly employed for base classifier with class probability outputs. The system can be divided into three types of weighted soft voting methods based on different levels of weight coefficients [17]:

• *Classifier-specific weight*: In soft voting field, the weight for each base classifier is named as classifier-specific weight. Thus, the final outputs for class $c_j$ can be formed as follows:

$$f_{com}^{(j)}(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t f_t^{(j)}(\boldsymbol{x}_n), \quad f_t^{(j)}(\boldsymbol{x}_n) \in [0, 1],$$

where $\alpha_t$ is the weight for classifier $h_t$. In Zhou's work [17], he pointed out that this method is similar with weighted voting.

• *Instance-specific weight*: The instance-specific weight concerns a large number of weight coefficients which increase the computational complexity drastically.

$$f_{com}^{(j)}(\boldsymbol{x}) = \sum_{t=1}^{T} \sum_{n=1}^{N} \alpha_{tn}^{(j)} f_t^{(j)}(\boldsymbol{x}_n), \quad f_t^{(j)}(\boldsymbol{x}_n) \in [0, 1],$$

where $\alpha_{tj}^{n}$ denotes the weight for each instance $\boldsymbol{x}_n$ of the class $c_j$ for classifier $h_t$.

• *Class-specific weight*: Generally, the class-specific based weight assignment is more appropriate than the above two methods. The approach considers the weight of each class of each classifier:

$$f_{com}^{(j)}(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t^{(j)} f_t^{(j)}(\boldsymbol{x}_n), \quad f_t^{(j)}(\boldsymbol{x}_n) \in [0, 1], \tag{8}$$

where $\alpha_{tj}$ is the weight of each class for classifier $h_t$.

Since ELM learner produce class probability outputs, then the class-specific weight based soft voting can be a more appropriate method for obtaining the multiple ELM classifiers than traditional simple and weighted voting.

### 2.3. Sparse representation based classification

Suppose a signal $\boldsymbol{y}$ and a dictionary $\boldsymbol{D}$ are given, the objective of sparse representation is to select a few atoms in $\boldsymbol{D}$ and linearly combine them to represent $\boldsymbol{y}$. Defined $\boldsymbol{\alpha}$ is a coding vector and $\epsilon$ is a scalar constant, and the problem can be formulated as follows:

$$\widehat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \| \boldsymbol{\alpha} \|_0, \quad \text{s.t. } \| \boldsymbol{y} - \boldsymbol{D}\boldsymbol{\alpha} \|_2^2 \le \epsilon, \tag{9}$$

where dictionary $\boldsymbol{D}$ is commonly an over-complete matrix. However, Eq. (10) is a NP-hard problem and the solution can hardly be approximated. A common way to deal with this problem is to use $l_1$-norm to instead $l_0$-norm to get the following objective function:

$$\widehat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \{ \| \boldsymbol{y} - \boldsymbol{D}\boldsymbol{\alpha} \|_2^2 + \lambda \| \boldsymbol{\alpha} \|_1 \} \tag{10}$$

This problem can be easily solved with many efficient methods, such as Lasso [31].

The above sparse representation has been extended to face recognition field by Wright et al. [28] and named sparse representation based classification (SRC). In their work, all the training

patterns is utilized as dictionary $\boldsymbol{D}=[\boldsymbol{D}^1,\boldsymbol{D}^2,\ldots,\boldsymbol{D}^m]$, where $\boldsymbol{D}^k$ is the subset of the training patterns that belong to class $k$. The assumption for this approach is that a test pattern $\boldsymbol{x}_p$ from class $k$ can be linearly represented by the training patterns from the same class, i.e., the subspace extended by training patterns from class $k$. Similarly, they propose the model as follows:

$$\widehat{\boldsymbol{\alpha}}=\arg\min_{\boldsymbol{\alpha}}\{\|\boldsymbol{x}_p-\boldsymbol{D}\boldsymbol{\alpha}\|_2^2+\lambda\|\boldsymbol{\alpha}\|_1\}, \tag{11}$$

where $\boldsymbol{\alpha}=[\boldsymbol{\alpha}_1;\boldsymbol{\alpha}_2;\ldots;\boldsymbol{\alpha}_m]$ denotes the corresponding the sub-coding vector.

Following SRC approach, several interesting works have been proposed mainly focusing on modifying model by adapting dictionary $D$ [32,33] or feature extraction projection matrix $P$ [25,34] to get more discriminative and robust classifier. Sparse coding methods have been applied in face recognition recently, in this work, we emphasize on introducing this framework to get the sparse weight vector under the framework of CSSV system.

## 3. Proposed ensemble method for ELM

In this section, the first part interprets the concept of condition number and how this indicator can be calculated from ELM structure. The second part introduces a classifier-based weighted voting method: best-worst weighted voting, and by combining these two parts, a new constraint for the optimization model 1 is prepared. Then Section 3.3 describes the detail of model 1: CSSV-ELM. Specifically, the class-specific weight soft voting method is reformed as the optimization objective of model. The final part of this section concerns sparse representation of that weight matrix that is obtained under the framework of CSSV, then model 2: SpaCSSV-ELM is also designed by utilizing the sparse coding approach.

### 3.1. Condition number of ELM

Condition number is a metric to qualify the stability of the solution that obtained by solving a linear system in the numerical analysis. A large condition number implies that any small perturbations of data in the problem may cause large variations in the solution of the linear equation system. This kind of equation system is regarded as "ill-conditioned". Considering a linear equation $Ax=b$ where $A$ is nonsingular and $b$ is nonzero, the unique solution could be expressed as $x=A^{-1}b$. If we add a slight change $\Delta b$, suppose the corresponding solution can be $x+\Delta x$, then we have the following inequality:

$$\frac{\|\Delta x\|}{\|x\|}\le\|A\|\,\|A^{-1}\|\cdot\frac{\|\Delta b\|}{\|b\|}$$

The condition number of the matrix $A$ is defined as $cond(A)_p=\|A\|_p\|A^{-1}\|_p$ [35], where $\|\cdot\|$ denotes the $p$-norm, $p=1$, 2, or $\infty$ and $p=2$ is utilized in this paper. Noted that $A$ is originally defined as a square matrix, while for non-square matrix, the condition number still can be taken as a stability property of $A$ by calculating its pseudo inverse instead [36].

Recalling the ELM system in Section 2.1, Eq. (12) presented as follows can also be considered as a linear equation problem:

$$\mathbf{H}\boldsymbol{\beta}=\mathbf{Y}, \tag{12}$$

where $\mathbf{H}_{ni}=g(\mathbf{w}_i\cdot\mathbf{x}_n+b_i)$ denotes the element of hidden layer output matrix $\mathbf{H}$. Let the $n$_th row of $\mathbf{H}$ denoted as $\mathbf{h}_n$ and recall that $\mathbf{Y}=(\boldsymbol{y}_1,\boldsymbol{y}_2,\ldots,\boldsymbol{y}_N)$, we get the following transformed set of equations with respect to training data $\mathbf{x}_n$:

$$(\boldsymbol{\beta}^T)^{-1}\boldsymbol{y}_n^T=\boldsymbol{h}_n^T\quad n=1,\ldots,N \tag{13}$$

Through the above linear equations, it can be seen that $cond((\boldsymbol{\beta}^T)^{-1})$ can represent the sensitivity of testing label prediction with respect to the change between hidden layer output vectors that generated from train data and test data, i.e., $\Delta\boldsymbol{h}=\boldsymbol{h}_{test}^T-\boldsymbol{h}_{train}^T$. In this work, to ease the negative effect that are produced by ill-conditioned classifier, we assign larger weight to the classifier with smaller value of condition number. Thus, the condition number is used as one of the weight constraints for governing the stability of the proposed CSSV-ELM.

