Extreme Learning Machine for Regression and Multiclass Classification

Guang-Bin Huang, *Senior Member, IEEE*, Hongming Zhou, Xiaojian Ding, and Rui Zhang

***Abstract*—Due to the simplicity of their implementations, least square support vector machine (LS-SVM) and proximal support vector machine (PSVM) have been widely used in binary classification applications. The conventional LS-SVM and PSVM cannot be used in regression and multiclass classification applications directly, although variants of LS-SVM and PSVM have been proposed to handle such cases. This paper shows that both LS-SVM and PSVM can be simplified further and a unified learning framework of LS-SVM, PSVM, and other regularization algorithms referred to extreme learning machine (ELM) can be built. ELM works for the "generalized" single-hidden-layer feedforward networks (SLFNs), but the hidden layer (or called feature mapping) in ELM need not be tuned. Such SLFNs include but are not limited to SVM, polynomial network, and the conventional feedforward neural networks. This paper shows the following: 1) ELM provides a unified learning platform with a widespread type of feature mappings and can be applied in regression and multiclass classification applications directly; 2) from the optimization method point of view, ELM has milder optimization constraints compared to LS-SVM and PSVM; 3) in theory, compared to ELM, LS-SVM and PSVM achieve suboptimal solutions and require higher computational complexity; and 4) in theory, ELM can approximate any target continuous function and classify any disjoint regions. As verified by the simulation results, ELM tends to have better scalability and achieve similar (for *regression* and *binary class* cases) or much better (for *multiclass* cases) generalization performance at much faster learning speed (up to *thousands* times) than traditional SVM and LS-SVM.**

***Index Terms*—Extreme learning machine (ELM), feature mapping, kernel, least square support vector machine (LS-SVM), proximal support vector machine (PSVM), regularization network.**

I. Introduction

In THE PAST two decades, due to their surprising classification capability, support vector machine (SVM) [1] and its variants [2]–[4] have been extensively used in classification applications. SVM has two main learning features: 1) In SVM, the training data are first mapped into a higher dimensional feature space through a nonlinear feature mapping function $\phi(\mathbf{x})$, and 2) the standard optimization method is then used to find the solution of maximizing the separating margin of two different classes in this feature space while minimizing the training errors. With the introduction of the epsilon-insensitive loss function, the support vector method has been extended to solve regression problems [5].

As the training of SVMs involves a quadratic programming problem, the computational complexity of SVM training algorithms is usually intensive, which is at least quadratic with respect to the number of training examples. It is difficult to deal with large problems using single traditional SVMs [6]; instead, SVM mixtures can be used in large applications [6], [7].

Least square SVM (LS-SVM) [2] and proximal SVM (PSVM) [3] provide fast implementations of the traditional SVM. Both LS-SVM and PSVM use equality optimization constraints instead of inequalities from the traditional SVM, which results in a direct least square solution by avoiding quadratic programming.

SVM, LS-SVM, and PSVM are originally proposed for binary classification. Different methods have been proposed in order for them to be applied in multiclass classification problems. One-against-all (OAA) and one-against-one (OAO) methods are mainly used in the implementation of SVM in multiclass classification applications [8]. OAA-SVM consists of m SVMs, where m is the number of classes. The ith SVM is trained with all of the samples in the ith class with positive labels and all the other examples from the remaining $m-1$ classes with negative labels. OAO-SVM consists of $m(m-1)/2$ SVMs, where each is trained with the samples from two classes only. Some encoding schemes such as minimal output coding (MOC) [9] and Bayesian coding–decoding schemes [10] have been proposed to solve multiclass problems with LS-SVM. Each class is represented by a unique binary output codeword of m bits. m outputs are used in MOC-LS-SVM in order to scale up to 2^m classes [9]. Bayes' rule-based LS-SVM uses m binary LS-SVM plug-in classifiers with its binary class probabilities inferred in a second step within the related probabilistic framework [10]. With the prior multiclass probabilities and the posterior binary class probabilities, Bayes' rule is then applied m times to infer posterior multiclass probabilities [10]. Bayes' rule and different coding scheme are used in PSVM for multiclass problems [11].

The decision functions of binary SVM, LS-SVM, and PSVM classifiers have the same form

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{N} \alpha_i t_i K(\mathbf{x}, \mathbf{x}_i) + b\right) \tag{1}$$

Manuscript received August 22, 2011; accepted September 4, 2011. Date of publication October 6, 2011; date of current version March 16, 2012. This work was supported by a grant from Singapore Academic Research Fund (AcRF) Tier 1 under Project RG 22/08 (M52040128) and also a grant from China National Natural Science Foundation under Project 61075050. This paper was recommended by Editor E. Santos, Jr.

G.-B. Huang and H. Zhou are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: egbhuang@ntu.edu.sg; hmzhou@ntu.edu.sg).

R. Zhang is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, and also with the Department of Mathematics, Northwest University, Xi'an, China 710069 (e-mail: zh0010ui@ntu.edu.sg).

X. Ding is with the School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China 710049 (e-mail: xjding@xjtu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2011.2168604

where t_i is the corresponding target class label of the training data $\mathbf{x}_i$, α_i is the Lagrange multiplier to be computed by the learning machines, and $K(\mathbf{u}, \mathbf{v})$ is a suitable kernel function to be given by users. From the network architecture point of view, SVM, LS-SVM, and PSVM can be considered as a specific type of single-hidden-layer feedforward network (SLFN) (the so-called support vector network termed by Cortes and Vapnik [1]) where the output of the ith hidden node is $K(\mathbf{x}, \mathbf{x}_i)$ and the output weight linking the ith hidden node to the output node is $\alpha_i t_i$. The term bias b plays an important role in SVM, LS-SVM, and PSVM, which produces the equality optimization constraints in the dual optimization problems of these methods. For example, the only difference between LS-SVM and PSVM is on how to use the bias b in the optimization formula while they have the same optimization constraint, resulting in different least square solutions. No learning parameter in the hidden-layer output function (kernel) $K(\mathbf{u}, \mathbf{v})$ needs to be tuned by SVM, LS-SVM, and PSVM, although some user-specified parameter needs to be chosen *a priori*.

Extreme learning machine (ELM) [12]–[16] studies a much wider type of "generalized" SLFNs whose hidden layer need not be tuned. ELM has been attracting the attentions from more and more researchers [17]–[22]. ELM was originally developed for the single-hidden-layer feedforward neural networks [12]–[14] and then extended to the "generalized" SLFNs which may not be neuron alike [15], [16]

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} \tag{2}$$

where $\mathbf{h}(\mathbf{x})$ is the hidden-layer output corresponding to the input sample $\mathbf{x}$ and $\boldsymbol{\beta}$ is the output weight vector between the hidden layer and the output layer. One of the salient features of ELM is that the hidden layer need not be tuned. Essentially, ELM originally proposes to apply random computational nodes in the hidden layer, which are independent of the training data. Different from traditional learning algorithms for a neural type of SLFNs [23], ELM aims to reach not only the smallest training error but also the smallest norm of output weights. ELM [12], [13] and its variants [14]–[16], [24]–[28] mainly focus on the *regression* applications. Latest development of ELM has shown some relationships between ELM and SVM [18], [19], [29].

Suykens and Vandewalle [30] described a training method for SLFNs which applies the hidden-layer output mapping as the feature mapping of SVM. However, the hidden-layer parameters need to be iteratively computed by solving an optimization problem (refer to the last paragraph in Section IV-A1 for details). As Suykens and Vandewalle stated in their work (see [30, p. 907]), the drawbacks of this method are the following: the high computational cost and larger number of parameters in the hidden layer. Liu *et al.* [18] and Frénay and Verleysen [19] show that the ELM learning approach can be applied to SVMs directly by simply replacing SVM kernels with (random) ELM kernels and better generalization can be achieved. Different from the study of Suykens and Vandewalle [30] in which the hidden layer is parametric, the ELM hidden layer used in the studies of Liu *et al.* [18] and Frénay and Verleysen [19] is nonparametric, and the hidden-layer parameters need not be tuned and can be fixed once randomly generated. Liu *et al.* [18] suggest to apply ELM kernel in SVMs and particularly study PSVM [3] with ELM kernel. Later, Frénay and Verleysen [19] show that the normalized ELM kernel can also be applied in the traditional SVM. Their proposed SVM with ELM kernel and the conventional SVM have the same optimization constraints (e.g., both inequality constraints and bias b are used). Recently, Huang *et al.* [29] further show the following: 1) SVM's maximal separating margin property and the ELM's minimal norm of output weight property are actually consistent, and with ELM framework, SVM's maximal separating margin property and Barlett's theory on feedforward neural networks remain consistent, and 2) compared to SVM, ELM requires fewer optimization constraints and results in simpler implementation, faster learning, and better generalization performance. However, similar to SVM, inequality optimization constraints are used in [29]. Huang *et al.* [29] use random kernels and discard the term bias b used in the conventional SVM. However, no direct relationship has so far been found between the original ELM implementation [12]–[16] and LS-SVM/PSVM. *Whether feedforward neural networks, SVM, LS-SVM, and PSVM can be unified still remains open.*

Different from the studies of Huang *et al.* [29], Liu *et al.* [18], and Frénay and Verleysen [19], this paper extends ELM to LS-SVM and PSVM and provides a unified solution for LS-SVM and PSVM under equality constraints. In particular, the following contributions have been made in this paper.

1) ELM was originally developed from feedforward neural networks [12]–[16]. Different from other ELM work in literature, this paper manages to extend ELM to kernel learning: It is shown that ELM can use a wide type of feature mappings (hidden-layer output functions), including *random hidden nodes* and *kernels*. With this extension, the unified ELM solution can be obtained for feedforward neural networks, RBF network, LS-SVM, and PSVM.
2) Furthermore, ELM, which is with higher scalability and less computational complexity, not only unifies different popular learning algorithms but also provides a unified solution to different practical applications (e.g., regression, binary, and multiclass classifications). Different variants of LS-SVM and SVM are required for different types of applications. ELM avoids such trivial and tedious situations faced by LS-SVM and SVM. In ELM method, all these applications can be resolved in one formula.
3) From the optimization method point of view, ELM and LS-SVM have the same optimization cost function; however, ELM has milder optimization constraints compared to LS-SVM and PSVM. As analyzed in this paper and further verified by simulation results over 36 wide types of data sets, compared to ELM, LS-SVM achieves *suboptimal solutions* (when the same kernels are used) and has higher computational complexity. *As verified by simulations, the resultant ELM method can run much faster than LS-SVM. ELM with random hidden nodes can run even up to tens of thousands times faster than SVM and LS-SVM*. Different from earlier ELM works which do not perform well in sparse data sets, the ELM method proposed in this paper can handle sparse data sets well.
4) This paper also shows that the proposed ELM method not only has universal approximation capability (of approximating any target continuous function) but also has classification capability (of classifying any disjoint regions).

II. Brief of SVMs

This section briefs the conventional SVM [1] and its variants, namely, LS-SVM [2] and PSVM [3].

A. SVM

Cortes and Vapnik [1] study the relationship between SVM and multilayer feedforward neural networks and showed that SVM can be seen as a specific type of SLFNs, the so-called support vector networks. In 1962, Rosenblatt [31] suggested that multilayer feedforward neural networks (perceptrons) can be trained in a feature space Z of the last hidden layer. In this feature space, a linear decision function is constructed

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{L} \alpha_i z_i(\mathbf{x})\right) \tag{3}$$

where $z_i(\mathbf{x})$ is the output of the ith neuron in the last hidden layer of a perceptron. In order to find an alternative solution of $z_i(\mathbf{x})$, in 1995, Cortes and Vapnik [1] proposed the SVM which maps the data from the input space to some feature space Z through some nonlinear mapping $\boldsymbol{\phi}(\mathbf{x})$ chosen *a priori*. Constrained-optimization methods are then used to find the separating hyperplane which maximizes the separating margins of two different classes in the feature space.

Given a set of training data $(\mathbf{x}_i, t_i)$, $i = 1, \ldots, N$, where $\mathbf{x}_i \in \mathbf{R}^d$ and $t_i \in \{-1, 1\}$, due to the nonlinear separability of these training data in the input space, in most cases, one can map the training data $\mathbf{x}_i$ from the input space to a feature space Z through a nonlinear mapping $\boldsymbol{\phi} : \mathbf{x}_i \rightarrow \boldsymbol{\phi}(\mathbf{x}_i)$. The distance between two different classes in the feature space Z is $2/\|\mathbf{w}\|$. To maximize the separating margin and to minimize the training errors, ξ_i, is equivalent to

$$\begin{aligned} \text{Minimize : } & L_{P_{\text{SVM}}} = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{Subject to : } & t_i\left(\mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x}_i) + b\right) \geq 1 - \xi_i, \qquad i = 1, \ldots, N \\ & \xi_i \geq 0, \qquad i = 1, \ldots, N \end{aligned} \tag{4}$$

where C is a user-specified parameter and provides a tradeoff between the distance of the separating margin and the training error.

Based on the Karush–Kuhn–Tucker (KKT) theorem [32], to train such an SVM is equivalent to solving the following dual optimization problem:

$$\begin{aligned} \text{minimize : } & L_{D_{\text{SVM}}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} t_i t_j \alpha_i \alpha_j \boldsymbol{\phi}(\mathbf{x}_i)\cdot\boldsymbol{\phi}(\mathbf{x}_j) \\ & \qquad - \sum_{i=1}^{N}\alpha_i \\ \text{subject to : } & \sum_{i=1}^{N} t_i\alpha_i = 0 \\ & 0 \leq \alpha_i \leq C, \qquad i = 1, \ldots, N \end{aligned} \tag{5}$$

where each Lagrange multiplier α_i corresponds to a training sample $(\mathbf{x}_i, t_i)$. Vectors $\mathbf{x}_i$'s for which $t_i(\mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x}_i) + b) = 1$ are termed support vectors [1].

