Contents lists available at ScienceDirect

Neurocomputing

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

Optimization method based extreme learning machine for classification ☆

Guang-Bin Huang [a,*], Xiaojian Ding [a,b], Hongming Zhou [a]

[a] School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798, Singapore
[b] School of Electronic and Information Engineering, Xi'an Jiaotong University, Shaanxi, Xi'an 710049, China

A R T I C L E   I N F O

Article history:
Received 8 January 2010
Received in revised form
7 February 2010
Accepted 7 February 2010
Communicated by D. Wang
Available online 10 May 2010

Keywords:
Extreme learning machine
Support vector machine
Support vector network
ELM kernel
ELM feature space
Equivalence between ELM and SVM
Maximal margin
Minimal norm of weights
Primal and dual ELM networks

A B S T R A C T

Extreme learning machine (ELM) as an emergent technology has shown its good performance in regression applications as well as in large dataset (and/or multi-label) classification applications. The ELM theory shows that the hidden nodes of the "generalized" single-hidden layer feedforward networks (SLFNs), which need not be neuron alike, can be randomly generated and the universal approximation capability of such SLFNs can be guaranteed. This paper further studies ELM for classification in the aspect of the standard optimization method and extends ELM to a specific type of "generalized" SLFNs—support vector network. This paper shows that: (1) under the ELM learning framework, SVM's maximal margin property and the minimal norm of weights theory of feedforward neural networks are actually consistent; (2) from the standard optimization method point of view ELM for classification and SVM are equivalent but ELM has less optimization constraints due to its special separability feature; (3) as analyzed in theory and further verified by the simulation results, ELM for classification tends to achieve better generalization performance than traditional SVM. ELM for classification is less sensitive to user specified parameters and can be implemented easily.

1. Introduction

There are three main approaches in the training of feedforward networks: (1) gradient-descent based (e.g, backpropagation (BP) method [1] for multi-layer feedforward "neural" networks); (2) least square based (e.g. extreme learning machines (ELMs) [2–6] for the "generalized" single-hidden layer feedforward networks (SLFNs)); (3) standard optimization method based (e.g. support vector machines (SVMs) [7] for a specific type of single-hidden layer feedforward networks, the so-called support vector network).

Support vector machines (SVMs) [7] have been extensively used in widespread applications. The main learning feature of SVM is that the standard optimization method is used to find the solution of maximizing the separating margin of two different classes while minimizing the training errors.

Extreme learning machines (ELMs) were originally developed for the single-hidden layer feedforward neural networks [2–4] and then extended to the "generalized" single-hidden layer feedforward networks (SLFNs) which may not be neuron alike [5,6]. ELM proposes to apply random computational nodes in the hidden layer, which may be independent of the training data. ELM [2,3] and its variants [4–6,8–12] mainly focus on the function approximation applications. Different from traditional learning algorithms for neural networks ELM not only tends to reach the smallest training error but also the smallest norm of output weights. Bartlett's theory [13] shows that for feedforward neural networks reaching smaller training error the smaller the norm of weights is, the better generalization performance the networks tend to have.

Intuitively speaking, SVMs and ELMs do not seem to have much direct relevance although one of the main reasons behind the extension of ELM to "generalized" SLFNs is to possibly apply ELM learning approach in SVM in some way [5,6]. It is also not clear whether there is any close relationship between SVM's maximal separating margin of two different classes and Bartlett's minimal size of weights theory on feedforward neural networks [13]. However, Liu et al. [14] and Frénay and Verleysen [15] have made a significant contribution showing 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.[1] Random ELM kernels can be applied in SVM,

☆This research was sponsored by the grant from Academic Research Fund (AcRF) Tier 1 under project No. RG 22/08 (M52040128). X. Ding's research was also under the sponsorship from Chinese Scholarship Council (CSC), China.

* Corresponding author. Tel.: +65 6790 4489; fax: +65 6793 3318.
E-mail addresses: egbhuang@ntu.edu.sg, gbhuang@ieee.org (G.-B. Huang).

[1] Liu et al. [14] suggests to apply ELM kernel in SVMs, especially studies PSVM [16] with ELM kernel. Later Frénay and Verleysen [15] shows that the normalized ELM kernel can also be applied in the traditional SVM.