### 3.2. Weight constraint

In this work, we design a weight structure by taking into account the condition number of ELM and a traditional weighted voting method: best–worst weighted voting (BW-WV) [37].

BW-WV is proposed under the assumption that the weight of each classifier is proportional to its accuracy performance, which is one popular direction to assign weight to each base classifier. "best" and "worst" represent the best and worst classifiers in term of the training classification rate. The best classifier can be assigned the weight of one while the weight of the worst classifier is zero. Whereas the classifiers whose performances between them are given weights linearly in interval $(0,1)$ with ascending order. The weight evaluation can be formulated as follows:

$$\alpha_{BW}^t=\frac{\zeta_t}{\sum_{t=1}^T\zeta_t},\quad\text{where }\zeta_t=1-\frac{e_t-\min_t(e_t)}{\max_t(e_t)-\min_t(e_t)}, \tag{14}$$

where $e_t$ is the error rate of classifier $t$, and $\zeta_t\in[0,1]$. $\zeta_t=1(0)$ if the classifier has the least (most) error rate among all the classifiers. Besides, Eq. (14) also implies that the summation of all the weights is equal to 1, which represents a normalization operation.

In the previous section, we have present the condition number of ELM. Compared with any two classifiers, the classifier who has larger value of $cond((\beta^T)^{-1})$ is regarded as less stable than the other classifier. Therefore, the designed weight should satisfy that it is inversely proportion to $cond((\beta^T)^{-1})$. Furthermore, Zhou's work [17] pointed out that the voting method can well combine classifiers with equal performance while the weighted voting method can better deal with classifiers with different performances. Thus, we design a weight structure $w_t$ for model 1 (CSSV-ELM) by considering two situations as follows:

$$w_t=\begin{cases}1/T & e_i=e_j,\ \forall i,j\\ l_t\Big/\sum_{t=1}^T l_t, & \text{otherwise}\end{cases} \tag{15}$$

where $e_i$, $e_j$ represent the training error rate of the $i$th and the $j$th classifier. The weight assignment is activated if any paired classifiers have different performances. For $l_t$, it is defined by combining two factors, which can evaluate the stable and accuracy of each base classifier. The specific form of $l_t$ is introduced as follows:

$$l_t=(1+\exp(-cond((\beta^T)^{-1})))*\alpha_{BW}^t, \tag{16}$$

where $1+\exp(-(\cdot))$ is a scaling function which confine the value of the first term of Eq. (16) in the range of $[1,1+e]$. The scaling function maps the value of $cond((\beta^T)^{-1})$ to a comparable range with $\alpha_{BW}^t$.

### 3.3. Model 1: CSSV-ELM

During the process of ELM algorithm, the target class labels are transformed to a $\{+1,-1\}$ based vector. For example, a four class column vector $\widehat{\boldsymbol{y}}_n=[-1,+1,-1,-1]$ represents that the target class label for an instance $\boldsymbol{x}_n$ is class 2, since the second element of $\widehat{\boldsymbol{y}}_n$ is $+1$ while others are all $-1$. Intuitively, the goal for multiple ELMs with regard to instance $\boldsymbol{x}_n$ is to minimize the least square error $\|\boldsymbol{f}_{com}(\boldsymbol{x}_n)-\widehat{\boldsymbol{y}}_n\|_2^2$, then for all training instances, the model

should aim at minimizing the summation of all errors as follows:

$$\sum_{n=1}^{N} \| \boldsymbol{f}_{com}(\boldsymbol{x}_n) - \widehat{\boldsymbol{y}}_n \|_2^2, \tag{17}$$

where $\boldsymbol{f}_{com}(\boldsymbol{x}_n) = (f_{com}^{(1)}(\boldsymbol{x}_n), \ldots, f_{com}^{(m)}(\boldsymbol{x}_n))$ and $f_{com}^{(j)}(\boldsymbol{x}_n) = \sum_{t=1}^{T} \alpha_t^{(j)} f_t^{(j)}(\boldsymbol{x}_n)$. To form the model more conveniently, we extend this vector to a "sparse" matrix $\boldsymbol{F}_n$, where

$$\boldsymbol{F}_n(j, (t-1)*T+j) = f_t^{(j)}(\boldsymbol{x}_n)$$

The rest elements of matrix $\boldsymbol{F}_n$ is zero. Therefore, the dimension of matrix $\boldsymbol{F}_n$ is $m \times M$, where $M = m \times T$. Then, we define a column vector $\boldsymbol{\alpha}_{M\times 1} = [\alpha_1^{(1)}, \alpha_2^{(1)}, \ldots, \alpha_T^{(1)}, \alpha_1^{(2)}, \ldots, \alpha_T^{(2)}, \ldots, \alpha_1^{(m)}, \ldots, \alpha_T^{(m)}]^{\mathcal{T}}$, where $T$ and $m$ is the number of classifiers and feature. As for $\boldsymbol{F}_n$, the nonzero elements of row $j$ is corresponding to the $[\alpha_1^{(j)}, \alpha_2^{(j)}, \ldots, \alpha_T^{(j)}]$ section in $\boldsymbol{\alpha}$ in the matrix multiplication $\boldsymbol{F}_n\boldsymbol{\alpha}$.

Then we can obtain that $\boldsymbol{f}_{com}(\boldsymbol{x}_n) = \boldsymbol{F}_n\boldsymbol{\alpha}$. Therefore, the new objective function Eq. (17) can be represented as $\min_{\boldsymbol{\alpha}} \sum_{n=1}^{N} \|\boldsymbol{F}_n\boldsymbol{\alpha} - \widehat{\boldsymbol{y}}_n\|_2^2$. Next, we can easily rewritten this formula by integrating the outer summation into norm as follows:

$$\boldsymbol{\alpha} = \arg\min_{\boldsymbol{\alpha}} \| \begin{pmatrix} \widehat{\boldsymbol{y}}_1 \\ \widehat{\boldsymbol{y}}_2 \\ \vdots \\ \widehat{\boldsymbol{y}}_N \end{pmatrix} - \begin{pmatrix} \boldsymbol{F}_1 \\ \boldsymbol{F}_2 \\ \vdots \\ \boldsymbol{F}_N \end{pmatrix} \boldsymbol{\alpha} \|_2^2 \tag{18}$$

Let $\widehat{\boldsymbol{y}}_{new} = (\widehat{\boldsymbol{y}}_1^{\mathcal{T}}, \widehat{\boldsymbol{y}}_2^{\mathcal{T}}, \ldots, \widehat{\boldsymbol{y}}_N^{\mathcal{T}})^{\mathcal{T}}$, $\boldsymbol{F}_{new} = (\boldsymbol{F}_1^{\mathcal{T}}, \boldsymbol{F}_2^{\mathcal{T}}, \ldots, \boldsymbol{F}_N^{\mathcal{T}})^{\mathcal{T}}$, where the dimension of $\boldsymbol{F}_{new}$ becomes $(m \times N) \times (m \times T)$. Overall, the model 1 can be formed as follows:

$$\begin{aligned} \text{CSSV} - \text{ELM}: \quad & \min_{\boldsymbol{\alpha}} \quad \|\widehat{\boldsymbol{y}}_{new} - \boldsymbol{F}_{new}\boldsymbol{\alpha}\|_2^2 \\ \text{s.t.} \quad & \sum_{t=1}^{T} \alpha_t^{(j)} = 1 \\ & \sum_{j=1}^{m} \alpha_t^{(j)} = w_t \\ & \alpha_t^{(j)} \geq 0, \quad j = 1, \ldots, m, t = 1, \ldots, T \end{aligned} \tag{19}$$

Obviously, this model is a convex optimization problem, therefore the CVX toolbox [38,39] for matlab can be applied to get the best solution.