Kernel functions $K(\mathbf{u}, \mathbf{v}) = \boldsymbol{\phi}(\mathbf{u})\cdot\boldsymbol{\phi}(\mathbf{v})$ are usually used in the implementation of SVM learning algorithm. In this case, we have

$$\begin{aligned} \text{minimize : } & L_{D_{\text{SVM}}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} t_i t_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i \\ \text{subject to : } & \sum_{i=1}^{N} t_i\alpha_i = 0 \\ & 0 \leq \alpha_i \leq C, \qquad i = 1, \ldots, N. \end{aligned} \tag{6}$$

The SVM kernel function $K(\mathbf{u}, \mathbf{v})$ needs to satisfy Mercer's condition [1]. The decision function of SVM is

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{s=1}^{N_s} \alpha_s t_s K(\mathbf{x}, \mathbf{x}_s) + b\right) \tag{7}$$

where N_s is the number of support vectors $\mathbf{x}_s$'s.

B. LS-SVM

Suykens and Vandewalle [2] propose a least square version to SVM classifier. Instead of the inequality constraint (4) adopted in SVM, equality constraints are used in the LS-SVM [2]. Hence, by solving a set of linear equations instead of quadratic programming, one can implement the least square approach easily. LS-SVM is proven to have excellent generalization performance and low computational cost in many applications.

In LS-SVM, the classification problem is formulated as

$$\begin{aligned} \text{Minimize : } & L_{P_{\text{LS-SVM}}} = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\frac{1}{2}\sum_{i=1}^{N}\xi_i^2 \\ \text{Subject to : } & t_i\left(\mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x}_i) + b\right) = 1 - \xi_i, \qquad i = 1, \ldots, N. \end{aligned} \tag{8}$$

Based on the KKT theorem, to train such an LS-SVM is equivalent to solving the following dual optimization problem:

$$\begin{aligned} L_{D_{\text{LS-SVM}}} = & \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\frac{1}{2}\sum_{i=1}^{N}\xi_i^2 \\ & - \sum_{i=1}^{N}\alpha_i\left(t_i\left(\mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x}_i) + b\right) - 1 + \xi_i\right). \end{aligned} \tag{9}$$

Different from Lagrange multipliers (5) in SVM, in LS-SVM, Lagrange multipliers α_i's can be either positive or negative due to the equality constraints used. Based on the KKT theorem, we can have the optimality conditions of (9) as follows:

$$\frac{\partial L_{D_{\text{LS-SVM}}}}{\partial \mathbf{w}} = 0 \rightarrow \mathbf{w} = \sum_{i=1}^{N}\alpha_i t_i \boldsymbol{\phi}(\mathbf{x}_i) \tag{10a}$$

$$\frac{\partial L_{D_{\mathrm{LS-SVM}}}}{\partial b} = 0 \rightarrow \sum_{i=1}^{N} \alpha_i t_i = 0 \tag{10b}$$

$$\frac{\partial L_{D_{\mathrm{LS-SVM}}}}{\partial \xi_i} = 0 \rightarrow \alpha_i = C\xi_i, \qquad i = 1, \ldots, N \tag{10c}$$

$$\frac{\partial L_{D_{\mathrm{LS-SVM}}}}{\partial \alpha_i} = 0 \rightarrow t_i \left(\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i) + b\right) - 1 + \xi_i = 0, \quad i = 1, \ldots, N. \tag{10d}$$

By substituting (10a)–(10c) into (10d), the aforementioned equations can be equivalently written as

$$\begin{bmatrix} \mathbf{0} & \mathbf{T}^{\mathrm{T}} \\ \mathbf{T} & \frac{\mathbf{I}}{\mathbf{C}} + \boldsymbol{\Omega}_{\mathrm{LS-SVM}} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{T}^{\mathrm{T}} \\ \mathbf{T} & \frac{\mathbf{I}}{C} + \mathbf{Z}\mathbf{Z}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \vec{\mathbf{1}} \end{bmatrix} \tag{11}$$

where

$$\mathbf{Z} = \begin{bmatrix} t_1 \boldsymbol{\phi}(\mathbf{x}_1) \\ \vdots \\ t_N \boldsymbol{\phi}(\mathbf{x}_N) \end{bmatrix}$$

$$\boldsymbol{\Omega}_{\mathrm{LS-SVM}} = \mathbf{Z}\mathbf{Z}^{\mathrm{T}}. \tag{12}$$

The feature mapping $\boldsymbol{\phi}(\mathbf{x})$ is a row vector,[1] $\mathbf{T} = [t_1, t_2, \ldots, t_N]^{\mathrm{T}}$, $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_N]^{\mathrm{T}}$, and $\vec{\mathbf{1}} = [1, 1, \ldots, 1]^{\mathrm{T}}$. In LS-SVM, as $\boldsymbol{\phi}(\mathbf{x})$ is usually unknown, Mercer's condition [33] can be applied to matrix $\boldsymbol{\Omega}_{\mathrm{LS-SVM}}$

$$\Omega_{\mathrm{LS-SVM_{i,j}}} = t_i t_j \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j) = t_i t_j K(\mathbf{x}_i, \mathbf{x}_j). \tag{13}$$

The decision function of LS-SVM classifier is $f(\mathbf{x}) = \mathrm{sign}(\sum_{i=1}^{N} \alpha_i t_i K(\mathbf{x}, \mathbf{x}_i) + b)$.

The Lagrange multipliers α_i's are proportional to the training errors ξ_i's in LS-SVM, while in the conventional SVM, many Lagrange multipliers α_i's are typically equal to zero. Compared to the conventional SVM, sparsity is lost in LS-SVM [9]; this is true to PSVM [3].

C. PSVM

Fung and Mangasarian [3] propose the PSVM classifier, which classifies data points depending on proximity to either one of the two separation planes that are aimed to be pushed away as far apart as possible. Similar to LS-SVM, the key idea of PSVM is that the separation hyperplanes are not bounded planes anymore but "proximal" planes, and such effect is reflected in mathematical expressions that the inequality constraints are changed to equality constraints. Different from LS-SVM, in the objective formula of linear PSVM, $(\mathbf{w} \cdot \mathbf{w} + b^2)$ is used instead of $\mathbf{w} \cdot \mathbf{w}$, making the optimization problem strongly convex, and has little or no effect on the original optimization problem.

[1]In order to keep the consistent notation and formula formats, similar to LS-SVM [2], PSVM [3], ELM [29], and TER-ELM [22], feature mappings $\boldsymbol{\phi}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$ are defined as a row vector while the rest of the vectors are defined as column vectors in this paper unless explicitly specified.

The mathematical model built for linear PSVM is

$$\begin{aligned} &\text{Minimize}: L_{P_{\mathrm{PSVM}}} = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w} + b^2) + C\frac{1}{2}\sum_{i=1}^{N} \xi_i^2 \\ &\text{Subject to}: t_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1 - \xi_i, \qquad i = 1, \ldots, N. \end{aligned} \tag{14}$$

The corresponding dual optimization problem is

$$L_{D_{\mathrm{PSVM}}} = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w} + b^2) + C\frac{1}{2}\sum_{i=1}^{N} \xi_i^2 - \sum_{i=1}^{N} \alpha_i \left(t_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i\right). \tag{15}$$

By applying KKT optimality conditions [similar to (10a)–(10d)], we have

$$\left(\frac{\mathbf{I}}{C} + \boldsymbol{\Omega}_{\mathrm{PSVM}} + \mathbf{T}\mathbf{T}^{\mathrm{T}}\right)\boldsymbol{\alpha} = \left(\frac{\mathbf{I}}{C} + \mathbf{Z}\mathbf{Z}^{\mathrm{T}} + \mathbf{T}\mathbf{T}^{\mathrm{T}}\right)\boldsymbol{\alpha} = \vec{\mathbf{1}} \tag{16}$$

where $\mathbf{Z} = [t_1\mathbf{x}_1, \ldots, t_N\mathbf{x}_N]^{\mathrm{T}}$ and $\boldsymbol{\Omega}_{\mathrm{PSVM}} = \mathbf{Z}\mathbf{Z}^{\mathrm{T}}$.

Similar to LS-SVM, the training data $\mathbf{x}$ can be mapped from the input space into a feature space $\boldsymbol{\phi} : \mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x})$, and one can obtain the nonlinear version of PSVM: $\mathbf{Z} = [t_1\boldsymbol{\phi}(\mathbf{x}_1)^{\mathrm{T}}, \ldots, t_N\boldsymbol{\phi}(\mathbf{x}_N)^{\mathrm{T}}]^{\mathrm{T}}$. As feature mapping $\boldsymbol{\phi}(\mathbf{x})$ is usually unknown, Mercer's conditions can be applied to matrix $\boldsymbol{\Omega}_{\mathrm{PSVM}} : \Omega_{\mathrm{PSVM}_{i,j}} = t_i t_j \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j) = t_i t_j K(\mathbf{x}_i, \mathbf{x}_j)$, which is the same as LS-SVM's kernel matrix $\Omega_{\mathrm{LS-SVM}}$ (13). The decision function of PSVM classifier is $f(\mathbf{x}) = \mathrm{sign}(\sum_{i=1}^{N} \alpha_i t_i K(\mathbf{x}, \mathbf{x}_i) + b)$.

III. Proposed Constrained-Optimization-Based ELM

ELM [12]–[14] was originally proposed for the single-hidden-layer feedforward *neural* networks and was then extended to the generalized SLFNs where the hidden layer need not be neuron alike [15], [16]. In ELM, the hidden layer need not be tuned. The output function of ELM for generalized SLFNs (take one output node case as an example) is

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i h_i(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} \tag{17}$$

where $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_L]^{\mathrm{T}}$ is the vector of the output weights between the hidden layer of L nodes and the output node and $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \ldots, h_L(\mathbf{x})]$ is the output (row) vector of the hidden layer with respect to the input $\mathbf{x}$. $\mathbf{h}(\mathbf{x})$ actually maps the data from the d-dimensional input space to the L-dimensional hidden-layer feature space (*ELM feature space*) H, and thus, $\mathbf{h}(\mathbf{x})$ is indeed a feature mapping. For the binary classification applications, the decision function of ELM is

$$f_L(\mathbf{x}) = \mathrm{sign}\left(\mathbf{h}(\mathbf{x})\boldsymbol{\beta}\right). \tag{18}$$

Different from traditional learning algorithms [23], ELM tends to reach not only the smallest training error but also the smallest norm of output weights. According to Bartlett's theory [34], for feedforward neural networks reaching smaller training error, the smaller the norms of weights are, the better

generalization performance the networks tend to have. We conjecture that this may be true to the generalized SLFNs where the hidden layer may not be neuron alike [15], [16]. ELM is to minimize the training error as well as the norm of the output weights [12], [13]

$$\text{Minimize}: \|\mathbf{H}\boldsymbol{\beta}-\mathbf{T}\|^2 \quad \text{and} \quad \|\boldsymbol{\beta}\| \tag{19}$$

where $\mathbf{H}$ is the hidden-layer output matrix

$$\mathbf{H}=\begin{bmatrix}\mathbf{h}(\mathbf{x}_1)\\ \vdots \\ \mathbf{h}(\mathbf{x}_N)\end{bmatrix}=\begin{bmatrix}h_1(\mathbf{x}_1) & \cdots & h_L(\mathbf{x}_1)\\ \vdots & \vdots & \vdots \\ h_1(\mathbf{x}_N) & \vdots & h_L(\mathbf{x}_N)\end{bmatrix}. \tag{20}$$

Seen from (18), to minimize the norm of the output weights $\|\boldsymbol{\beta}\|$ is actually to maximize the distance of the separating margins of the two different classes in the ELM feature space: $2/\|\boldsymbol{\beta}\|$.

The minimal norm least square method instead of the standard optimization method was used in the original implementation of ELM [12], [13]

$$\boldsymbol{\beta}=\mathbf{H}^{\dagger}\mathbf{T} \tag{21}$$

where $\mathbf{H}^{\dagger}$ is the *Moore–Penrose generalized inverse* of matrix $\mathbf{H}$ [35], [36]. Different methods can be used to calculate the Moore–Penrose generalized inverse of a matrix: orthogonal projection method, orthogonalization method, iterative method, and singular value decomposition (SVD) [36]. The orthogonal projection method [36] can be used in two cases: when $\mathbf{H}^{\mathrm{T}}\mathbf{H}$ is nonsingular and $\mathbf{H}^{\dagger}=(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}$, or when $\mathbf{H}\mathbf{H}^{\mathrm{T}}$ is nonsingular and $\mathbf{H}^{\dagger}=\mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{H}^{\mathrm{T}})^{-1}$.

According to the ridge regression theory [37], one can add a positive value to the diagonal of $\mathbf{H}^{\mathrm{T}}\mathbf{H}$ or $\mathbf{H}\mathbf{H}^{\mathrm{T}}$; the resultant solution is stabler and tends to have better generalization performance. Toh [22] and Deng *et al.* [21] have studied the performance of ELM with this enhancement under the *Sigmoid* additive type of SLFNs. This section extends such study to *generalized SLFNs* with a *different type of hidden nodes (feature mappings)* as well as *kernels*.

There is a gap between ELM and LS-SVM/PSVM, and it is not clear whether there is some relationship between ELM and LS-SVM/PSVM. This section aims to fill the gap and build the relationship between ELM and LS-SVM/PSVM.

A. Sufficient and Necessary Conditions for Universal Classifiers

1) Universal Approximation Capability: According to ELM learning theory, a widespread type of feature mappings $\mathbf{h}(\mathbf{x})$ can be used in ELM so that ELM can approximate any continuous target functions (refer to [14]–[16] for details). That is, given any target continuous function $f(\mathbf{x})$, there exists a series of β_i's such that

$$\lim_{L\to+\infty}\|f_L(\mathbf{x})-f(\mathbf{x})\|=\lim_{L\to+\infty}\left\|\sum_{i=1}^{L}\beta_i h_i(\mathbf{x})-f(\mathbf{x})\right\|=0. \tag{22}$$

With this universal approximation capability, the bias b in the optimization constraints of SVM, LS-SVM, and PSVM can be removed, and the resultant learning algorithm has milder optimization constraints. Thus, better generalization performance and lower computational complexity can be obtained. In SVM, LS-SVM, and PSVM, as the feature mapping $\boldsymbol{\phi}(\mathbf{x}_i)$ may be unknown, usually not every feature mapping to be used in SVM, LS-SVM, and PSVM satisfies the universal approximation condition. Obviously, a learning machine with a feature mapping which does not satisfy the universal approximation condition cannot approximate all target continuous functions. Thus, the universal approximation condition is not only a sufficient condition but also a necessary condition for a feature mapping to be widely used. This is also true to classification applications.