0925-2312/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2010.02.019

which validates the correctness of ELM's theory on the "generalized" SLFNs [5,6]. Their proposed SVM with ELM kernel and the conventional SVM have the same optimization constraints. Their work is exciting and has resolved a challenging issue whether ELM can be used for SVM in real applications, however, the reason why ELM for SVM can achieve better generalization performance than the conventional SVM has not been answered clearly.

Different from the earlier work on ELM for SVM [14,15] which effectively and simply apply ELM kernels in SVM, this paper studies ELM for classification in the aspect of the standard optimization method and further shows that (1) SVM's maximal separating margin property and the ELM's minimal norm of output weights property are actually consistent and with ELM framework SVM's maximal separating margin property and Barlett's theory on feedforward neural networks remain consistent; (2) in theory ELM can be linearly extended to SVMs (but with less optimization constraints) instead of only replacing SVM kernels with ELM kernels, and thus the learning and implementation of SVMs can be made much simpler and more efficient indeed; (3) since according to the ELM theories [2–6] all the training data are linearly separable by a hyperplane passing through the origin with probability one in the ELM feature space, ELM for classification tends to achieve better generalization performance than SVM.

2. Brief of support vector machine

In order to understand how Cortes and Vapnik's SVM [7] was proposed it may be useful to mention Rosenblatt's feature space on perceptrons [17] first.[2] As shown in Cortes and Vapnik's SVM [7], SVM can be seen as a specific type of SLFNs, the so-called support vector networks.

2.1. Rosenblatt's feature space

In 1962 Rosenblatt [17] investigated perceptrons (multi-layer feedforward neural networks). Due to lack of learning algorithms to adjust all the weights of perceptrons, Rosenblatt suggested a learning mechanism where only the weights of the connections from the last hidden layer to the output layer were adjusted. After all the rest weights fixed the input data are actually transformed into a feature space Z of the last hidden layer. In this feature space a linear decision function is constructed:

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

where $z_i(\mathbf{x})$ is the output of the i-th 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 [7] proposed the support vector machine which maps the data from the input space to some high dimensional feature space Z through some non-linear mapping chosen *a priori*. Standard optimization methods are used to find the separating hyperplane which maximizes the separating margins of two different classes in the feature space.

2.2. Maximal separating margins in 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 $t_i \in \{-1,1\}$, these training data are usually not separable in the input space in most cases. If the training data cannot be separated by a linear decision function $\mathbf{w}\cdot\mathbf{x}+b=0$ one can map the training data $\mathbf{x}_i$ from the input space to a feature space Z through a mapping $\phi(\mathbf{x}) : \mathbf{x}_i \rightarrow \phi(\mathbf{x}_i)$. In this feature space, to separate the training data with a minimal training error we have

$$\begin{aligned} &\mathbf{w}\cdot\phi(\mathbf{x}_i)+b \geq 1-\xi_i \quad \text{if } t_i = 1 \\ &\mathbf{w}\cdot\phi(\mathbf{x}_i)+b \leq -1+\xi_i \quad \text{if } t_i = -1 \end{aligned} \tag{2}$$

that is

$$t_i(\mathbf{w}\cdot\phi(\mathbf{x}_i)+b) \geq 1-\xi_i, \quad i=1,\ldots,N \tag{3}$$

Vectors $\mathbf{x}_i$ for which $t_i(\mathbf{w}\cdot\phi(\mathbf{x}_i)+b)=1$ is termed support vectors. The hyperplane $\mathbf{w}\cdot\phi(\mathbf{x})+b=0$ is the unique one which separates the training data with a maximal margin in the feature space: it maximizes the distance $2/\|\mathbf{w}\|$ between two different classes in the feature space Z.

To maximize such separating margin and to minimize the training error is equivalent to

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

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 theorem [18], to train such a SVM is equivalent to solving the following dual optimization problem

$$\begin{aligned} \text{Minimize}: \quad & L_D = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} t_i t_j \alpha_i \alpha_j \phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i \\ \text{Subject to}: \quad & \sum_{i=1}^{N} t_i\alpha_i = 0 \\ & 0 \leq \alpha_i \leq C, \quad i=1,\ldots,N \end{aligned} \tag{5}$$

where each Lagrange multiplier α_i corresponds to a training example $(\mathbf{x}_i, t_i)$.