### 3.4. Model 2: SpaCSSV-ELM

In the sparse ensemble field, the sparse weight vector for each classification model is desired. The proposed model 1 is simple and easy to implement, however, it does not enforce the sparse factor into weight optimization process. With the notation defined in model 1, it is not nature to use sparse representation since $F_{new}$ is overdetermined when $N > T$ or $Nm > Tm$. So we firstly define several new notations to conveniently form the sparse representation model.

Suppose we have a set of dictionaries defined as $\boldsymbol{D} = [\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots, \boldsymbol{D}_m]$, where $\boldsymbol{D}_{k(i,j)} = f_j^k(x_i)(i = 1, \ldots, N, j = 1, \ldots, T)$ and $\boldsymbol{D}_k \in I\!R^{N\times T}$ is the class-specified sub-dictionary that related to class $k$. Here $m$, $N$ and $T$ are the total number of classes, instances and classifiers. Let $\boldsymbol{Y} = (\widehat{\boldsymbol{y}}_1, \widehat{\boldsymbol{y}}_2, \ldots, \widehat{\boldsymbol{y}}_N)^{\mathcal{T}} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m]$, where $\boldsymbol{y}_k = [y_1^{(k)}, y_2^{(k)}, \ldots, y_N^{(k)}]$ is the true class label of all training instance associate with class $k$. Then, a weight matrix is set as $\boldsymbol{A} = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_m]$, and $\boldsymbol{\alpha}_k = [\alpha_1^{(k)}, \alpha_2^{(k)}, \ldots, \alpha_T^{(k)}]$ is the coding coefficient of $\boldsymbol{y}_k$ over $\boldsymbol{D}_k$. According to the objective of multiple ELMs, i.e., Eq. (17), it is easy to transform the objective as follows:

$$\sum_{k=1}^{m} \|\boldsymbol{y}_k - \boldsymbol{D}_k\boldsymbol{\alpha}_k\|_2^2; \tag{20}$$

In this model, we use the Frobenius norm instead of the $l_2$-norm and the Frobenius norm of matrix $\boldsymbol{X} \in \boldsymbol{R}^{m\times n}$ is defined as follows:

$$\|\boldsymbol{X}\|_F = (\text{tr}(\boldsymbol{X}^{\mathcal{T}}\boldsymbol{X}))^{1/2} = \left( \sum_{i=1}^{m} \sum_{j=1}^{n} (\boldsymbol{X}_{ij}) \right)^{1/2} \tag{21}$$

where $\text{tr}(\boldsymbol{Z})$ is the trace of matrix $\boldsymbol{Z}$ and the Frobenius norm is the Euclidean norm of the vector obtained by listing the coefficients of the matrix.

Unlike the objective function formed in model 1, Eq. (20) focuses on optimizing the objective in terms of class, not instances. For Eq. (20), $\boldsymbol{D}_k$ is overdetermined when the number of training instances is larger than the classifiers number, i.e., $N > T$. This condition is often satisfied in real world's application. However, in the sparse coding community, the problem of computing sparse linear representations with respect to an overcomplete dictionary arises an increasing interest. One benefit way for making the matrix over-complete is to utilize feature extraction, which can project the data from higher dimensional space to lower dimensional space linearly. A projection matrix $\boldsymbol{P} \in I\!R^{d\times N}$ with $d \ll N$ is typically generated. However, such projection operator may not preserve the most useful feature information and is lack of adaptive capability. Concretely, a projection matrix $\boldsymbol{P}$ is expected to sustain the energy of a dictionary $\boldsymbol{D}$ while making the different classes $\boldsymbol{D}_k$ more separable in the subspace defined by $\boldsymbol{P}$. Next, based on the new notations defined below, the sparse dimensionality reduction method is easy to be applied in model 2.

Then, the model is equivalent to solving the following problem by adding the sparsity constraint $\|\boldsymbol{\alpha}_k\|_1 \leq \lambda$:

$$\begin{aligned} \min_{\boldsymbol{\alpha}_k, \boldsymbol{P}} \quad & \sum_{k=1}^{m} \|\boldsymbol{P}\boldsymbol{y}_k - \boldsymbol{P}\boldsymbol{D}_k\boldsymbol{\alpha}_k\|_F^2 \\ \text{s.t.} \quad & \|\boldsymbol{\alpha}_k\|_1 \leq \lambda \end{aligned} \tag{22}$$

This is a joint optimization leaning of projection $\boldsymbol{P}$ and code coefficients $\boldsymbol{\alpha}_k$ since the $\boldsymbol{y}_k$ and $\boldsymbol{D}_k$ are fixed. The model can be easily extended to the following model by requiring that $\boldsymbol{P}$ is orthogonal and class label set $\boldsymbol{Y}$ can be well reconstructed by projection $\boldsymbol{P}$:

$$\begin{aligned} \text{SpaCSSV} - \text{ELM}: \quad & \min_{\boldsymbol{\alpha}_k, \boldsymbol{P}} \quad J = \sum_{k=1}^{m} \{\|\boldsymbol{P}\boldsymbol{y}_k - \boldsymbol{P}\boldsymbol{D}_k\boldsymbol{\alpha}_k\|_F^2 + \lambda_1 \|\boldsymbol{\alpha}_k\|_1\} \\ & + \lambda_2 \|\boldsymbol{Y} - \boldsymbol{P}^{\mathcal{T}}\boldsymbol{P}\boldsymbol{Y}\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{P}\boldsymbol{P}^{\mathcal{T}} = I \end{aligned} \tag{23}$$

This is exactly the model constructed in [27] but with different sense of variables. One difference between two works is the relationship among dictionaries $\boldsymbol{D}_k$. The $\boldsymbol{D}_k$ are similar with each other in Zhang's work since each time one training sample is removed from training set. However, in the proposed model, all the $\boldsymbol{D}_k$ can differ widely since they represent the probability output of different classes for all training samples. Another difference between tow models is the reconstruction matrix. In Zhang's work, they aim to reconstruct the training data set $\boldsymbol{X}$ well, but in our work, the class label matrix $\boldsymbol{Y}$ is used to restrict projection $\boldsymbol{P}$. By optimizing $\boldsymbol{P}$ and $\boldsymbol{\alpha}_k$ alternatively, we can automatically employs the optimization algorithm in [27] as follows, and the convergence of the below algorithm can be guaranteed by the method in Zhang's work.