2) Classification Capability: Similar to the classification capability theorem of single-hidden-layer feedforward neural networks [38], we can prove the classification capability of the generalized SLFNs with the hidden-layer mapping $\mathbf{h}(\mathbf{x})$ satisfying the universal approximation condition (22).

Definition 3.1: A closed set is called a region regardless whether it is bounded or not.

Lemma 3.1 [38]: Given disjoint regions $K_1, K_2, \ldots, K_m$ in $\mathbf{R}^d$ and the corresponding m arbitrary real values $c_1, c_2, \ldots, c_m$, and an arbitrary region X disjointed from any K_i, there exists a continuous function $f(\mathbf{x})$ such that $f(\mathbf{x})=c_i$ if $\mathbf{x}\in K_i$ and $f(\mathbf{x})=c_0$ if $x\in X$, where c_0 is an arbitrary real value different from $c_1, c_2, \ldots, c_p$.

The classification capability theorem of Huang *et al.* [38] can be extended to generalized SLFNs which need not be neuron alike.

Theorem 3.1: Given a feature mapping $\mathbf{h}(\mathbf{x})$, if $\mathbf{h}(\mathbf{x})\boldsymbol{\beta}$ is dense in $C(\mathbf{R}^d)$ or in $C(M)$, where M is a compact set of $\mathbf{R}^d$, then a generalized SLFN with such a hidden-layer mapping $\mathbf{h}(\mathbf{x})$ can separate arbitrary disjoint regions of any shapes in $\mathbf{R}^d$ or M.

Proof: Given m disjoint regions $K_1, K_2, \ldots, K_m$ in $\mathbf{R}^d$ and their corresponding m labels $c_1, c_2, \ldots, c_m$, according to Lemma 3.1, there exists a continuous function $f(\mathbf{x})$ in $C(\mathbf{R}^d)$ or on one compact set of $\mathbf{R}^d$ such that $f(\mathbf{x})=c_i$ if $\mathbf{x}\in K_i$. Hence, if $\mathbf{h}(\mathbf{x})\boldsymbol{\beta}$ is dense in $C(\mathbf{R}^d)$ or on one compact set of $\mathbf{R}^d$, then it can approximate the function $f(\mathbf{x})$, and there exists a corresponding generalized SLFN to implement such a function $f(\mathbf{x})$. Thus, such a generalized SLFN can separate these decision regions regardless of shapes of these regions. □

Seen from Theorem 3.1, it is a necessary and sufficient condition that the feature mapping $\mathbf{h}(\mathbf{x})$ is chosen to make $\mathbf{h}(\mathbf{x})\boldsymbol{\beta}$ have the capability of approximating any target continuous function. If $\mathbf{h}(\mathbf{x})\boldsymbol{\beta}$ cannot approximate any target continuous functions, there may exist some shapes of regions which cannot be separated by a classifier with such feature mapping $\mathbf{h}(\mathbf{x})$. In other words, as long as the dimensionality of the feature mapping (number of hidden nodes L in a classifier) is large enough, the output of the classifier $\mathbf{h}(\mathbf{x})\boldsymbol{\beta}$ can be as close to the class labels in the corresponding regions as possible.

In the binary classification case, ELM only uses a single-output node, and the class label closer to the output value of ELM is chosen as the predicted class label of the input data. There are two solutions for the multiclass classification case.

1) ELM only uses a single-output node, and among the multiclass labels, the class label closer to the output value of ELM is chosen as the predicted class label of the

input data. In this case, the ELM solution to the binary classification case becomes a specific case of multiclass solution.

2) ELM uses multioutput nodes, and the index of the output node with the highest output value is considered as the label of the input data.

For the sake of readability, these two solutions are analyzed separately. It can be found that, eventually, the same solution formula is obtained for both cases.

B. Simplified Constrained-Optimization Problems

1) Multiclass Classifier With Single Output: Since ELM can approximate any target continuous functions and the output of the ELM classifier $\mathbf{h}(\mathbf{x})\boldsymbol{\beta}$ can be as close to the class labels in the corresponding regions as possible, the classification problem for the proposed constrained-optimization-based ELM with a single-output node can be formulated as

$$\begin{aligned}&\text{Minimize}: \ L_{P_{\text{ELM}}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\xi_i^2\\ &\text{Subject to}: \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} = t_i - \xi_i, \qquad i = 1, \ldots, N. \end{aligned} \tag{23}$$

Based on the KKT theorem, to train ELM is equivalent to solving the following dual optimization problem:

$$L_{D_{\text{ELM}}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\xi_i^2 - \sum_{i=1}^{N}\alpha_i\left(\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - t_i + \xi_i\right) \tag{24}$$

where each Lagrange multiplier α_i corresponds to the ith training sample. We can have the KKT optimality conditions of (24) as follows:

$$\frac{\partial L_{D_{\text{ELM}}}}{\partial \boldsymbol{\beta}} = 0 \rightarrow \boldsymbol{\beta} = \sum_{i=1}^{N}\alpha_i\mathbf{h}(\mathbf{x}_i)^{\text{T}} = \mathbf{H}^{\text{T}}\boldsymbol{\alpha} \tag{25a}$$

$$\frac{\partial L_{D_{\text{ELM}}}}{\partial \xi_i} = 0 \rightarrow \alpha_i = C\xi_i, \qquad i = 1, \ldots, N \tag{25b}$$

$$\frac{\partial L_{D_{\text{ELM}}}}{\partial \alpha_i} = 0 \rightarrow \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - t_i + \xi_i = 0, \qquad i = 1, \ldots, N \tag{25c}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]^{\text{T}}$.

2) Multiclass Classifier With Multioutputs: An alternative approach for multiclass applications is to let ELM have multioutput nodes instead of a single-output node. m-class of classifiers have m output nodes. If the original class label is p, the expected output vector of the m output nodes is $\mathbf{t}_i = [0, \ldots, 0, \overset{p}{1}, 0, \ldots, 0]^{\text{T}}$. In this case, only the pth element of $\mathbf{t}_i = [t_{i,1}, \ldots, t_{i,m}]^{\text{T}}$ is one, while the rest of the elements are set to zero. The classification problem for ELM with multioutput nodes can be formulated as

$$\begin{aligned}&\text{Minimize}: L_{P_{\text{ELM}}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi_i}\|^2\\ &\text{Subject to}: \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} = \mathbf{t}_i^{\text{T}} - \boldsymbol{\xi}_i^{\text{T}}, \qquad i = 1, \ldots, N \end{aligned} \tag{26}$$

where $\boldsymbol{\xi}_i = [\xi_{i,1}, \ldots, \xi_{i,m}]^{\text{T}}$ is the training error vector of the m output nodes with respect to the training sample $\mathbf{x}_i$. Based on the KKT theorem, to train ELM is equivalent to solving the following dual optimization problem:

$$L_{D_{\text{ELM}}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|^2 - \sum_{i=1}^{N}\sum_{j=1}^{m}\alpha_{i,j}\left(\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta}_j - t_{i,j} + \xi_{i,j}\right) \tag{27}$$

where $\boldsymbol{\beta}_j$ is the vector of the weights linking hidden layer to the jth output node and $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_m]$. We can have the KKT corresponding optimality conditions as follows:

$$\frac{\partial L_{D_{\text{ELM}}}}{\partial \boldsymbol{\beta}_j} = 0 \rightarrow \boldsymbol{\beta}_j = \sum_{i=1}^{N}\alpha_{i,j}\mathbf{h}(\mathbf{x}_i)^{\text{T}} \rightarrow \boldsymbol{\beta} = \mathbf{H}^{\text{T}}\boldsymbol{\alpha} \tag{28a}$$

$$\frac{\partial L_{D_{\text{ELM}}}}{\partial \boldsymbol{\xi}_i} = 0 \rightarrow \boldsymbol{\alpha}_i = C\boldsymbol{\xi}_i, \qquad i = 1, \ldots, N \tag{28b}$$

$$\frac{\partial L_{D_{\text{ELM}}}}{\partial \boldsymbol{\alpha}_i} = 0 \rightarrow \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - \mathbf{t}_i^{\text{T}} + \boldsymbol{\xi}_i^{\text{T}} = 0, \qquad i = 1, \ldots, N \tag{28c}$$

where $\boldsymbol{\alpha}_i = [\alpha_{i,1}, \ldots, \alpha_{i,m}]^{\text{T}}$ and $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_N]^{\text{T}}$.

It can be seen from (24), (25a)–(25c), (27), and (28a)–(28c) that the single-output node case considered a specific case of multioutput nodes when the number of output nodes is set to one: $m = 1$. Thus, we only need to consider the multiclass classifier with multioutput nodes. For both cases, the hidden-layer matrix $\mathbf{H}$ (20) remains the same, and the size of $\mathbf{H}$ is only decided by the number of training samples N and the number of hidden nodes L, which is irrelevant to the number of output nodes (number of classes).

C. Equality Constrained-Optimization-Based ELM

Different solutions to the aforementioned KKT conditions can be obtained based on the concerns on the efficiency in different size of training data sets.

1) For the Case Where the Number of Training Samples is Not Huge: In this case, by substituting (28a) and (28b) into (28c), the aforementioned equations can be equivalently written as

$$\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^{\text{T}}\right)\boldsymbol{\alpha} = \mathbf{T} \tag{29}$$

where

$$\mathbf{T} = \begin{bmatrix}\mathbf{t}_1^{\text{T}}\\ \vdots \\ \mathbf{t}_N^{\text{T}}\end{bmatrix} = \begin{bmatrix}t_{11} & \cdots & t_{1m}\\ \vdots & \vdots & \vdots \\ t_{N1} & \cdots & t_{Nm}\end{bmatrix}. \tag{30}$$

From (28a) and (29), we have

$$\boldsymbol{\beta} = \mathbf{H}^{\text{T}}\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^{\text{T}}\right)^{-1}\mathbf{T}. \tag{31}$$

The output function of ELM classifier is

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = \mathbf{h}(\mathbf{x})\mathbf{H}^{\mathrm{T}}\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^{\mathrm{T}}\right)^{-1}\mathbf{T}. \quad (32)$$

1) Single-output node $(m = 1)$: For multiclass classifications, among all the multiclass labels, the predicted class label of a given testing sample is closest to the output of ELM classifier. For binary classification case, ELM needs only one output node $(m = 1)$, and the decision function of ELM classifier is

$$f(\mathbf{x}) = \operatorname{sign}\left(\mathbf{h}(\mathbf{x})\mathbf{H}^{\mathrm{T}}\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^{\mathrm{T}}\right)^{-1}\mathbf{T}\right). \quad (33)$$

2) Multioutput nodes $(m > 1)$: For multiclass cases, the predicted class label of a given testing sample is the index number of the output node which has the highest output value for the given testing sample. Let $f_j(\mathbf{x})$ denote the output function of the jth output node, i.e., $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})]^{\mathrm{T}}$; then, the predicted class label of sample $\mathbf{x}$ is

$$label(\mathbf{x}) = \arg\max_{i\in\{1,\ldots,m\}} f_i(\mathbf{x}). \quad (34)$$

2) For the Case Where the Number of Training Samples is Huge: If the number of training data is very large, for example, it is much larger than the dimensionality of the feature space, $N \gg L$, we have an alternative solution. From (28a) and (28b), we have

$$\boldsymbol{\beta} = C\mathbf{H}^{\mathrm{T}}\boldsymbol{\xi} \quad (35)$$

$$\boldsymbol{\xi} = \frac{1}{C}\left(\mathbf{H}^{\mathrm{T}}\right)^{\dagger}\boldsymbol{\beta}. \quad (36)$$

From (28c), we have

$$\mathbf{H}\boldsymbol{\beta} - \mathbf{T} + \frac{1}{C}(\mathbf{H}^{\mathrm{T}})^{\dagger}\boldsymbol{\beta} = 0$$

$$\mathbf{H}^{\mathrm{T}}\left(\mathbf{H} + \frac{1}{C}(\mathbf{H}^{\mathrm{T}})^{\dagger}\right)\boldsymbol{\beta} = \mathbf{H}^{\mathrm{T}}\mathbf{T}$$

$$\boldsymbol{\beta} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{T}. \quad (37)$$

In this case, the output function of ELM classifier is

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = \mathbf{h}(\mathbf{x})\left(\frac{\mathbf{I}}{C} + \mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{T}. \quad (38)$$

1) Single-output node $(m = 1)$: For multiclass classifications, the predicted class label of a given testing sample is the class label closest to the output value of ELM classifier. For binary classification case, the decision function of ELM classifier is

$$f(\mathbf{x}) = \operatorname{sign}\left(\mathbf{h}(\mathbf{x})\left(\frac{\mathbf{I}}{C} + \mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{T}\right). \quad (39)$$

2) Multioutput nodes $(m > 1)$: The predicted class label of a given testing sample is the index of the output node which has the highest output.

Remark: Although the alternative approaches for the different size of data sets are discussed and provided separately, in theory, there is no specific requirement on the size of the training data sets in all the approaches [see (32) and (38)], and all the approaches can be used in any size of applications. However, different approaches have different computational costs, and their efficiency may be different in different applications. In the implementation of ELM, it is found that the generalization performance of ELM is not sensitive to the dimensionality of the feature space (L) and good performance can be reached as long as L is large enough. In our simulations, $L = 1000$ is set for all tested cases no matter whatever size of the training data sets. Thus, if the training data sets are very large $N \gg L$, one may prefer to apply solutions (38) in order to reduce computational costs. However, if a feature mapping $\mathbf{h}(\mathbf{x})$ is *unknown*, one may prefer to use solutions (32) instead (which will be discussed later in Section IV).

IV. Discussions

A. Random Feature Mappings and Kernels

1) Random Feature Mappings: Different from SVM, LS-SVM, and PSVM, in ELM, a feature mapping (hidden-layer output vector) $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \ldots, h_L(\mathbf{x})]$ is usually known to users. According to [15] and [16], almost all nonlinear piecewise continuous functions can be used as the hidden-node output functions, and thus, the feature mappings used in ELM can be very diversified.