We also have

$$\mathbf{w} = \sum_{s=1}^{N_s} \alpha_s t_s \phi(\mathbf{x}_s) \tag{6}$$

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

2.3. SVM kernel functions

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

$$\begin{aligned} \text{Minimize}: \quad & L_D = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} t_i t_j K(\mathbf{x}_i,\mathbf{x}_j)\alpha_i \alpha_j - \sum_{i=1}^{N}\alpha_i \\ \text{Subject to}: \quad & \sum_{i=1}^{N} t_i\alpha_i = 0 \\ & 0 \leq \alpha_i \leq C, \quad i=1,\ldots,N \end{aligned} \tag{7}$$

The SVM kernel function $K(\mathbf{u},\mathbf{v})$ needs to satisfy Mercer's condition [7]. The decision function of SVM is:

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{s=1}^{N_s} \alpha_s t_s K(\mathbf{x},\mathbf{x}_s) + b\right) \tag{8}$$

[2] One may refer to [7] for the details of the relationship between SVM and Rosenblatt's feature space.

3. Brief of extreme learning machine

Extreme learning machine (ELM) [2–4] was originally proposed for the single-hidden layer feedforward neural networks and then extended to the generalized single-hidden layer feedforward networks where the hidden layer needn't be neuron alike [5,6]. In ELM *all* the hidden node parameters are randomly generated (even before ELM sees the training data) without tuning.

3.1. ELM feature space

The output of ELM is

$$f(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) = \boldsymbol{\beta} \cdot h(\mathbf{x}) \tag{9}$$

where β_i is the output weight from the i-th hidden node to the output node and $G(\mathbf{a}_i, b_i, \mathbf{x})$ is the output of the i-th hidden node. $h(\mathbf{x}) = [G(\mathbf{a}_1, b_1, \mathbf{x}), \ldots, G(\mathbf{a}_L, b_L, \mathbf{x})]^T$ is the output vector of the hidden layer with respect to the input $\mathbf{x}$. $h(\mathbf{x})$ actually maps the data from the d-dimensional input space to the L-dimensional hidden layer feature space (*ELM feature space*) H. For the binary classification applications, the decision function of ELM is

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) \right) = \text{sign}(\boldsymbol{\beta} \cdot h(\mathbf{x})) \tag{10}$$

3.2. Minimal norm of output weights

Different from traditional learning algorithms ELM not only tends to reach the smallest training error but also the smallest norm of output weights. According to Bartlett's theory [13], for feedforward neural networks reaching smaller training error, the smaller the norm of weights is, the better generalization performance the networks tends to have. We conjecture that this may be true to generalized SLFNs. ELM is to minimize the training error as well as the norm of the output weights [2,3]:

$$\begin{aligned} &\text{Minimize}: \quad \sum_{i=1}^{N} \|\boldsymbol{\beta} \cdot h(\mathbf{x}_i) - t_i\| \\ &\quad \text{and} \\ &\text{Minimize}: \quad \|\boldsymbol{\beta}\| \end{aligned} \tag{11}$$

Seen from (10), 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: $2/\|\boldsymbol{\beta}\|$ although the minimal norm least square method instead of the standard optimization method was used in the original implementation of ELM [2,3].

4. ELM for classification

This section shows that with the standard optimization method ELM can be linearly extended to SVM (with less optimization constraints) and the implementation of SVM can be made much simpler.

Seen from the decision function of SVM (8): $f(\mathbf{x}) = \text{sign}(\sum_{s=1}^{N_s} \alpha_s t_s K(\mathbf{x}, \mathbf{x}_s) + b)$, apparently SVM can be considered as one of the "generalized" single-hidden layer feedforward networks (SLFNs) which has N_s hidden nodes. The output function of the s-th hidden node of this support vector network is $K(\mathbf{x}, \mathbf{x}_s)$, the output weight between the s-th hidden node to the output node (with bias b) is $\alpha_s t_s$.

Huang et al. [5,6] extends ELM from the SLFNs with neuron type of hidden nodes to the generalized SLFNs where the hidden nodes may not neuron alike. One of the aims of this extension is to possibly apply ELM learning approach in SVM. Since both ELM and SVM work for SLFNs and the hidden layers of both ELM and SVM are not tuned, the learning mechanism of ELM and SVM may be combined in some way: (1) similar to the standard ELM but different from SVM, the random kernels are used; (2) similar to SVM but different from the least square solution of ELM, the standard optimization method is adopted to find the solution of ELM, resulting in support vectors as well.