Overall, two models are both designed based on the framework of class-specific soft voting for combining multiple ELMs. CSSV-ELM connects the ill-condition problem of ELM to optimize the ensemble class weight coefficients, while SpaCSSV-ELM develops sparse ensemble method for weight coefficients of each class as well. Then, both CSSV-ELM and SpaCSSV-ELM can be summarized under the framework of CSSV as Algorithm 1 describes.

**Algorithm 1.** CSSV based ELMs.

Given a training set $Z = \{(\boldsymbol{x}_n, y_n) | \boldsymbol{x}_n \in \boldsymbol{R}^d, \boldsymbol{y}_n \in \boldsymbol{R}^m\}_{n=1}^{N}$, hidden node output function $g(\boldsymbol{w}_i \cdot \boldsymbol{x}_n + b_i)$, hidden node number $L$, learning iteration $T$.

**Initialization**: $t=1$
**While** $t \le T$ **do**
1. Randomly generate the learning parameters $(\mathbf{w}_i^t, b_i^t)$ $(i=1,2,\ldots,L)$ of the $t$th ELM.
2. Calculate the hidden layer output matrix $\mathbf{H}^t$
3. Calculate the output weight $\boldsymbol{\beta}^t : \boldsymbol{\beta}^t = (\mathbf{H}^t)^{\dagger}\mathbf{Y}$, where $\mathbf{Y}$ is the target output matrix.
4. $t = t+1$.
5. Calculate the $\boldsymbol{\alpha}$ for CSSV-ELM or $\boldsymbol{\alpha}_k$ of each class $k$ for SpaCSSV-ELM, $k=1,\ldots,m$.

5′. Reshape the weight vector $\boldsymbol{\alpha}$ to $\boldsymbol{\alpha}_k$ of each class $k$ for CSSV-ELM.

Final class label $\hat{y}_p$ obtained by weighted majority voting for a testing instance $\boldsymbol{x}_p$; $\hat{y}_p = \arg\max_k \boldsymbol{\alpha}_k^T \boldsymbol{f}^{(k)}(\boldsymbol{x}_p)$

## 4. Experiments

In order to gain insight into the performances of the proposed two models, which are constructed based on class-specific soft voting framework, twenty UCI data sets [40] are employed to compare the behaviors among CSSV-ELM, CSSVSpa-ELM, single ELM (S-ELM), voting ELM (V-ELM) and several weighted voting methods (MSE-ELM, LP1-ELM, LP2-ELM and LP3-ELM), which employ ELM as base classifier.

The MSE-based method [41] mainly emphasizes on calculating an optimal weight parameter $\hat{\boldsymbol{\alpha}}^{(j)}$, which can minimize the squared error (expression (17)). Particular, MSE-ELM can be considered as a class-specific based voting as well. Different from MSE-based weighted voting, Zhang et al. [22] propose LP1, LP2 and LP3 methods by employing linear programming (LP) algorithm for classifiers selection and classifier weights evaluation simultaneously. The motivation of this group of methods is that the combinative output value, $f_{com}^{(j)}(\boldsymbol{x}_n)$ should be larger than the outputs of other classes as much as possible. For LP1-ELM, LP2-ELM and LP3-ELM algorithms, the constant parameter $\varepsilon$ is chosen as 0.1 and in LP2-ELM and LP3-ELM, the penalty factor $C$ is set as 1.

Table 1 exhibits the basic information of the UCI data sets. The data sets are normalized into interval [0, 1]. The experiments also utilize two performance measures: error rate and Cohen's kappa. Error rate is the number of misclassified patterns relative to the total number of classified patterns. Cohen's kappa [42] is one of popular evaluation measures that is suitable for both binary-class and multi-class problems. Comparing with the classification rate, the Cohen's kappa score is more tolerant to randomness caused by class imbalance case. The range of Cohen's kappa score is from $-1$ to 1, which implies total disagreement and total agreement respectively, and its value is computed by the following equation:

$$\kappa = \frac{N\sum_{j=1}^{m} h_{jj} - \sum_{j=1}^{m} H_{rj}H_{cj}}{N^2 - \sum_{j=1}^{m} H_{rj}H_{cj}} \tag{24}$$

where $N$ is the number of training patterns, $h_{jj}$ is the number of true positives for each class. $H_{rj}$ and $H_{cj}$ are the sum of row $j$ and column $j$ in confusion matrix, i.e., $H_{rj} = \sum_{k=1}^{m} h_{jk}$, $H_{cj} = \sum_{k=1}^{m} h_{kj}$.

In the two model's experiments, some parameters are set as follows unless noted otherwise: the maximum number of training classifiers is assumed to be $T=7$, and the RBF function is employed as activation function of basic ELM, and the initial weight and bias parameters are drawn from a uniform distribution with ranges of $[-1, 1]$ and $[0, 1]$ respectively. Besides, the number of hidden nodes is fixed to 100 which follows the principal that it should be larger than the number of input nodes (features), and smaller than the number of instances simultaneously.

**Table 1**
Data set information.

| Data | Pattern # | Feature # | Class # | Data | Pattern # | Feature # | Class # |
|---|---|---|---|---|---|---|---|
| Australian | 690 | 14 | 2 | Iris | 150 | 4 | 3 |
| Balance | 625 | 4 | 3 | Libras | 360 | 90 | 15 |
| Breast | 699 | 9 | 2 | Pima | 768 | 8 | 2 |
| Bupa | 345 | 6 | 2 | Sonar | 208 | 60 | 2 |
| Car | 1728 | 6 | 4 | Spam | 4601 | 58 | 2 |
| Chart | 600 | 60 | 6 | Tae | 151 | 6 | 3 |
| Cmc | 1473 | 9 | 3 | Tic | 958 | 9 | 2 |
| German | 1000 | 24 | 2 | Wdbc | 569 | 30 | 2 |
| Heart | 270 | 13 | 2 | Wine | 178 | 13 | 3 |
| Ionosphere | 351 | 33 | 2 | Wpbc | 198 | 33 | 2 |

### 4.1. Experiments on model 1 (accuracy perspective)

In this subsection, we mainly focus on evaluating the effectiveness of the proposed model 1 by considering the characteristics of ELM. This simulation conducted two fold cross-validation (CV) as a trail and repeat it 100 times in MATLAB 2010b environment. Table 2 shows the statistical comparisons among the testing errors of S-ELM, variants of weighted voting ELMs and the proposed CSSV-ELM algorithm. In this table, we adopt two sorts of non-parametric tests: paired-sample sign test and paired-sample Wilcoxon signed rank test. The paired-sample sign test is used to determine whether two independent samples are from populations having the same distribution, while the paired-sample Wilcoxon signed rank test is used to examine whether or not two independent samples have the same distribution. Instead of the commonly used hypothesis $F(x)=G(y)$, the hypothesis $F(x) \ge G(y)$ is employed as the null hypothesis for both tests in this work. The $p$-values of the two tests are denoted as $Prob < S$ and $Prob < W$ respectively, and $w/t/l$ represents the wins/ties/losses records of each pair of classifiers, respectively. The sign "✓" means that at the 0.05 level, the $x$'s do significantly tends to be less than the $y$'s.