For example, as mentioned in [29], we can have

$$\mathbf{h}(\mathbf{x}) = [G(\mathbf{a}_1, b_1, \mathbf{x}), \ldots, G(\mathbf{a}_L, b_L, \mathbf{x})] \quad (40)$$

where $G(\mathbf{a}, b, \mathbf{x})$ is a nonlinear piecewise continuous function satisfying ELM universal approximation capability theorems [14]–[16] and $\{(\mathbf{a}_i, b_i)\}_{i=1}^{L}$ are *randomly* generated according to any continuous probability distribution. For example, such nonlinear piecewise continuous functions can be as follows.

1) Sigmoid function

$$G(\mathbf{a}, b, \mathbf{x}) = \frac{1}{1 + \exp\left(-(\mathbf{a}\cdot\mathbf{x} + b)\right)}. \quad (41)$$

2) Hard-limit function

$$G(\mathbf{a}, b, \mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{a}\cdot\mathbf{x} - b \geq 0 \\ 0, & \text{otherwise.} \end{cases} \quad (42)$$

3) Gaussian function

$$G(\mathbf{a}, b, \mathbf{x}) = \exp\left(-b\|\mathbf{x} - \mathbf{a}\|^2\right). \quad (43)$$

4) Multiquadric function

$$G(\mathbf{a}, b, \mathbf{x}) = \left(\|\mathbf{x} - \mathbf{a}\|^2 + b^2\right)^{1/2}. \quad (44)$$

Sigmoid and Gaussian functions are two of the major hidden-layer output functions used in the feedforward neural networks and RBF networks[2], respectively. Interestingly, ELM with

[2]Readers can refer to [39] for the difference between ELM and RBF networks.

hard-limit [24] and multiquadric functions can have good generalization performance as well.

Suykens and Vandewalle [30] described a training method for SLFNs which applies the hidden-layer output mapping as the feature mapping of SVM. However, different from ELM where the hidden layer is *not parametric* and need not be tuned, the feature mapping of their SVM implementation is *parametric*, and the hidden-layer parameters need to be iteratively computed by solving an optimization problem. Their learning algorithm was briefed as follows:

$$\begin{aligned}&\text{minimize} : \|r\mathbf{w}\|^2 \\ &\text{subject to} : \\ &\text{C1} : \text{QP subproblem} : \\ &\quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i^* t_i \tanh(\mathbf{V}\mathbf{x}_i + \mathbf{B}) \\ &\quad \alpha_i^* = \arg\max_{\alpha_i} \mathcal{Q}\left(\alpha_i; \tanh(\mathbf{V}\mathbf{x}_i + \mathbf{B})\right) \\ &\quad 0 \le \alpha_i^* \le c \\ &\text{C2} : \ \|\mathbf{V}(:); \mathbf{B}\|_2 \le \gamma \\ &\text{C3} : \ r \text{ is radius of smallest ball containing} \\ &\qquad \{\tanh(\mathbf{V}\mathbf{x}_i) + \mathbf{B}\}_{i=1}^{N}\end{aligned} \tag{45, 46}$$

where $\mathbf{V}$ denotes the interconnection matrix for the hidden layer, $\mathbf{B}$ is the bias vector, (:) is a columnwise scan of the interconnection matrix for the hidden layer, and γ is a positive constant. In addition, $\mathcal{Q}$ is the cost function of the corresponding SVM dual problem

$$\max_{\alpha_i} \mathcal{Q}\left(\alpha_i; K(\mathbf{x}_i, \mathbf{x}_j)\right) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} t_i t_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{N} \alpha_i. \tag{47}$$

QP subproblems need to be solved for hidden-node parameters $\mathbf{V}$ *and* $\mathbf{B}$, *while the hidden-node parameters of ELM are randomly generated and known to users*.

2) Kernels: If a feature mapping $\mathbf{h}(\mathbf{x})$ is unknown to users, one can apply Mercer's conditions on ELM. We can define a kernel matrix for ELM as follows:

$$\mathbf{\Omega}_{\text{ELM}} = \mathbf{H}\mathbf{H}^{\text{T}} : \Omega_{\text{ELM}i,j} = h(\mathbf{x}_i) \cdot h(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j). \tag{48}$$

Then, the output function of ELM classifier (32) can be written compactly as

$$\begin{aligned}\mathbf{f}(\mathbf{x}) &= \mathbf{h}(\mathbf{x})\mathbf{H}^{\text{T}}\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^{\text{T}}\right)^{-1}\mathbf{T} \\ &= \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N)\end{bmatrix}^{\text{T}}\left(\frac{\mathbf{I}}{C} + \mathbf{\Omega}_{\text{ELM}}\right)^{-1}\mathbf{T}.\end{aligned} \tag{49}$$

In this specific case, similar to SVM, LS-SVM, and PSVM, the feature mapping $\mathbf{h}(\mathbf{x})$ need not be known to users; instead, its corresponding kernel $K(\mathbf{u}, \mathbf{v})$ (e.g., $K(\mathbf{u}, \mathbf{v}) = \exp(-\gamma\|\mathbf{u} - \mathbf{v}\|^2)$) is given to users. The dimensionality L of the feature space (number of hidden nodes) need not be given either.

3) Feature Mapping Matrix: In ELM, $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1)^{\text{T}}, \ldots, \mathbf{h}(\mathbf{x}_N)^{\text{T}}]^{\text{T}}$ is called the hidden-layer output matrix (or called feature mapping matrix) due to the fact that it represents the corresponding hidden-layer outputs of the given N training samples. $\mathbf{h}(\mathbf{x}_i)$ denotes the output of the hidden layer with regard to the input sample $\mathbf{x}_i$. Feature mapping $\mathbf{h}(\mathbf{x}_i)$ maps the data $\mathbf{x}_i$ from the input space to the hidden-layer feature space, and the feature mapping matrix $\mathbf{H}$ is irrelevant to target t_i. As observed from the essence of the feature mapping, it is reasonable to have the feature mapping matrix independent from the target values t_i's. However, in both LS-SVM and PSVM, the feature mapping matrix $\mathbf{Z} = [t_1\boldsymbol{\phi}(\mathbf{x}_1)^{\text{T}}, \ldots, t_N\boldsymbol{\phi}(\mathbf{x}_N)^{\text{T}}]^{\text{T}}$ (12) is designed to depend on the targets t_i's of the training samples $\mathbf{x}_i$'s.

B. ELM: Unified Learning Mode for Regression, Binary, and Multiclass Classification

As observed from (32) and (38), ELM has the unified solutions for regression, binary, and multiclass classification. The kernel matrix $\mathbf{\Omega}_{\text{ELM}} = \mathbf{H}\mathbf{H}^{\text{T}}$ is only related to the input data $\mathbf{x}_i$ and the number of training samples. The kernel matrix $\mathbf{\Omega}_{\text{ELM}}$ is neither relevant to the number of output nodes m nor to the training target values t_i's. However, in multiclass LS-SVM, aside from the input data $\mathbf{x}_i$, the kernel matrix $\mathbf{\Omega}_M$ (52) also depends on the number of output nodes m and the training target values t_i's.

For the multiclass case with m labels, LS-SVM uses m output nodes in order to encode multiclasses where $t_{i,j}$ denotes the output value of the jth output node for the training data $\mathbf{x}_i$ [10]. The m outputs can be used to encode up to 2^m different classes. For multiclass case, the primal optimization problem of LS-SVM can be given as [10]

$$\begin{aligned}&\text{Minimize} : L_{P_{\text{LS-SVM}}}^{(m)} = \frac{1}{2}\sum_{j=1}^{m}\mathbf{w}_j \cdot \mathbf{w}_j + C\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{m}\xi_{i,j}^2 \\ &\text{Subject to} : \begin{cases} t_{i,1}\left(\mathbf{w}_1 \cdot \boldsymbol{\phi}_1(\mathbf{x}_i) + b_1\right) = 1 - \xi_{i,1} \\ t_{i,2}\left(\mathbf{w}_2 \cdot \boldsymbol{\phi}_2(\mathbf{x}_i) + b_2\right) = 1 - \xi_{i,2} \\ \ldots \\ t_{i,m}\left(\mathbf{w}_m \cdot \boldsymbol{\phi}_m(\mathbf{x}_i) + b_m\right) = 1 - \xi_{i,m}\end{cases} \\ &\qquad\qquad i = 1, \ldots, N.\end{aligned} \tag{50}$$

Similar to the LS-SVM solution (11) to the binary classification, with KKT conditions, the corresponding LS-SVM solution for multiclass cases can be obtained as follows:

$$\begin{bmatrix}\mathbf{0} & \mathbf{T}^{\text{T}} \\ \mathbf{T} & \mathbf{\Omega}_M\end{bmatrix}\begin{bmatrix}\mathbf{b}_M \\ \boldsymbol{\alpha}_M\end{bmatrix} = \begin{bmatrix}\mathbf{0} \\ \vec{\mathbf{1}}\end{bmatrix} \tag{51}$$

$$\begin{aligned}&\mathbf{\Omega}_M = \text{blockdiag}\left[\mathbf{\Omega}^{(1)} + \frac{\mathbf{I}}{C}, \ldots, \mathbf{\Omega}^{(m)} + \frac{\mathbf{I}}{C}\right] \\ &\Omega_{kl}^{(j)} = t_{k,j}t_{l,j}K^{(j)}(\mathbf{x}_k, \mathbf{x}_l) \\ &\mathbf{b}_M = [b_1, \ldots, b_m] \\ &\boldsymbol{\alpha}_M = [\alpha_{1,1}, \ldots, \alpha_{N,1}, \ldots, \alpha_{1,m}, \ldots, \alpha_{N,m}] \\ &K^{(j)}(\mathbf{x}_k, \mathbf{x}_l) = \boldsymbol{\phi}_j(\mathbf{x}_k) \cdot \boldsymbol{\phi}_j(\mathbf{x}_l)\end{aligned} \tag{52}$$

$$= \exp\left(-\frac{\|\mathbf{x}_k - \mathbf{x}_l\|^2}{\sigma_j^2}\right), \qquad j = 1, \ldots, N. \tag{53}$$

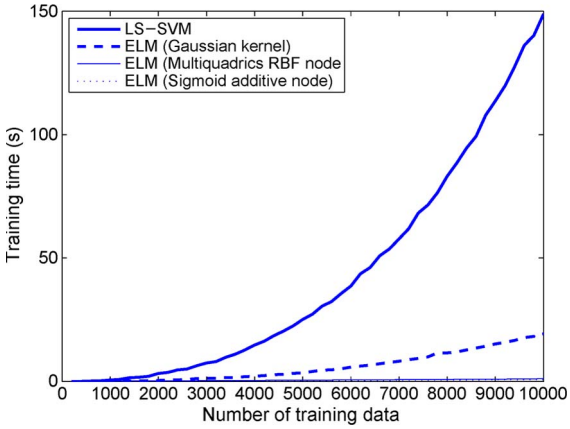


Fig. 1. Scalability of different classifiers: An example on Letter data set. The training time spent by LS-SVM and ELM (Gaussian kernel) increases sharply when the number of training data increases. However, the training time spent by ELM with Sigmoid additive node and multiquadric function node increases very slowly when the number of training data increases.

Seen from (52), the multiclass LS-SVM actually uses m binary-class LS-SVM concurrently for m class labels of classifications; each of the m binary-class LS-SVMs may have different kernel matrix $\mathbf{\Omega}^{(j)}$, $j = 1, \ldots, m$. However, in any cases, ELM has one hidden layer linking to all the m output nodes. In multiclass LS-SVM, different kernels may be used in each individual binary LS-SVM, and the jth LS-SVM uses kernel $K^{(j)}(\mathbf{u}, \mathbf{v})$. Take Gaussian kernel as an example, $K^{(j)}(\mathbf{u}, \mathbf{v}) = \exp(-(\|\mathbf{x}_k - \mathbf{x}_l\|^2/\sigma_j^2))$; from practical point of view, it may be time consuming and tedious for users to choose different kernel parameters σ_i, and thus, one may set a common value $\sigma_i = \sigma$ for all the kernels. In multiclass LS-SVM, the size of $\mathbf{\Omega}_M$ is $N \times Nm$, which is related to the number of output nodes m. However, in ELM, the size of kernel matrix $\mathbf{\Omega}_{\text{ELM}} = \mathbf{H}\mathbf{H}^{\text{T}}$ is $N \times N$, which is fixed for all the regression, binary, and multiclass classification cases.

C. Computational Complexity and Scalability

For LS-SVM and PSVM, the main computational cost comes from calculating the Lagrange multipliers $\boldsymbol{\alpha}$'s based on (11) and (16). Obviously, ELM computes $\boldsymbol{\alpha}$ based on a simpler method (29). More importantly, in large-scale applications, instead of $\mathbf{H}\mathbf{H}^{\text{T}}$ (size: $N \times N$), ELM can get a solution based on (37), where $\mathbf{H}^{\text{T}}\mathbf{H}$ (size: $L \times L$) is used. As in most applications, the number of hidden nodes L can be much smaller than the number of training samples: $L \ll N$, the computational cost reduces dramatically. For the case $L \ll N$, ELM can use $\mathbf{H}^{\text{T}}\mathbf{H}$ (size: $L \times L$). Compared with LS-SVM and PSVM which use $\mathbf{H}\mathbf{H}^{\text{T}}$ (size: $N \times N$), ELM has much better computational scalability with regard to the number of training samples N. (cf. Fig. 1 for example.)

In order to reduce the computational cost of LS-SVM in large-scale problems, fixed-size LS-SVM has been proposed by Suykens *et al.* [40]–[44]. Fixed-size LS-SVM uses an M-sample subset of the original training data set $(M \ll N)$ to compute a finite dimensional approximation $\hat{\boldsymbol{\phi}}(\mathbf{x})$ to the feature map $\boldsymbol{\phi}(\mathbf{x})$. However, different from the fixed-size LS-SVM, if $L \ll N$, $L \times L$ solution of ELM still uses the entire N training samples. In any case, the feature map $\mathbf{h}(\mathbf{x})$ of ELM is not approximated. In fact, the feature map $\mathbf{h}(\mathbf{x})$ of ELM is randomly generated and independent of the training samples (if random hidden nodes are used). The kernel matrix of the fixed-size LS-SVM is built with the subset of size $M \ll N$, while the kernel matrix of ELM is built with the entire data set of size N in all cases.