From Table 2, it is easy to conclude that CSSV-ELM algorithm gets the best performance among all weighted voting based ELM methods. More concretely, the results of the proposed algorithm are significantly better than other compared algorithms on both two statistical tests. LP3-ELM performs the worst result among all the methods and its testing accuracy significantly less than almost all other algorithms except S-ELM. Comparing the results among LP1-ELM, LP2-ELM and LP3-ELM methods, LP1-ELM and LP2-ELM can achieve the best testing accuracy and significantly better than LP3-ELM, regarding the $p$-value of the paired-sample sign test between LP1-ELM ($x$) and LP2-ELM ($y$), $Prob < S$ equals to 0.5 which implies that LP2-ELM can perform comparable to LP1-ELM. Moreover, the results that generated by MSE-ELM method is significantly superior to other algorithms except for CSSV-ELM. Table 3 shows the kappa results in test for all UCI data sets. It can be seen that the result analysis of this table is similar to Table 2.

### 4.2. Experiments on model 2 (sparsity perspective)

In this subsection, the sparsity performances of weight coefficients among MSE-ELM, LP1-ELM, LP2-ELM, LP3-ELM and SpaCSSV-ELM methods are compared in Table 4. In the training process, we select the number of classes as the value of parameter $\lambda_1$ in SpaCSSV-ELM, and parameter $\lambda_2$ is fixed as one. In this work, we define a sparsity indicator as follows:

$$Spa = \frac{\| \delta(\boldsymbol{\alpha}) < \varepsilon \|_0}{\| \boldsymbol{\alpha} \|_0} \tag{25}$$

where $\|\cdot\|$ represents the number of elements in $\boldsymbol{\alpha}$, i.e., $l_0$-norm of $\boldsymbol{\alpha}$, and $\delta(\boldsymbol{\alpha})$ extracts the zero-elements in $\boldsymbol{\alpha}$. $\varepsilon$ is a small positive

**Table 2**
Statistical significant testing error comparison among weighted voting methods.

| $x/y$ | | S-ELM | MSE-ELM | LP1-ELM | LP2-ELM | LP3-ELM | CSSV-ELM |
|---|---|---|---|---|---|---|---|
| | Mean error | 0.2565 | 0.2207 | 0.2247 | 0.2260 | 0.2398 | 0.2176 |
| S-ELM | $w/t/l$ | | 20/0/0 | 19/0/1 | 19/0/1 | 13/0/7 | 20/0/0 |
| | $Prob < S$ | | 9.5367E−7 ✓ | 2.0027E−5 ✓ | 2.0027E−5 ✓ | 0.1316 | 9.5367E−7 ✓ |
| | $Prob < W$ | | 4.7846E−5 ✓ | 1.9074E−6 ✓ | 5.5788E−5 ✓ | 0.0413 | 4.7846E−5 ✓ |
| MSE-ELM | $w/t/l$ | 0/0/20 | | 3/3/14 | 3/3/14 | 3/3/14 | 15/1/4 |
| | $Prob < S$ | 1 | | 0.9936 | 0.9936 | 0.9936 | 0.0096 ✓ |
| | $Prob < W$ | 0.9999 | | 0.9983 | 0.9977 | 0.9991 | 0.0026 ✓ |
| LP1-ELM | $w/t/l$ | 1/0/19 | 14/3/3 | | 9/3/8 | 2/3/15 | 18/0/2 |
| | $Prob < S$ | 0.9999 | 0.0063 ✓ | | 0.5 | 0.9988 | 2.0123E−4 ✓ |
| | $Prob < W$ | 0.9999 | 0.0017 ✓ | | 0.5565 | 0.9994 | 5.4422E−4 ✓ |
| LP2-ELM | $w/t/l$ | 1/0/19 | 14/3/3 | 8/3/9 | | 3/0/17 | 17/0/3 |
| | $Prob < S$ | 0.9998 | 0.0064 ✓ | 0.5 | | 0.9999 | 0.0012 ✓ |
| | $Prob < W$ | 0.9999 | 0.0023 ✓ | 0.4500 | | 0.9998 | 0.0011 ✓ |
| LP3-ELM | $w/t/l$ | 7/0/13 | 14/3/3 | 15/3/2 | 17/0/3 | | 18/0/2 |
| | $Prob < S$ | 0.8684 | 0.0064 ✓ | 0.0012 ✓ | 7.6294E−6 ✓ | | 2.0123E−4 ✓ |
| | $Prob < W$ | 0.9587 | 8.9110E−4 ✓ | 6.4321E−4 ✓ | 1.6052E−4 ✓ | | 2.7692E−4✓ |

**Table 3**
Statistical significant Cohen's kappa comparison among weighted voting methods.

| $x/y$ | | S-ELM | MSE-ELM | LP1-ELM | LP2-ELM | LP3-ELM | CSSV-ELM |
|---|---|---|---|---|---|---|---|
| | Mean error | 0.5236 | 0.5857 | 0.5783 | 0.5499 | 0.5626 | 0.5913 |
| S-ELM | $w/t/l$ | | 20/0/0 | 20/0/0 | 20/0/0 | 14/0/6 | 20/0/0 |
| | $Prob < S$ | | 9.5367E−7 ✓ | 9.5367E−7 ✓ | 9.5367E−7 ✓ | 0.0577 | 9.5367E−7 ✓ |
| | $Prob < W$ | | 4.7846E−5 ✓ | 4.7846E−5 ✓ | 4.7846E−5 ✓ | 0.0563 | 4.7846E−5 ✓ |
| MSE-ELM | $w/t/l$ | 0/0/20 | | 3/3/14 | 3/3/14 | 3/3/14 | 15/0/5 |
| | $Prob < S$ | 1 | | 0.9936 | 0.9936 | 0.9936 | 0.0207 ✓ |
| | $Prob < W$ | 1 | | 0.9990 | 0.9978 | 0.9993 | 0.0030 ✓ |
| LP1-ELM | $w/t/l$ | 0/0/20 | 14/3/3 | | 8/3/9 | 2/3/15 | 18/0/2 |
| | $Prob < S$ | 1 | 0.0064 ✓ | | 0.5 | 0.9988 | 2.0123E−4 ✓ |
| | $Prob < W$ | 0.9999 | 0.0011 ✓ | | 0.5377 | 0.9994 | 4.7668E−4 ✓ |
| LP2-ELM | $w/t/l$ | 0/0/20 | 14/3/3 | 9/3/8 | | 3/0/17 | 16/0/4 |
| | $Prob < S$ | 1 | 0.0064 ✓ | 0.5 | | 0.9999 | 0.0059✓ |
| | $Prob < W$ | 0.9999 | 0.0023 ✓ | 0.4623 | | 0.9998 | 0.0012 ✓ |
| LP3-ELM | $w/t/l$ | 6/0/14 | 14/3/3 | 15/3/2 | 17/0/3 | | 17/0/3 |
| | $Prob < S$ | 0.9423 | 0.0064 ✓ | 0.0012 ✓ | 7.6294E−6 ✓ | | 0.0013 ✓ |
| | $Prob < W$ | 0.9437 | 6.4321E−4 ✓ | 6.4321E−4 ✓ | 1.6052E−4 ✓ | | 4.1697E−4✓ |

**Table 4**
Sparsity comparison among five weighted voting based ELM methods.

| Data sets | MSE | LP1 | LP2 | LP3 | SPA | Data sets | MSE | LP1 | LP2 | LP3 | SPA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Ave.spa (%) | 2.35 | 69.80 | 85.90 | 51.55 | **87.38** | Ave.ER(%) | **18.46** | 18.93 | 18.95 | 25.38 | 19.58 |
| Australian | 0.00 | 0.87 | **0.91** | 0.48 | 0.88 | Iris | 0.05 | 0.00 | **0.98** | 0.52 | 0.77 |
| Balance | 0.00 | 0.96 | 0.94 | 0.51 | **0.98** | Libras | 0.02 | 0.75 | 0.71 | 0.50 | **0.97** |
| Breast | 0.02 | **0.96** | 0.93 | 0.49 | 0.94 | Pima | 0.02 | 0.83 | 0.85 | 0.51 | **0.87** |
| Bupa | 0.02 | 0.87 | 0.88 | 0.46 | **0.91** | Sonar | 0.02 | 0.00 | 0.80 | 0.56 | **0.90** |
| Car | 0.02 | 0.86 | 0.83 | 0.49 | **0.98** | Spam | 0.02 | 0.81 | 0.84 | 0.57 | **0.90** |
| Chart | 0.01 | 0.60 | 0.72 | 0.47 | **0.82** | Tae | 0.04 | 0.89 | 0.91 | 0.46 | **0.96** |
| Cmc | 0.04 | 0.69 | 0.77 | 0.47 | **0.81** | Tic | 0.01 | 0.69 | 0.80 | 0.50 | **0.88** |
| German | 0.02 | 0.67 | 0.81 | 0.49 | **0.89** | Wdbc | 0.02 | **0.92** | 0.91 | 0.58 | 0.42 |
| Heart | 0.02 | 0.87 | **0.89** | 0.56 | 0.88 | Wine | 0.04 | 0.00 | 0.92 | 0.61 | **0.94** |
| Ionosphere | 0.06 | 0.85 | **0.88** | 0.51 | **0.88** | Wpbc | 0.02 | 0.87 | **0.90** | 0.57 | **0.90** |

constant and is set as 0.005 in this paper. Therefore, the larger the value of *Spa*, the better the sparse ensemble weights are. The first row of Table 4 show the average of sparsity and error rate among all 20 UCI data set, respectively. It can be seen that SpaCSSV-ELM can preserve the classification performance and improve the sparsity simultaneously. The rest rows of the table exhibit the detail of *Spa* values among five methods. Concretely, the overall performance of SpaCSSV-ELM is comparable with that of LP2-ELM. MSE-ELM can achieve the least error rate but has the most dense weight coefficients. LP2-ELM performs worst with respect to error rate and the second smallest sparsity since it allows the weights to be negative. Regarding to all UCI data sets, the proposed method can obtained the best result on 15 out of 20 data sets, and failed on 5 data sets, especially for Iris and Wdbc data sets. The result on these data sets may caused by parameter setting since we fixed the sparsity control parameter $\lambda_1 = class$ for each data set.

In addition, another experiment is conducted to investigate how the classification and sparsity performances will be affected by parameter $\lambda_1$ and the number of classifiers $T$. In the training process, the value of $\lambda_1$ is selected from $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 2, 5, 10\}$ and the size of ensemble classifiers is selected from $\{10, 20, \ldots, 100\}$. Figs. 1 and 2 show the average test accuracy and average weight sparsity respectively with regard to different values of $\lambda_1$ and $T$. The colorbar on the right side of figures displays the color scale with

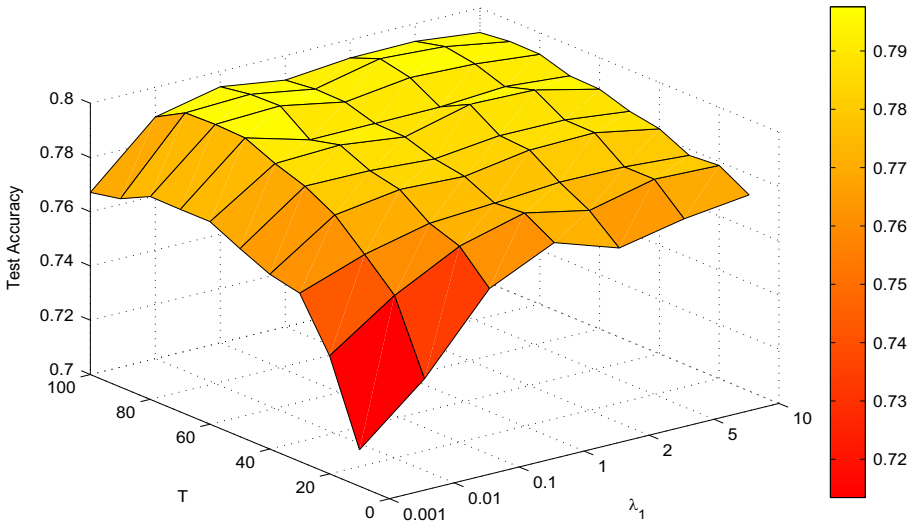



**Fig. 1.** Surface plot in term of test accuracy. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

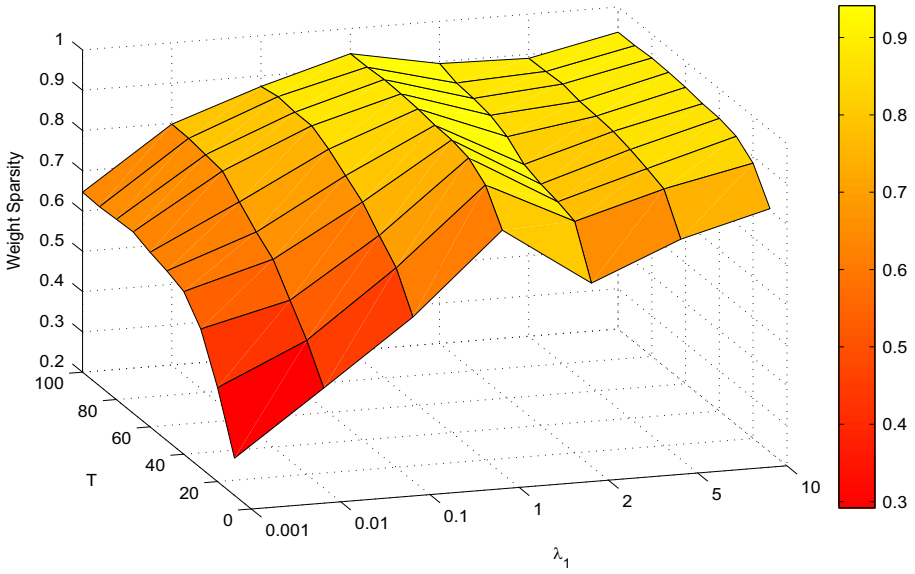



**Fig. 2.** Surface plot in term of weight sparsity. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

respect to *z*-axis of each figures, i.e., test accuracy and weight sparsity. We can see from Fig. 1 that the testing accuracy is almost proportional to $\lambda_1$ and *T*. Whereas the weighted sparsity achieve the best performance mostly when $\lambda_1 = 1$. Therefore, we believe that if $\lambda_1$ is equal to 1, the sparsity performance can even better than that $\lambda_1 = class$. But through Table 4, we can see that even $\lambda_1 = class$, SpaCSSV-ELM can outperform other four methods in term of sparsity.