D. Difference From Other Regularized ELMs

Toh [22] and Deng *et al.* [21] proposed two different types of weighted regularized ELMs.

The total error rate (TER) ELM [22] uses m output nodes for m class label classification applications. In TER-ELM, the counting cost function adopts a quadratic approximation. The OAA method is used in the implementation of TER-ELM in multiclass classification applications. Essentially, TER-ELM consists of m binary TER-ELM, where jth TER-ELM is trained with all of the samples in the jth class with positive labels and all the other examples from the remaining $m-1$ classes with negative labels. Suppose that there are m_j^+ number of positive category patterns and m_j^- number of negative category patterns in the jth binary TER-ELM. We have a positive output $y_j^+ = (\tau + \eta)\mathbf{1}_j^+$ for the jth class of samples and a negative class output $y_j^- = (\tau - \eta)\mathbf{1}_j^-$ for all the non-jth class of samples, where $\mathbf{1}_j^+ = [1, \ldots, 1]^{\text{T}} \in \mathbf{R}^{m_j^+}$ and $\mathbf{1}_j^- = [1, \ldots, 1]^{\text{T}} \in \mathbf{R}^{m_j^-}$. A common setting for threshold (τ) and bias (η) will be set for all the m outputs. The output weight vector $\boldsymbol{\beta}_j$ in jth binary TER-ELM is calculated as

$$\boldsymbol{\beta}_j = \left(\frac{1}{m_j^-}\mathbf{H}_j^{-T}\mathbf{H}_j^- + \frac{1}{m_j^+}\mathbf{H}_j^{+T}\mathbf{H}_j + \right) \cdot \left(\frac{1}{m_j^-}\mathbf{H}_j^{-T}y_j^- + \frac{1}{m_j^+}\mathbf{H}_j^{+T}y_j^+\right) \quad (54)$$

where $\mathbf{H}_j^+$ and $\mathbf{H}_j^-$ denote the hidden-layer matrices of the jth binary TER-ELM corresponding to the positive and negative samples, respectively.

By defining two class-specific diagonal weighting matrices $\mathbf{W}_j^+ = \text{diag}(0, \ldots, 0, 1/m_j^+, \ldots, m_j^+)$ and $\mathbf{W}_j^- = \text{diag}(1/m_j^-, \ldots, m_j^-, 0, \ldots, 0)$, the solution formula (54) of TER-ELM can be written as

$$\boldsymbol{\beta}_j = \left(\frac{\mathbf{I}}{C} + \mathbf{H}_j^{\text{T}}\mathbf{W}_j\mathbf{H}_j\right)^{-1}\mathbf{H}_j^{\text{T}}\mathbf{W}_j\mathbf{y}_j \quad (55)$$

where $\mathbf{W}_j = \mathbf{W}_j^+ + \mathbf{W}_j^-$ and the elements of $\mathbf{H}_j$ and $\mathbf{y}_j$ are ordered according to the positive and negative samples of the two classes (jth class samples and all the non-jth class samples). In order to improve the stability of the learning, $\mathbf{I}/C$ is introduced in the aforementioned formula. If the dimensionality of the hidden layer is much larger than the number of the training data $(L \gg N)$, an alternative solution suggested in [22] is

$$\boldsymbol{\beta}_j = \mathbf{H}_j^{\text{T}}\left(\frac{\mathbf{I}}{C} + \mathbf{W}_j\mathbf{H}_j\mathbf{H}_j^{\text{T}}\right)^{-1}\mathbf{W}_j\mathbf{y}_j. \quad (56)$$

Kernels and generalized feature mappings are not considered in TER-ELM.

Deng *et al.* [21] mainly focus on the case where $L < N$, and (37) of ELM and the solution formula of Deng *et al.* [21] look similar to each other. However, different from the ELM solutions provided in this paper, Deng *et al.* [21] do not consider kernels and generalized feature mappings in their weighted regularized ELM. In the proposed solutions of ELM, L hidden nodes may have a different type of hidden-node output function $h_i(\mathbf{x}) : \mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \ldots, h_L(\mathbf{x})]$, while in [21], all the hidden nodes use the Sigmoid type of activation functions. Deng *et al.* [21] do not handle the alternative solution (31).

Seen from (37), multivariate polynomial model [45] can be considered as a specific case of ELM.

The original solutions (21) of ELM [12], [13], [26], TER-ELM [22], and the weighted regularized ELM [21] are not able to apply kernels in their implementations. With the new suggested approach, kernels can be used in ELM [cf. (49)].

E. Milder Optimization Constraints

In LS-SVM, as the feature mapping $\boldsymbol{\phi}(\mathbf{x})$ is usually unknown, it is reasonable to think that the separating hyperplane in LS-SVM may not necessarily pass through the origin in the LS-SVM feature space, and thus, a term bias b is required in their optimization constraints: $t_i(\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i) + b) = 1 - \xi_i$. The corresponding KKT condition (necessary condition) [cf. (10b)] for the conventional LS-SVM is $\sum_{i=1}^{N} \alpha_i t_i = 0$. Poggio *et al.* [46] prove in theory that the term bias b is not required in positive definite kernel and that it is *not incorrect* to have the term bias b in the SVM model. Different from the analysis of Poggio *et al.* [46], Huang *et al.* [29] show that, from the practical and universal approximation point of view, the term bias b should not be given in the ELM learning.

According to ELM theories [12]–[16], almost all nonlinear piecewise continuous functions as feature mappings can make ELM satisfy universal approximation capability, and the separating hyperplane of ELM basically tends to pass through the origin in the ELM feature space. There is no term bias b in the optimization constraint of ELM, $\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} = t_i - \xi_i$, and thus, different from LS-SVM, ELM does not need to satisfy the condition $\sum_{i=1}^{N} \alpha_i t_i = 0$. Although LS-SVM and ELM have the same primal optimization formula, ELM has milder optimization constraints than LS-SVM, and thus, compared to ELM, LS-SVM obtains a suboptimal optimization.

The differences and relationships among ELM, LS-SVM/PSVM, and SVM can be summarized in Table I.

V. Performance Verification

This section compares the performance of different algorithms (SVM, LS-SVM, and ELM) in real-world benchmark regression, binary, and multiclass classification data sets. In order to test the performance of the proposed ELM with various feature mappings in supersmall data sets, we have also tested ELM on the XOR problem.

A. Benchmark Data Sets

In order to extensively verify the performance of different algorithms, wide types of data sets have been tested in our simulations, which are of *small sizes*, *low dimensions*, *large sizes*, and/or *high dimensions*. These data sets include 12 binary classification cases, 12 multiclassification cases, and 12 regression cases. Most of the data sets are taken from UCI Machine Learning Repository [47] and Statlib [48].

TABLE II
Specification of Binary Classification Problems

Datasets	# train	# test	# features	Random Perm
Diabetes	512	256	8	Yes
Australian Credit	460	230	6	Yes
Liver	230	115	6	Yes
Banana	400	4900	2	No
Colon	30	32	2000	No
Colon (Gene Sel)	30	32	60	No
Leukemia	38	34	7129	No
Leukemia (Gene Sel)	38	34	60	No
Brightdata	1000	1462	14	Yes
Dimdata	1000	3192	14	Yes
Mushroom	1500	6624	22	Yes
Adult	4781	27780	123	No

1) Binary Class Data Sets: The 12 binary class data sets (cf. Table II) can be classified into four groups of data:

1) data sets with relatively small size and low dimensions, e.g., Pima Indians *diabetes*, Statlog *Australian credit*, Bupa *Liver* disorders [47], and *Banana* [49];
2) data sets with relatively small size and high dimensions, e.g., *leukemia* data set [50] and *colon* microarray data set [51];
3) data sets with relatively large size and low dimensions, e.g., Star/Galaxy-*Bright* data set [52], Galaxy *Dim* data set [52], and *mushroom* data set [47];
4) data sets with large size and high dimensions, e.g., *adult* data set [47].

The leukemia data set was originally taken from a collection of leukemia patient samples [53]. The data set consists of 72 samples: 25 samples of AML and 47 samples of ALL. Each sample of leukemia data set is measured over 7129 genes (cf. Leukemia in Table II). The colon microarray data set consists of 22 normal and 40 tumor tissue samples. In this data set, each sample of colon microarray data set contains 2000 genes (cf. Colon in Table II).

Performances of the different algorithms have also been tested on both leukemia data set and colon microarray data set after the minimum-redundancy–maximum-relevance feature selection method [54] being taken (cf. *Leukemia (Gene Sel)* and *Colon (Gene Sel)* in Table II).

2) Multiclass Data Sets: The 12 multiclass data sets (cf. Table III) can be classified into four groups of data as well:

1) data sets with relatively small size and low dimensions, e.g., *Iris*, *Glass* Identification, and *Wine* [47];
2) data sets with relatively medium size and medium dimensions, e.g., *Vowel* Recognition, Statlog *Vehicle* Silhouettes, and Statlog Image *Segmentation* [47];
3) data sets with relatively large size and medium dimensions, e.g., *letter* and *shuttle* [47];
4) data sets with large size and/or large dimensions, e.g., *DNA*, *Satimage* [47], and USPS [50].

3) Regression Data Sets: The 12 regression data sets (cf. Table IV) can be classified into three groups of data:

1) data sets with relatively small size and low dimensions, e.g., *Basketball*, *Strike* [48], *Cloud*, and *Autoprice* [47];

TABLE I
FEATURE COMPARISONS AMONG ELM, LS-SVM, AND SVM

	ELM	LS-SVM and PSVM	SVM
Feature mapping	i) $\mathbf{h}(\mathbf{x})$ is usually known to users (Wide type of nonlinear piecewise continuous functions (with *randomly* generated parameters) can be used [14]–[16]: e.g. additive, RBF, trigonometric, threshold, fully complex, high-order, ridge polynomial, etc.) ii) If $\mathbf{h}(\mathbf{x})$ is unknown, kernels can be used in ELM then.	$\boldsymbol{\phi}(\mathbf{x})$ is usually unknown to users and kernels are often used.	$\boldsymbol{\phi}(\mathbf{x})$ is usually unknown to users and kernels are often used.
Kernel matrix	$\boldsymbol{\Omega}_{\text{ELM}}=[\Omega_{ij}]$ where $\Omega_{ij}=K(\mathbf{x}_i,\mathbf{x}_j)$	$\boldsymbol{\Omega}_{\text{LS-SVM}}=\boldsymbol{\Omega}_{PSVM}=[\Omega_{ij}]$ where $\Omega_{ij}=t_it_jK(\mathbf{x}_i,\mathbf{x}_j)$	$\boldsymbol{\Omega}_{\text{SVM}}=[\Omega_{ij}]$ where $\Omega_{ij}=K(\mathbf{x}_i,\mathbf{x}_j)$
Universal approximation capability	Ensured for almost all type of nonlinear piecewise random hidden nodes and often used kernels	Not ensured. It depends on the kernel to be used	Not ensured. It depends on the kernel to be used
Support vectors based	No (α_i are proportional to the training errors)	No (α_i are proportional to the training errors)	Yes (Support vectors $\mathbf{x}_i$ corresponding to $\alpha_i=0$)
Conditions on α_i	No conditions on α_i	For LS-SVM: $\sum_{i=1}^{N}t_i\alpha_i=0$; For PSVM: $\sum_{i=1}^{N}t_i\alpha_i=b$	$\sum_{i=1}^{N}t_i\alpha_i=0$ and $0\le\alpha_i\le C$
Multi classes	Single ELM for m classes	m binary LS-SVM for m classes, depending on coding/decoding scheme	m or $m(m-1)/2$ binary SVM for m classes
Regression	ELM unified for both regression and binary / multi-class classification	Different variants required for regression and classification	Different variants required for regression and classification
Output function	Kernel based: $\mathbf{f}(\mathbf{x})=\begin{bmatrix}K(\mathbf{x},\mathbf{x}_1)\\ \vdots\\ K(\mathbf{x},\mathbf{x}_N)\end{bmatrix}^T\left(\frac{\mathbf{I}}{C}+\boldsymbol{\Omega}_{\text{ELM}}\right)^{-1}\mathbf{T}$ Non-kernel based: $\mathbf{f}(\mathbf{x})=\mathbf{h}(\mathbf{x})\mathbf{H}^T\left(\frac{\mathbf{I}}{C}+\mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T}$ $\mathbf{f}(\mathbf{x})=\mathbf{h}(\mathbf{x})\left(\frac{\mathbf{I}}{C}+\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}$	Kernel based LS-SVM and PSVM: $f(\mathbf{x})=\text{sign}\left(\sum_{i=1}^{N}\alpha_it_iK(\mathbf{x},\mathbf{x}_i)+b\right)$ Non-kernel based PSVM: $f(\mathbf{x})=\text{sign}\left(\boldsymbol{\phi}(\mathbf{x})\mathbf{Z}^T\left(\frac{\mathbf{I}}{C}+\mathbf{Z}\mathbf{Z}^T+\mathbf{T}\mathbf{T}^T\right)^{-1}\vec{\mathbf{1}}+b\right)$ where $b=\left(\frac{\mathbf{I}}{C}+\mathbf{Z}\mathbf{Z}^T+\mathbf{T}\mathbf{T}^T\right)^{-1}\vec{\mathbf{1}}\mathbf{T}$	$f(\mathbf{x})=\text{sign}\left(\sum_{i=1}^{N}\alpha_it_iK(\mathbf{x},\mathbf{x}_i)+b\right)$

TABLE III
SPECIFICATION OF MULTICLASS CLASSIFICATION PROBLEMS

Datasets	# train	# test	# features	# classes	Random Perm
Iris	100	50	4	3	Yes
Glass	142	72	9	6	Yes
Wine	118	60	13	3	Yes
Ecoli	224	112	7	8	Yes
Vowel	528	462	10	11	No
Vehicle	564	282	18	4	Yes
Segment	1540	770	19	7	Yes
Satimage	4435	2000	36	6	No
DNA	2000	1186	180	3	No
Letter	13333	6667	16	26	Yes
Shuttle	43500	14500	9	7	No
USPS	7291	2007	256	10	No

TABLE IV
SPECIFICATION OF REGRESSION PROBLEMS

Datasets	# train	# test	# features	Random Perm
Baskball	64	32	4	Yes
Cloud	72	36	9	Yes
Autoprice	106	53	9	Yes
Strike	416	209	6	Yes
Pyrim	49	25	27	Yes
Bodyfat	168	84	14	Yes
Cleveland	202	101	13	Yes
Housing	337	169	13	Yes
Balloon	1334	667	2	Yes
Quake	1452	726	3	Yes
Space-ga	2071	1036	6	Yes
Abalone	2784	1393	8	Yes

2) data sets with relatively small size and medium dimensions, e.g., *Pyrim*, *Housing* [47], *Bodyfat*, and *Cleveland* [48];
3) data sets with relatively large size and low dimensions, e.g., *Balloon*, *Quake*, *Space-ga* [48], and *Abalone* [47].