### 4.3. Experiment on finance event series data

In the third part of our experiment, we address the problem of profit making in stock market [43]. Specifically, the data set is composed of 1721 patterns and 11 features. Each pattern is generated from a re-sampling in one year from market prices of 23 stocks in HKEx (Hong Kong Exchanges and Clearing Limited). All the features represent stock technical indicators and they are used to predict the market prices in the next five minutes.

The simulation is implemented by 10-fold cross-validation method with 10 trails and sigmoidal activation function is utilized. The number of classifiers and hidden nodes are both set as 20. Besides, for SpaCSSV-ELM, the parameter $\lambda_1$ is selected as the number of class ($m=2$) which controls the sparsity. Table 5 lists three evaluation criteria including mean and standard deviation of test error rate, Cohen's kappa value and sparsity of weight coefficients among S-ELM, V-ELM and the two proposed algorithms. In terms of error rate and kappa, the best result is produced by CSSV-ELM while the second best result refers to SpaCSSV-ELM. Regarding to the sparsity, the proposed sparsity of SpaCSSV-ELM is 10 times better than that of CSSV-ELM algorithm. As for V-ELM, it can be treated as all the weights are equal. Therefore the sparsity of V-ELM is zero which denotes that all the weights are non-zero. It should be noted that V-ELM has the most stable performance since it has the smallest deviation among all four methods.

### 4.4. Discussions

In the above experiment results, we have shown that the proposed two CSSV-based approaches can outperform the other benchmark methods in terms of accuracy and sparsity perspective, respectively. In the following part, we analyze and discuss the main reasons for such an outcome from three aspects. Firstly, for

**Table 5**
Error rates, kappa and sparsity values for different ELM methods on a finance event series data.

| Method | S-ELM | V-ELM | CSSV-ELM | SpaCSSV-ELM |
|---|---|---|---|---|
| Error rate | 45.98 ±0.0129 | 44.76 ±0.0073 | **43.75 ±0.0077** | 44.08 ±0.0122 |
| Kappa | 10.03 ±0.0160 | 10.44 ±0.0147 | **11.34 ±0.0155** | 10.95 ±0.0227 |
| Sparsity | 0 | 0 | 0.0735 | **0.7035** |

CSSV-based models, the class-specific soft voting scheme provides a much larger search space than classifier-based weighted voting methods do. This characteristic makes the designed approaches can search more appropriate weight coefficients. Additionally, traditional classifier-based weighted voting scheme can be regarded as one special case of CSSV scheme if all the class weights of a classifier are the same. Secondly, a classifier-based weighted voting method (BW-WV) and the condition number of ELM are both utilized to determine the summation weights of each class for one classifier. This design not only reserves the conventional classifier weight assignment strategy, but also avoid the "ill-condition" problem of linear equation system that ELM may suffer. Thirdly, as for SpaCSSV-ELM, the reason why it derives great sparsity performance is mainly due to the effect of the sparse coding algorithm [27]. However, the testing error rate for SpaCSSV-ELM is worse than most compared methods. It implies that the SpaCSSV-ELM may sacrifice the accuracy performance in some degree for sparsity improvement. With regard to real-world stock market price application, the performance of two proposed models are similar with the one on UCI data sets. Therefore, we can believe that the models have a certain degree of flexibility for different data sets.

## 5. Conclusion

In this work, we discussed the weight optimization issues based on class-specific soft voting for combining multiple ELMs. The first model not only deals with the ill-conditioned problem of ELM, but also integrates the traditional weighted voting schemes. Whereas the second model considers the sparsity issue of weight coefficients and retains the classification performance at the same time. Experimental results show that the proposed model 1 is statistically superior to all the compared methods, including S-ELM and weighted voting based ELM. As for SpaCSSV-ELM, it can efficiently reduce the non-zero weight coefficients and sustain the classification performance simultaneously. Further, both proposed algorithms are conducted on a finance data set and achieve a better performance than S-ELM and V-ELM.

## Acknowledgements

The work described in this paper was partially supported by the Fundamental Research Funds for the Central Universities (WUT:2014-IV-054), National Natural Science Foundation of China under the Grant No. 61175123, City University Applied Research Grant 9667094 and Chinese National Science Foundation (Grant No. 61272289).