Column "random perm" in Tables II–IV shows whether the training and testing data of the corresponding data sets are reshuffled at each trial of simulation. If the training and testing data of the data sets remain fixed for all trials of simulations, it is marked "No." Otherwise, it is marked "Yes."

B. Simulation Environment Settings

The simulations of different algorithms on all the data sets except for *Adult*, *Letter*, *Shuttle*, and *USPS* data sets are carried out in MATLAB 7.0.1 environment running in Core 2 Quad, 2.66-GHZ CPU with 2-GB RAM. The codes used for SVM and LS-SVM are downloaded from [55] and [56], respectively.

Simulations on large data sets (e.g., *Adult*, *Letter*, *Shuttle*, and *USPS* data sets) are carried out in a high-performance computer with 2.52-GHz CPU and 48-GB RAM. The symbol "*" marked in Tables VI and VII indicates that the corresponding data sets are tested in such a high-performance computer.

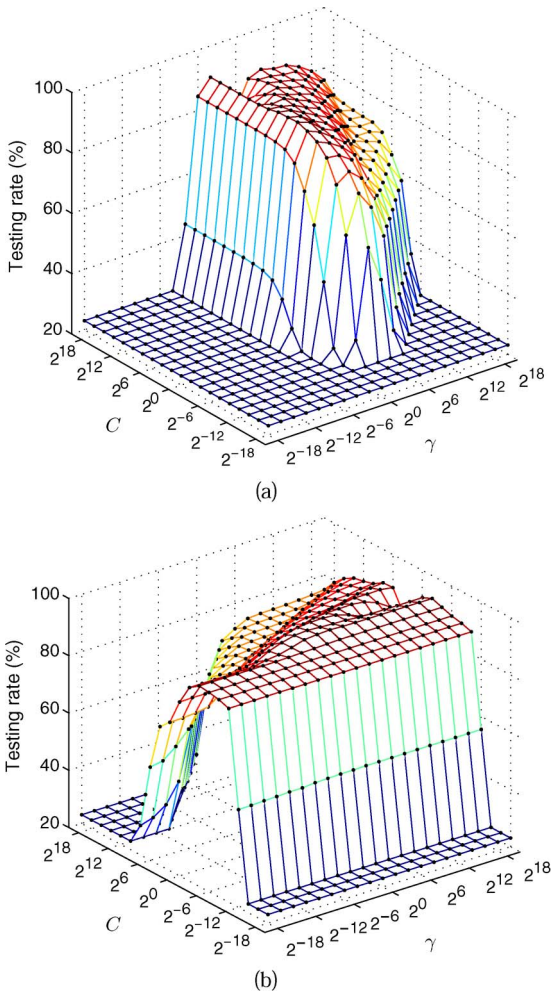


Fig. 2. Performances of LS-SVM and ELM with Gaussian kernel are sensitive to the user-specified parameters (C, γ): An example on Satimage data set. (a) LS-SVM with Gaussian kernel. (b) ELM with Gaussian kernel.

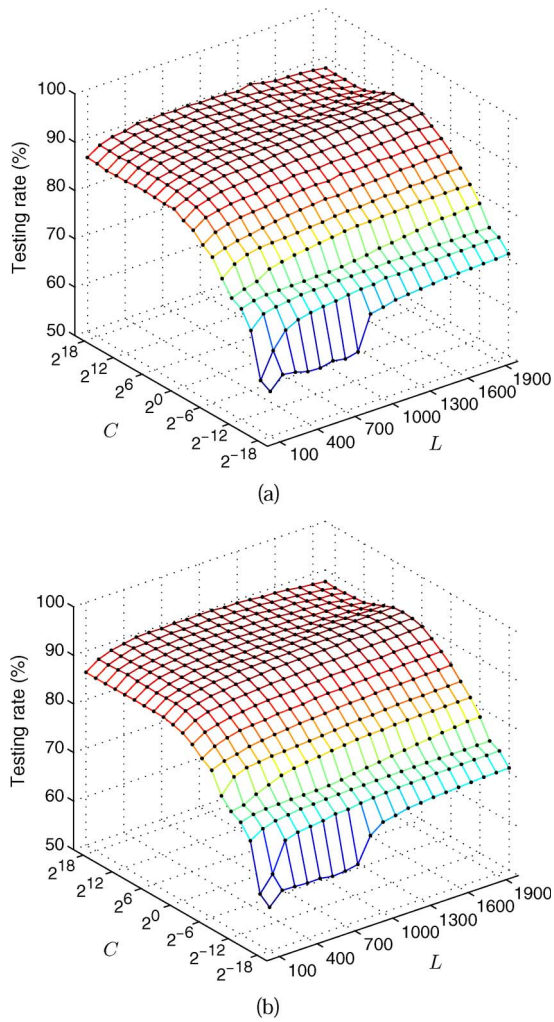


Fig. 3. Performance of ELM (with Sigmoid additive node and multiquadric RBF node) is not very sensitive to the user-specified parameters (C, L), and good testing accuracies can be achieved as long as L is large enough: An example on Satimage data set. (a) ELM with Sigmoid additive node. (b) ELM with multiquadric RBF node.

C. User-Specified Parameters

The popular Gaussian kernel function $K(\mathbf{u}, \mathbf{v}) = \exp(-\gamma\|\mathbf{u} - \mathbf{v}\|^2)$ is used in SVM, LS-SVM, and ELM. ELM performance is also tested in the cases of Sigmoid type of additive hidden node and multiquadric RBF hidden node.

In order to achieve good generalization performance, the cost parameter C and kernel parameter γ of SVM, LS-SVM, and ELM need to be chosen appropriately. We have tried a wide range of C and γ. For each data set, we have used 50 different values of C and 50 different values of γ, resulting in a total of 2500 pairs of (C, γ). The 50 different values of C and γ are $\{2^{-24}, 2^{-23}, \ldots, 2^{24}, 2^{25}\}$.

It is known that the performance of SVM is sensitive to the combination of (C, γ). Similar to SVM, the generalization performance of LS-SVM and ELM with Gaussian kernel depends closely on the combination of (C, γ) as well (see Fig. 2 for the performance sensitivity of LS-SVM and ELM with Gaussian kernel on the user-specified parameters (C, γ)). The best generalization performance of SVM, LS-SVM, and ELM with Gaussian kernel is usually achieved in a very narrow range of such combinations. Thus, the best combination of (C, γ) of SVM, LS-SVM, and ELM with Gaussian kernel needs to be chosen for each data set.

For ELM with Sigmoid additive hidden node and multiquadric RBF hidden node, $\mathbf{h}(\mathbf{x}) = [G(\mathbf{a}_1, b_1, \mathbf{x}), \ldots, G(\mathbf{a}_L, b_L, \mathbf{x})]$, where $G(\mathbf{a}, b, \mathbf{x}) = 1/(1 + \exp(-(\mathbf{a} \cdot \mathbf{x} + b)))$ for Sigmoid additive hidden node or $G(\mathbf{a}, b, \mathbf{x}) = (\|\mathbf{x} - \mathbf{a}\|^2 + b^2)^{1/2}$ for multiquadric RBF hidden node. All the hidden-node parameters $(\mathbf{a}_i, b_i)_{i=1}^{L}$ are randomly generated based on uniform distribution. The user-specified parameters are (C, L), where C is chosen from the range $\{2^{-24}, 2^{-23}, \ldots, 2^{24}, 2^{25}\}$. Seen from Fig. 3, ELM can achieve good generalization performance as long as the number of hidden nodes L is large enough. In all our simulations on ELM with Sigmoid additive hidden node and multiquadric RBF hidden node, $L = 1000$. In other words, the performance of ELM with Sigmoid additive hidden node and multiquadric RBF hidden node is not sensitive to the number of hidden nodes L. Moreover, L need not be specified by users; instead, users only need to specify one parameter: C.

Fifty trials have been conducted for each problem. Simulation results, including the average testing accuracy, the corresponding standard deviation (Dev), and the training times, are given in this section.

TABLE V
PARAMETERS OF THE CONVENTIONAL SVM, LS-SVM, AND ELM

Datasets	SVM (Gaussian Kernel)		LSSVM (Gaussian Kernel)		Extreme Learning Machine					
					Gaussian Kernel		Sigmoid Additive Node		Multiquadrics RBF Node	
	C	γ	C	γ	C	γ	C	L	C	L
Binary class datasets										
Diabetes	2^{10}	2^{4}	2^{10}	2^{10}	2^{10}	2^{5}	2^{-2}	1000	2^{-2}	1000
Australian Credit	2^{14}	2^{4}	2^{7}	2^{10}	2^{10}	2^{9}	2^{-1}	1000	2^{-1}	1000
Liver	2^{18}	2^{4}	2^{5}	2^{7}	2^{8}	2^{5}	2^{1}	1000	2^{2}	1000
Banana	2^{25}	2^{2}	2^{14}	2^{2}	2^{0}	2^{4}	2^{22}	1000	2^{2}	1000
Colon	2^{12}	2^{6}	2^{4}	2^{16}	2^{7}	2^{20}	2^{1}	1000	2^{-7}	1000
Colon (Gene Sel)	2^{2}	2^{4}	2^{0}	2^{12}	2^{7}	2^{20}	2^{-14}	1000	2^{-13}	1000
Leukemia	2^{12}	2^{10}	2^{10}	2^{6}	2^{15}	2^{20}	2^{17}	1000	2^{13}	1000
Leukemia (Gene Sel)	2^{8}	2^{8}	2^{10}	2^{10}	2^{15}	2^{0}	2^{6}	1000	2^{-7}	1000
Brightdata	2^{12}	2^{4}	2^{2}	2^{3}	2^{4}	2^{-4}	2^{5}	1000	2^{5}	1000
Dimdata	2^{14}	2^{4}	2^{3}	2^{3}	2^{5}	2^{8}	2^{1}	1000	2^{1}	1000
Mushroom	2^{20}	2^{0}	2^{3}	2^{3}	2^{13}	2^{4}	2^{19}	1000	2^{19}	1000
Adult	2^{2}	2^{2}	2^{0}	2^{7}	2^{25}	2^{18}	2^{-6}	1000	2^{-6}	1000
Multi-class datasets										
Iris	1	2^{-2}	2^{14}	2^{6}	2^{0}	2^{0}	2^{5}	1000	2^{-3}	1000
Glass	1	2^{-2}	2^{2}	2^{2}	2^{5}	2^{0}	2^{3}	1000	2^{2}	1000
Wine	1	1	2^{16}	2^{14}	2^{5}	2^{3}	2^{0}	1000	2^{-1}	1000
Ecoli	2^{16}	2^{5}	2^{5}	2^{5}	2^{22}	2^{12}	2^{2}	1000	2^{0}	1000
Vowel	1	1	2^{8}	2^{2}	2^{5}	2^{-1}	2^{1}	1000	2^{0}	1000
Vehicle	2^{14}	2^{2}	2^{33}	2^{20}	2^{6}	2^{3}	2^{7}	1000	2^{10}	1000
Segment	2^{25}	2^{5}	2^{6}	2^{2}	2^{13}	2^{-5}	2^{10}	1000	2^{17}	1000
Satimage	1	1	2^{8}	2^{2}	2^{4}	2^{-2}	2^{7}	1000	2^{12}	1000
DNA	2^{18}	2^{12}	2^{12}	2^{8}	2^{6}	2^{6}	2^{-7}	1000	2^{1}	1000
Letter	2^{10}	2^{-4}	2^{8}	2^{0}	2^{3}	2^{-2}	2^{10}	1000	2^{24}	1000
shuttle	2^{10}	2^{-2}	2^{18}	2^{-4}	2^{20}	2^{-10}	2^{25}	1000	2^{25}	1000
USPS	2^{8}	2^{0}	2^{5}	2^{8}	2^{4}	2^{8}	2^{10}	1000	2^{20}	1000
Regression datasets										
Baskball	2^{0}	2^{0}	2^{0}	2^{3}	2^{0}	2^{0}	2^{0}	1000	2^{-4}	1000
Cloud	2^{2}	2^{0}	2^{-18}	2^{-17}	2^{15}	2^{9}	2^{-5}	1000	2^{-14}	1000
Autoprice	2^{13}	2^{5}	2^{5}	2^{2}	2^{6}	2^{4}	2^{-1}	1000	2^{6}	1000
Strike	2^{0}	2^{-4}	2^{0}	2^{-2}	2^{-1}	2^{5}	2^{-5}	1000	2^{8}	1000
Pyrim	2^{10}	2^{8}	2^{5}	2^{7}	2^{2}	2^{6}	2^{-3}	1000	2^{3}	1000
Bodyfat	2^{25}	2^{7}	2^{25}	2^{20}	2^{12}	2^{16}	2^{0}	1000	2^{6}	1000
Cleveland	2^{2}	2^{2}	2^{8}	2^{14}	2^{13}	2^{15}	2^{-3}	1000	2^{1}	1000
Housing	2^{4}	2^{2}	2^{4}	2^{4}	2^{2}	2^{8}	2^{5}	1000	2^{7}	1000
Balloon	2^{4}	2^{-2}	2^{10}	2^{0}	2^{-6}	2^{10}	2^{20}	1000	2^{15}	1000
Quake	2^{5}	2^{5}	2^{5}	2^{14}	2^{5}	2^{14}	2^{0}	1000	2^{10}	1000
Space-ga	2^{8}	2^{-1}	2^{8}	2^{2}	2^{2}	2^{20}	2^{4}	1000	2^{13}	1000
Abalone	2^{1}	2^{-1}	2^{4}	2^{4}	2^{0}	2^{0}	2^{0}	1000	2^{0}	1000

The user-specified parameters chosen in our simulations are given in Table V.