## References

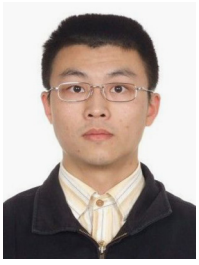
[1] G. Huang, Q. Zhu, C. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1) (2006) 489–501.
[2] J. Zhai, H. Xu, X. Wang, Dynamic ensemble extreme learning machine based on sample entropy, Soft Comput. 16 (9) (2012) 1493–1502.
[3] J. Cao, Z. Lin, G. Huang, N. Liu, Voting based extreme learning machine, Inf. Sci. 185 (1) (2011) 66–77.
[4] X. Wang, A. Chen, H. Feng, Upper integral network with extreme learning mechanism, Neurocomputing 74 (16) (2011) 2520–2525.
[5] Y. Lan, Y. Soh, G. Huang, Ensemble of online sequential extreme learning machine, Neurocomputing 72 (13-15) (2009) 3391–3395.
[6] N. Liu, H. Wang, Ensemble based extreme learning machine, IEEE Signal Process. Lett. 17 (8) (2010) 754–757.
[7] G. Wang, P. Li, Dynamic adaboost ensemble extreme learning machine, in: 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), vol. 3, 2010, pp. V3–54.
[8] S. Avidan, Ensemble tracking, IEEE Trans. Pattern Anal. Mach. Intell. 29 (2) (2007) 261–271.
[9] G. Martínez-Muñoz, D. Hernández-Lobato, A. Suárez, An analysis of ensemble pruning techniques based on ordered aggregation, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2) (2009) 245–259.
[10] T. Windeatt, Accuracy/diversity and ensemble MLP classifier design, IEEE Trans. Neural Netw. 17 (5) (2006) 1194–1211.
[11] H. Masnadi-Shirazi, N. Vasconcelos, Cost-sensitive boosting, IEEE Trans. Pattern Anal. Mach. Intell. 33 (2) (2011) 294–309.
[12] P.K. Mallapragada, R. Jin, A.K. Jain, Y. Liu, Semiboost: boosting for semi-supervised learning, IEEE Trans. Pattern Anal. Mach. Intell. 31 (11) (2009) 2000–2014.
[13] W. Hu, W. Hu, S.J. Maybank, Adaboost-based algorithm for network intrusion detection, IEEE Trans. Syst. Man Cybern. Part B 38 (2) (2008) 577–583.
[14] N. Liang, G. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Trans. Neural Netw. 17 (6) (2006) 1411–1423.
[15] M. van Heeswijk, Y. Miche, T. Lindh-Knuutila, P.A.J. Hilbers, T. Honkela, E. Oja, A. Lendasse, Adaptive ensemble models of extreme learning machines for time series prediction, in: ICANN (2), 2009, pp. 305–314.
[16] 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.
[17] Z. Zhou, Ensemble Methods: Foundations and Algorithms, Chapman & Hall, BocaRaton, FL, USA, 2012.
[18] L. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley, New York, 2004.
[19] D. Opitz, J. Shavlik, et al., Generating accurate and diverse members of a neural-network ensemble, Adv. Neural Inf. Process. Syst. (1996) 535–541.
[20] F. Moreno-Seco, J. Iñesta, P. de León, L. Micó, Comparison of classifier fusion methods for classification in pattern recognition tasks, Struc. Synt. Stat. Pattern Recognit. (2006) 705–713.
[21] J.A. Benediktsson, J.R. Sveinsson, O.K. Ersoy, P.H. Swain, Parallel consensual neural networks, IEEE Trans. Neural Netw. 8 (1) (1997) 54–64.
[22] L. Zhang, W. Zhou, Sparse ensembles using weighted combination methods based on linear programming, Pattern Recognit. 44 (1) (2011) 97–106.
[23] A. Grove, D. Schuurmans, Boosting in the limit: maximizing the margin of learned ensembles, in: Proceedings of the National Conference on Artificial Intelligence, 1998, pp. 692–699.
[24] H. Chen, P. Tiňo, X. Yao, A probabilistic ensemble pruning algorithm, in: ICDM Workshops, 2006, pp. 878–882.
[25] H. Zhang, Y. Zhang, T.S. Huang, Simultaneous discriminative projection and dictionary learning for sparse representation based classification, Pattern Recognit. 46 (1) (2013) 346–354.
[26] H. Zhang, N.M. Nasrabadi, Y. Zhang, T.S. Huang, Joint dynamic sparse representation for multi-view face recognition, Pattern Recognit. 45 (4) (2012) 1290–1298.
[27] L. Zhang, M. Yang, Z. Feng, D. Zhang, On the dimensionality reduction for sparse representation based face recognition, in: The 20th International Conference on Pattern Recognition, ICPR 2010, 2010, pp. 1237–1240.
[28] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, Y. Ma, Robust face recognition via sparse representation, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2) (2009) 210–227.
[29] D. Serre, Matrices: Theory and Applications, 2010.
[30] G. 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.
[31] T. Hastie, R. Tibshirani, J.J.H. Friedman, The Elements of Statistical Learning, vol. 1, Springer, New York, 2001.
[32] M. Yang, L. Zhang, X. Feng, D. Zhang, Fisher discrimination dictionary learning for sparse representation, in: IEEE International Conference on Computer Vision, ICCV 2011, 2011, pp. 543–550.
[33] Z. Jiang, Z. Lin, L.S. Davis, Learning a discriminative dictionary for sparse coding via label consistent k-svd, in: The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, 2011, pp. 1697–1704.
[34] D. Pham, S. Venkatesh, Joint learning and dictionary construction for pattern recognition, in: 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2008, 2008, pp. 1–8.

[35] D.S. Watkins, Fundamentals of Matrix Computations, vol. 64, John Wiley & Sons, New York, 2004.
[36] R. Hagen, S. Roch, B. Silbermann, C*-Algebras and Numerical Analysis, vol. 236, CRC Press LLC, BocaRaton, FL, USA, 2001.
[37] L. Rokach, Pattern Classification Using Ensemble Methods, vol. 75, World Scientific Publishing Company, Singapore, 2010.
[38] M. Grant, S. Boyd, CVX: Matlab software for disciplined convex programming, version 2.0 beta. ⟨http://cvxr.com/cvx⟩, Sept. 2012.
[39] M. Grant, S. Boyd, Graph implementations for nonsmooth convex programs, in: V. Blondel, S. Boyd, H. Kimura (Eds.), Recent Advances in Learning and Control, Lecture Notes in Control and Information Sciences, Springer-Verlag Limited, Berlin, Germany, 2008, pp. 95–110 ⟨http://stanford.edu/~boyd/graph_dcp.html⟩.
[40] A. Frank, A. Asuncion, UCI Machine Learning Repository, 2010.
[41] N. Ueda, Optimal linear combination of neural networks for improving classification performance, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2) (2000) 207–215.
[42] B. Arie, A lot of randomness is hiding in accuracy, Eng. Appl. Artif. Intell. 20 (7) (2007) 875–885.
[43] X. Li, C. Wang, J. Dong, F. Wang, X. Deng, S. Zhu, Improving stock market prediction by integrating both market news and stock prices, in: Database and Expert Systems Applications, Springer, Berlin, Germany, 2011, pp. 279–293.

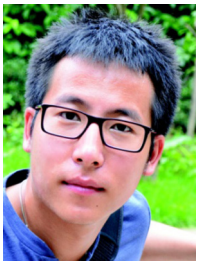
**Jingjing Cao** received her B.Sc. degree in Information and Computing Science, Dalian Maritime University, China in 2006 and the M.Sc. degree in Applied Mathematics from the same university in 2008. She got her Ph.D. degree in the Department of Computer Science in City University of Hong Kong, Hong Kong. She joined the Wuhan University of Technology, China, in 2013, where she is currently a lecturer in the School of Logistics Engineering. Her research interests are in ensemble learning, extreme learning machine, evolutionary algorithms, and their applications.

**Sam Kwong** received his B.Sc. degree and M.A.Sc. degree in electrical engineering from the State University of New York at Buffalo, USA and University of Waterloo, Canada, in 1983 and 1985 respectively. In 1996, he later obtained his Ph.D. from the University of Hagen, Germany. From 1985 to 1987, he was a diagnostic engineer with the Control Data Canada where he designed the diagnostic software to detect the manufacture faults of the VLSI chips in the Cyber 430 machine. He later joined the Bell Northern Research Canada as a Member of Scientific staff. In 1990, he joined the City University of Hong Kong as a lecturer in the Department of Electronic Engineering. He is currently an associate Professor in the department of computer Science.

**Ran Wang** (S'09) received the Bachelors degree from the College of Information Science and Technology, Beijing Forestry University, Beijing, China, in 2009. She is currently pursuing the Ph.D. degree at the Department of Computer Science, City University of Hong Kong, Hong Kong. Her current research interests include support vector machines, extreme learning machines, decision tree induction, active learning, multiclass classification, and the related applications of machine learning.



**Xiaodong Li** received the Bachelor degree from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2006. He is currently pursuing the Ph.D. degree at the Department of Computer Science, City University of Hong Kong, Hong Kong. His current research interests include support vector machines, boosting, market micro structure and algorithmic trading.



**Ke Li** was born in Hunan, China, in 1985. He received the B.Sc. and M.Sc. degrees in computer science and technology from the Xiangtan University, China, in 2007 and 2010, respectively. He is current pursuing the Ph.D. degree at City University of Hong Kong. His current research interests include the evolutionary multi-objective optimization, surrogate-assisted evolutionary algorithms and statistical machine learning techniques.

**Xiangfei Kong** received his Bachelor's degree at Shandong University of Science and Technology in 2009. He is a Ph.D. student of Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong. His research interests include pattern recognition, image denoising, and computer vision.