D. Performance Comparison on XOR Problem

The performance of SVM, LS-SVM, and ELM has been tested in the XOR problem which has two training samples in each class. The aim of this simulation is to verify whether ELM can handle some rare cases such as the cases with extremely few training data sets. Fig. 4 shows the boundaries of different classifiers in XOR problem. It can be seen that, similar to SVM and LS-SVM, ELM is able to solve the XOR problem well. User-specified parameters used in this XOR problem are chosen as follows: (C, γ) for SVM is $(2^{10}, 2^{0})$, (C, γ) for LS-SVM is $(2^{4}, 2^{14})$, (C, γ) for ELM with Gaussian kernel is $(2^{5}, 2^{15})$, and (C, L) for ELM with Sigmoid additive hidden node is $(2^{0}, 3000)$.

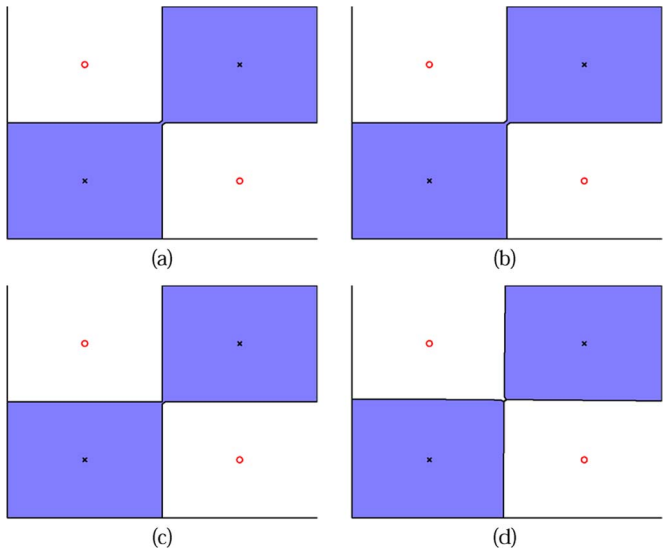


Fig. 4. Separating boundaries of different classifiers in XOR problem. (a) SVM. (b) LS-SVM. (c) ELM (Gaussian kernel). (d) ELM (Sigmoid additive node).

E. Performance Comparison on Real-World Benchmark Data sets

Tables VI–VIII show the performance comparison of SVM, LS-SVM, and ELM with Gaussian kernel, random Sigmoid hidden nodes, and multiquadric RBF nodes. It can be seen that ELM can always achieve comparable performance as SVM and LS-SVM with much faster learning speed. Seen from Tables VI–VIII, different output functions of ELM can be used in different data sets in order to have efficient implementation

TABLE VI
PERFORMANCE COMPARISON OF SVM, LS-SVM, AND ELM: BINARY CLASS DATA SETS

Datasets	SVM			LSSVM			Extreme Learning Machine								
							Gaussian Kernel			Sigmoid Additive Node			Multiquadrics RBF Node		
	Testing		Training	Testing		Training	Testing		Training	Testing		Training	Testing		Training
	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)
										$f(\mathbf{x}) = \text{sign}\left(\mathbf{h}(\mathbf{x})\mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T}\right)$					
Diabete	76.97	2.70	0.6759	77.18	2.07	0.1406	**77.52**	2.46	0.0528	77.95	2.18	0.2075	78.09	2.17	0.2306
Australian	85.79	2.03	0.7042	85.91	1.85	0.1250	**86.29**	1.43	0.0403	86.18	1.80	0.1709	86.70	1.90	0.1691
Credit															
Liver	72.65	3.61	0.5616	71.98	3.37	0.0625	72.14	3.74	0.0066	73.01	3.78	0.0528	71.57	0.04	0.0531
Banana	89.84	0	0.8120	89.63	0	0.0620	89.83	0	0.0469	89.61	0.05	0.1350	89.30	0.01	0.1416
Colon	84.38	0	0.1617	81.25	0	0.4531	**84.38**	0	0.0031	81.63	3.32	0.1103	82.13	1.55	0.1472
Colon	84.38	0	0.0462	87.50	0	0.0469	**90.63**	0	0.0010	89.62	2.85	0.0075	87.50	0	0.0072
(Gene Sel)															
Leukemia	82.34	0	1.007	**85.29**	0	1.703	82.35	0	0.0309	80.08	3.85	0.4288	83.47	3.41	0.5550
Leukemia	100	0	0.0494	100	0	0.0625	100	0	0.0003	100	0	0.0075	100	0	0.0106
(Gene Sel)															
										$f(\mathbf{x}) = \text{sign}\left(\mathbf{h}(\mathbf{x})\left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}\right)$					
Brightdata	99.46	0.23	1.289	**99.24**	0.17	0.5413	98.91	0.25	0.2984	99.31	0.2	0.7573	99.31	0.17	0.7401
Dimdata	95.85	0.27	0.8908	95.19	0.35	0.5781	**95.89**	0.34	0.2734	95.75	0.29	0.749	95.67	0.26	0.75
Mushroom	89.88	0.43	46.56	88.87	0.41	1.531	88.84	0.36	0.8133	88.91	0.36	1.038	88.88	0.33	1.047
* Adult	84.51	0	29.382	84.79	0	5.5703	84.58	0	3.3116	84.59	0.05	0.4246	84.51	0.02	0.5682

TABLE VII
PERFORMANCE COMPARISON OF SVM, LS-SVM, AND ELM: MULTICLASS DATA SETS

Datasets	SVM			LSSVM			Extreme Learning Machine								
							Gaussian Kernel			Sigmoid Additive Node			Multiquadrics RBF Node		
	Testing		Training	Testing		Training	Testing		Training	Testing		Training	Testing		Training
	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)
										$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T}$					
Iris	95.12	2.45	0.075	96.28	2.36	0.0021	96.04	2.37	0.0022	97.6	2.29	0.0156	97.33	2.12	0.0161
Glass	67.83	4.67	0.2871	67.22	5.04	0.0097	**68.41**	4.81	0.0026	67.12	4.99	0.0262	66.89	4.97	0.0264
Wine	98.37	1.41	0.075	97.63	1.82	0.0043	**98.48**	1.7	0.0019	98.47	1.81	0.0222	98.57	1.26	0.0206
Ecoli	86.56	3.65	0.2469	85.93	2.82	0.0244	**87.48**	2.8	0.008	87.23	2.88	0.053	87.79	2.74	0.054
Vowel	56.28	0	2.172	52.81	0	0.3290	**58.66**	0	0.0688	53.73	0.91	0.2187	54.75	0.89	0.2231
Vehicle	84.37	1.71	1.5144	83.19	1.93	0.2029	83.16	1.89	0.0831	83.48	1.78	0.2557	83.95	1.85	0.2547
										$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}$					
Segment	96.53	0.64	14.30	96.12	0.75	4.302	**96.53**	0.54	0.9272	96.07	0.69	1.889	95.54	0.41	1.070
Satimage	89.75	0	698.4	90.05	0	82.67	**92.35**	0	15.70	89.8	0.32	2.808	89.06	0.38	2.803
DNA	92.86	0	7732	93.68	0	6.359	**96.29**	0	2.156	93.81	0.24	1.586	94.81	0.32	1.597
* Letter	92.87	0.26	302.9	93.12	0.27	335.838	**97.41**	0.13	41.89	93.51	0.15	0.7881	93.96	0.15	1.4339
* **shuttle**	*99.74*	*0*	*2864.0*	*99.82*	*0*	**24767.0**	*99.91*	*0*	*4029.0*	*99.64*	*0.01*	**3.3379**	*99.65*	*0.02*	**5.5455**
* **USPS**	*96.14*	*0*	**12460**	*96.76*	*0*	*59.1357*	**98.9**	*0*	*9.2784*	*96.28*	*0.28*	**0.6877**	*97.25*	*0.24*	**0.9008**

in different size of data sets, although any output function can be used in all types of data sets.

Take Shuttle (*large number of training samples*) and USPS (*medium size of data set with high input dimensions*) data sets in Table VII as examples.

1) For Shuttle data sets, ELM with Gaussian kernel and random multiquadric RBF nodes runs 6 and 4466 times faster than LS-SVM, respectively.
2) For USPS data sets, ELM with Gaussian kernel and random multiquadric RBF nodes runs 6 and 65 times faster than LS-SVM, respectively, and runs 1342 and 13 832 times faster than SVM, respectively.

On the other hand, different from LS-SVM which is sensitive to the combinations of parameters (C, γ), ELM with random multiquadric RBF nodes is not sensitive to the unique user-specified parameter C [cf. Fig. 3(b)] and is ease of use in the respective implementations.

Tables VI–VIII particularly highlight the performance comparison between LS-SVM and ELM with Gaussian kernel, and among the comparisons of these two algorithms, apparently, better test results are given in boldface. It can be seen that ELM with Gaussian kernel achieves the same generalization performance in almost all the binary classification and regression cases as LS-SVM at much faster learning speeds; however, ELM usually achieves much better generalization performance in multiclass classification cases (cf. Table VII) than LS-SVM.

Fig. 5 shows the boundaries of different classifiers in Banana case. It can be seen that ELM can classify different classes well.

VI. CONCLUSION

ELM is a learning mechanism for the generalized SLFNs, where learning is made without iterative tuning. The essence of ELM is that the hidden layer of the generalized SLFNs should not be tuned. Different from traditional learning theories on learning, ELM learning theory [14]–[16] shows that if SLFNs $\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta}$ with tunable *piecewise continuous* hidden-layer feature mapping $\mathbf{h}(\mathbf{x})$ can approximate any target continuous

TABLE VIII
PERFORMANCE COMPARISON OF SVM, LS-SVM, AND ELM: REGRESSION DATA SETS

Datasets	SVR			LSSVR			Extreme Learning Machine								
							Gaussian Kernel			Sigmoid Additive Node			Multiquadrics RBF Node		
	Testing		Training	Testing		Training	Testing		Training	Testing		Training	Testing		Training
	RMSE	Dev	Time(s)	RMSE	Dev	Time(s)	RMSE	Dev	Time(s)	RMSE	Dev	Time(s)	RMSE	Dev	Time(s)
										$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})\mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T}$					
Baskball	0.162	0.0138	0.0160	**0.1564**	0.0165	0.0010	0.1617	0.0175	0.0005	0.1629	0.0169	0.0066	0.1630	0.0155	0.0094
Cloud	0.3262	0.0668	0.0151	0.3049	0.0203	0.0008	0.3061	0.0239	0.0013	0.3165	0.0178	0.0063	0.2969	0.0243	0.0130
Autoprice	0.1776	0.0179	0.0620	**0.1601**	0.0217	0.0031	0.1697	0.015	0.0020	0.1710	0.0165	0.0188	0.1678	0.0189	0.0247
Strike	0.2282	0.0078	1.578	0.2479	0.0106	0.0531	**0.2322**	0.0128	0.0358	0.2985	0.0053	0.0791	0.2646	0.0186	0.1651
Pyrim	0.128	0.0315	0.0160	0.1272	0.0388	0.0006	**0.0921**	0.0167	0.0003	0.1194	0.0311	0.0059	0.1486	0.0369	0.0083
Bodyfat	0.0279	0.0081	0.0470	0.0280	0.0129	0.0059	0.0242	0.0131	0.0028	0.0286	0.0083	0.0231	0.0262	0.0113	0.0354
Cleveland	0.1646	0.0067	0.0930	0.1605	0.0071	0.0075	0.1618	0.0088	0.0053	0.1603	0.0070	0.0284	0.1609	0.0093	0.0474
Housing	0.0976	0.0085	0.2040	0.0704	0.0068	0.0343	0.0744	0.0106	0.0197	0.0805	0.0077	0.0616	0.0779	0.0084	0.1005
										$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})\left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}$					
Balloon	0.059	0.0034	11.30	0.0536	0.0044	1.072	0.0516	0.0043	0.6084	0.0553	0.0013	0.7688	0.0588	0.0121	1.344
Quake	0.1797	0.0068	19.59	**0.1446**	0.0079	1.387	0.1712	0.0099	0.6972	0.1649	0.0061	0.7625	0.1696	0.0098	1.607
Space-ga	0.0648	0.0016	52.75	0.0330	0.0008	3.510	0.0338	0.0018	1.770	0.0624	0.0021	1.220	0.0335	0.0013	1.369
Abalone	0.0764	0.0015	113.1	0.0746	0.0021	7.674	0.0768	0.0021	4.112	0.0761	0.0019	1.324	0.0761	0.0023	1.793

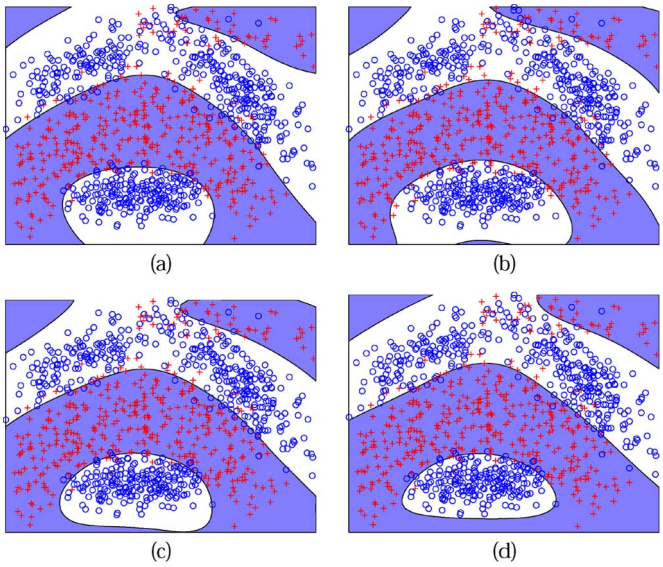


Fig. 5. Separating boundaries of different classifiers in Banana case. (a) SVM. (b) LS-SVM. (c) ELM (Gaussian kernel). (d) ELM (Sigmoid additive node).

functions, tuning is not required in the hidden layer then. All the hidden-node parameters which are supposed to be tuned by conventional learning algorithms can be randomly generated according to any continuous sampling distribution [14]–[16].

This paper has shown that both LS-SVM and PSVM can be simplified by removing the term bias b and the resultant learning algorithms are unified with ELM. Instead of different variants requested for different types of applications, ELM can be applied in regression and multiclass classification applications directly. More importantly, according to ELM theory [14]–[16], ELM can work with a widespread type of feature mappings (including Sigmoid networks, RBF networks, trigonometric networks, threshold networks, fuzzy inference systems, fully complex neural networks, high-order networks, ridge polynomial networks, etc).

ELM requires less human intervention than SVM and LS-SVM/PSVM. If the feature mappings $\mathbf{h}(\mathbf{x})$ are known to users, in ELM, only one parameter C needs to be specified by users. The generalization performance of ELM is not sensitive to the dimensionality L of the feature space (the number of hidden nodes) as long as L is set large enough (e.g., $L \geq 1000$ for all the real-world cases tested in our simulations). Different from SVM, LS-SVM, and PSVM which usually request two parameters (C, γ) to be specified by users, single-parameter setting makes ELM be used easily and efficiently.

If feature mappings are unknown to users, similar to SVM, LS-SVM, and PSVM, kernels can be applied in ELM as well. Different from LS-SVM and PSVM, ELM does not have constraints on the Lagrange multipliers α_i's. Since LS-SVM and ELM have the same optimization objective functions and LS-SVM has some optimization constraints on Lagrange multipliers α_i's, in this sense, LS-SVM tends to obtain a solution which is suboptimal to ELM.

As verified by the simulation results, compared to SVM and LS-SVM ELM achieves similar or better generalization performance for regression and binary class classification cases, and much better generalization performance for multiclass classification cases. ELM has better scalability and runs at much faster learning speed (up to thousands of times) than traditional SVM and LS-SVM.

This paper has also shown that, in theory, ELM can approximate any target continuous function and classify any disjoint regions.

ACKNOWLEDGMENT

The authors would like to thank L. Ljung from Linköpings Universitet, Sweden, for reminding us of possible relationships between ELM and LS-SVM. The authors would also like to thank H. White from the University of California, San Diego, for his constructive and inspiring comments and suggestions on our research on ELM.

REFERENCES

[1] C. Cortes and V. Vapnik, "Support vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[2] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, Jun. 1999.
[3] G. Fung and O. L. Mangasarian, "Proximal support vector machine classifiers," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, San Francisco, CA, 2001, pp. 77–86.
[4] Y.-J. Lee and O. L. Mangasarian, "RSVM: Reduced support vector machines," in *Proc. SIAM Int. Conf. Data Mining*, Chicago, IL, Apr. 5–7, 2001.
[5] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Neural Information Processing Systems 9*,

M. Mozer, J. Jordan, and T. Petscbe, Eds. Cambridge, MA: MIT Press, 1997, pp. 155–161.
[6] G.-B. Huang, K. Z. Mao, C.-K. Siew, and D.-S. Huang, "Fast modular network implementation for support vector machines," *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1651–1663, Nov. 2005.
[7] R. Collobert, S. Bengio, and Y. Bengio, "A parallel mixtures of SVMs for very large scale problems," *Neural Comput.*, vol. 14, no. 5, pp. 1105–1114, May 2002.
[8] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2002.
[9] J. A. K. Suykens and J. Vandewalle, "Multiclass least squares support vector machines," in *Proc. IJCNN*, Jul. 10–16, 1999, pp. 900–903.
[10] T. Van Gestel, J. A. K. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, and J. Vandewalle, "Multiclass LS-SVMs: Moderated outputs and coding-decoding schemes," *Neural Process. Lett.*, vol. 15, no. 1, pp. 48–58, Feb. 2002.
[11] Y. Tang and H. H. Zhang, "Multiclass proximal support vector machines," *J. Comput. Graph. Statist.*, vol. 15, no. 2, pp. 339–355, Jun. 2006.
[12] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proc. IJCNN*, Budapest, Hungary, Jul. 25–29, 2004, vol. 2, pp. 985–990.
[13] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, Dec. 2006.
[14] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.
[15] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16–18, pp. 3056–3062, Oct. 2007.
[16] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, no. 16–18, pp. 3460–3468, Oct. 2008.
[17] X. Tang and M. Han, "Partial Lanczos extreme learning machine for single-output regression problems," *Neurocomputing*, vol. 72, no. 13–15, pp. 3066–3076, Aug. 2009.
[18] Q. Liu, Q. He, and Z. Shi, "Extreme support vector machine classifier," *Lecture Notes in Computer Science*, vol. 5012, pp. 222–233, 2008.
[19] B. Frénay and M. Verleysen, "Using SVMs with randomised feature spaces: An extreme learning approach," in *Proc. 18th ESANN*, Bruges, Belgium, Apr. 28–30, 2010, pp. 315–320.
[20] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "OP-ELM: Optimally pruned extreme learning machine," *IEEE Trans. Neural Netw.*, vol. 21, no. 1, pp. 158–162, Jan. 2010.
[21] W. Deng, Q. Zheng, and L. Chen, "Regularized extreme learning machine," in *Proc. IEEE Symp. CIDM*, Mar. 30–Apr. 2, 2009, pp. 389–395.
[22] K.-A. Toh, "Deterministic neural classification," *Neural Comput.*, vol. 20, no. 6, pp. 1565–1595, Jun. 2008.
[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagation errors," *Nature*, vol. 323, pp. 533–536, 1986.
[24] G.-B. Huang, Q.-Y. Zhu, K. Z. Mao, C.-K. Siew, P. Saratchandran, and N. Sundararajan, "Can threshold networks be trained directly?" *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 3, pp. 187–191, Mar. 2006.
[25] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate on-line sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.
[26] M.-B. Li, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "Fully complex extreme learning machine," *Neurocomputing*, vol. 68, pp. 306–314, Oct. 2005.
[27] G. Feng, G.-B. Huang, Q. Lin, and R. Gay, "Error minimized extreme learning machine with growth of hidden nodes and incremental learning," *IEEE Trans. Neural Netw.*, vol. 20, no. 8, pp. 1352–1357, Aug. 2009.
[28] H.-J. Rong, G.-B. Huang, N. Sundararajan, and P. Saratchandran, "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 4, pp. 1067–1072, Aug. 2009.
[29] G.-B. Huang, X. Ding, and H. Zhou, "Optimization method based extreme learning machine for classification," *Neurocomputing*, vol. 74, no. 1–3, pp. 155–163, Dec. 2010.
[30] J. A. K. Suykens and J. Vandewalle, "Training multilayer perceptron classifier based on a modified support vector method," *IEEE Trans. Neural Netw.*, vol. 10, no. 4, pp. 907–911, Jul. 1999.
[31] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. New York: Spartan Books, 1962.
[32] R. Fletcher, *Practical Methods of Optimization: Volume 2 Constrained Optimization*. New York: Wiley, 1981.
[33] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Englewood Cliffs, NJ: Prentice-Hall, 1999.
[34] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network," *IEEE Trans. Inf. Theory*, vol. 44, no. 2, pp. 525–536, Mar. 1998.
[35] D. Serre, *Matrices: Theory and Applications*. New York: Springer-Verlag, 2002.
[36] C. R. Rao and S. K. Mitra, *Generalized Inverse of Matrices and Its Applications*. New York: Wiley, 1971.
[37] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, Feb. 1970.
[38] G.-B. Huang, Y.-Q. Chen, and H. A. Babri, "Classification ability of single hidden layer feedforward neural networks," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 799–801, May 2000.
[39] G.-B. Huang, M.-B. Li, L. Chen, and C.-K. Siew, "Incremental extreme learning machine with fully complex hidden nodes," *Neurocomputing*, vol. 71, no. 4–6, pp. 576–583, Jan. 2008.
[40] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Scientific, 2002.
[41] M. Espinoza, J. A. K. Suykens, and B. De Moor, "Fixed-size least squares support vector machines: A large scale application in electrical load forecasting," *Comput. Manage. Sci.—Special Issue on Support Vector Machines*, vol. 3, no. 2, pp. 113–129, Apr. 2006.
[42] M. Espinoza, J. A. K. Suykens, R. Belmans, and B. De Moor, "Electric load forecasting—Using kernel based modeling for nonlinear system identification," *IEEE Control Syst. Mag.—Special Issue on Applications of System Identification*, vol. 27, no. 5, pp. 43–57, Oct. 2007.
[43] K. D. Brabanter, J. D. Brabanter, J. A. K. Suykens, and B. De Moor, "Optimized fixed-size kernel models for large data sets," *Comput. Statist. Data Anal.*, vol. 54, no. 6, pp. 1484–1504, Jun. 2010.
[44] P. Karsmakers, K. Pelckmans, K. D. Brabanter, H. V. Hamme, and J. A. K. Suykens, "Sparse conjugate directions pursuit with application to fixed-size kernel models," *Mach. Learn.*, vol. 85, no. 1–2, pp. 109–148, 2011.
[45] K.-A. Toh, Q.-L. Tran, and D. Srinivasan, "Benchmarking a reduced multivariate polynomial pattern classifier," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 740–755, Jun. 2004.
[46] T. Poggio, S. Mukherjee, R. Rifkin, A. Rakhlin, and A. Verri, " *b*," Artif. Intell. Lab., MIT, Cambridge, MA, A.I. Memo No. 2001-011, CBCL Memo 198, 2001.
[47] C. L. Blake and C. J. Merz, "UCI Repository of Machine Learning Databases," Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, 1998. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html
[48] M. Mike, "Statistical Datasets," Dept. Statist., Univ. Carnegie Mellon, Pittsburgh, PA, 1989. [Online]. Available: http://lib.stat.cmu.edu/datasets/
[49] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast kernel classifiers with online and active learning," *J. Mach. Learn. Res.*, vol. 6, pp. 1579–1619, Sep. 2005.
[50] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, May 1994.
[51] J. Li and H. Liu, "Kent Ridge Bio-Medical Data Set Repository," School Comput. Eng., Nanyang Technol. Univ., Singapore, 2004. [Online]. Available: http://levis.tongji.edu.cn/gzli/data/mirror-kentridge.html
[52] S. C. Odewahn, E. B. Stockwell, R. L. Pennington, R. M. Humphreys, and W. A. Zumach, "Automated star/galaxy discrimination with neural networks," *Astron. J.*, vol. 103, no. 1, pp. 318–331, Jan. 1992.
[53] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander, "Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, no. 5439, pp. 531–537, Oct. 1999.
[54] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1226–1238, Aug. 2005.
[55] S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy, "SVM and Kernel Methods Matlab Toolbox," Perception Systémes et Information, INSA de Rouen, Rouen, France, 2005. [Online]. Available: http://asi.insa-rouen.fr/enseignants/~arakotom/toolbox/index.html
[56] K. Pelckmans, J. A. K. Suykens, T. Van Gestel, J. De Brabanter, L. Lukas, B. Hamers, B. De Moor, and J. Vandewalle, "LS-SVMLab Toolbox," Dept. Elect. Eng., ESAT-SCD-SISTA, Leuven, Belgium, 2002. [Online]. Available: http://www.esat.kuleuven.be/sista/lssvmlab/

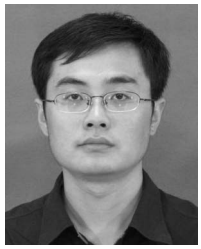
Guang-Bin Huang (M'98–SM'04) received the B.Sc degree in applied mathematics and M.Eng degree in computer engineering from Northeastern University, Shenyang, China, in 1991 and 1994, respectively, and the Ph.D. degree in electrical engineering from Nanyang Technological University, Singapore, in 1999.

During undergraduate period, he also concurrently studied in the Applied Mathematics Department and Wireless Communication Department, Northeastern University, China. From June 1998 to May 2001, he was a Research Fellow with Singapore Institute of Manufacturing Technology (formerly known as Gintic Institute of Manufacturing Technology), where he has led/implemented several key industrial projects (e.g., Chief Designer and Technical Leader of Singapore Changi Airport Cargo Terminal Upgrading Project, etc). Since May 2001, he has been an Assistant Professor and Associate Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His current research interests include machine learning, computational intelligence, and extreme learning machines.

Dr. Huang serves as an Associate Editor of *Neurocomputing* and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B.

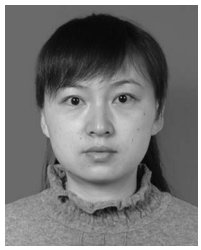
Hongming Zhou received the B.Eng. degree from Nanyang Technological University, Singapore, in 2009, where he is currently working toward the Ph.D. degree in the School of Electrical and Electronic Engineering.

His research interests include extreme learning machines, neural networks, and support vector machines.



Xiaojian Ding received the B.Sc. degree in applied mathematics and the M.Sc. degree in computer engineering from Xi'an University of Technology, Xi'an, China, in 2003 and 2006, respectively, and the Ph.D. degree from the School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, in 2010. He studied in the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, under the award from the Chinese Scholarship Council of China from August 2009 to August 2010.

His research interests include extreme learning machines, neural networks, pattern recognition, and machine learning.



Rui Zhang received the B.Sc. and M.Sc. degrees in mathematics from Northwest University, Xi'an, China, in 1994 and 1997, respectively. She is currently working toward the Ph.D. degree in the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

From August 2004 to January 2005, she was a Visiting Scholar with the Department of Mathematics, University of Illinois at Champaign–Urbana, Urbana. Since 1997, she has been with the Department of Mathematics, Northwest University, where she is currently an Associate Professor. Her current research interests include extreme learning machines, machine learning, and neural networks.