4.1. Optimization method based solution to ELM

Different from the standard SVM [7], in theory [2–6] any set of distinct training data transformed from the input space to the ELM feature space with the ELM mapping $h(\mathbf{x})$ are linearly separable in the ELM feature space with probability one. However, it is most possible that some testing data may be within the classification margin if zero training error is strictly obtained. In this sense, one may wish to separate the training data with an acceptable minimal training error instead of the zero training error so that the testing error can thus be minimized accordingly

$$\begin{aligned} &\boldsymbol{\beta} \cdot h(\mathbf{x}_i) \geq 1 - \xi_i \quad \text{if } t_i = 1 \\ &\boldsymbol{\beta} \cdot h(\mathbf{x}_i) \leq -1 + \xi_i \quad \text{if } t_i = -1 \end{aligned} \tag{12}$$

That is,

$$t_i \boldsymbol{\beta} \cdot h(\mathbf{x}_i) \geq 1 - \xi_i, \quad i = 1, \ldots, N \tag{13}$$

Thus, from the standard optimization theory point of view, the objective (11) of ELM in minimizing both the training errors and the output weights can be written as

$$\begin{aligned} &\text{Minimize}: \quad L_P = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N} \xi_i \\ &\text{Subject to}: \quad t_i \boldsymbol{\beta} \cdot h(\mathbf{x}_i) \geq 1 - \xi_i, \quad i = 1, \ldots, N \\ &\qquad\qquad\quad \xi_i \geq 0, \quad i = 1, \ldots, N \end{aligned} \tag{14}$$

which is very similar to SVM's optimization problem (4) with two main differences:

(1) Different from the conventional SVM the randomness is adopted in the ELM mapping $h(\mathbf{x})$, that is, all the parameters of $h(\mathbf{x})$ are chosen randomly.
(2) The bias b is not required in the ELM's optimization constrains since in theory the separating hyperplane in the ELM feature space passes through the origin.

The Lagrange function of the primal ELM optimization (14) is

$$L_{\text{ELM}}(\boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\boldsymbol{\beta} \cdot \boldsymbol{\beta} + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i (t_i \boldsymbol{\beta} \cdot h(\mathbf{x}_i) - (1 - \xi_i)) - \sum_{i=1}^{N} \mu_i \xi_i \tag{15}$$

where α_i and μ_i are the Lagrange multipliers and are non-negative values.

In order to find the optimal solutions of (15) we should have

$$\frac{\partial L_{\text{ELM}}(\boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu})}{\partial \boldsymbol{\beta}} = 0 \implies \boldsymbol{\beta} = \sum_{i=1}^{N} \alpha_i t_i h(\mathbf{x}_i) \tag{16}$$

$$\frac{\partial L_{\text{ELM}}(\boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu})}{\partial \boldsymbol{\xi}} = 0 \implies C = \alpha_i + \mu_i, \quad \forall i \tag{17}$$

Substitute (16) and (17) into (15) and to train ELM for classification is then equivalent to solving the following dual optimization problem

$$\text{Minimize}: \quad L_D=\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} t_i t_j \alpha_i \alpha_j h(\mathbf{x}_i)\cdot h(\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i$$
$$\text{Subject to}: \quad 0 \le \alpha_i \le C, \quad i=1,\ldots,N \tag{18}$$

Different from the conventional dual SVM optimization problem (5), the above dual ELM optimization problem does not have the condition $\sum_{i=1}^{N} t_i\alpha_i=0, \forall i$ due to the truth that in theory the separating hyperplane in the ELM feature space tends to pass through the origin.

4.2. ELM kernel

An ELM kernel function can be defined as

$$\begin{aligned} K_{\mathrm{ELM}}(\mathbf{x}_i,\mathbf{x}_j) &= h(\mathbf{x}_i)\cdot h(\mathbf{x}_j) \\ &= [G(\mathbf{a}_1,b_1,\mathbf{x}_i),\ldots,G(\mathbf{a}_L,b_L,\mathbf{x}_i)]^T \cdot [G(\mathbf{a}_1,b_1,\mathbf{x}_j),\ldots,G(\mathbf{a}_L,b_L,\mathbf{x}_j)]^T \end{aligned} \tag{19}$$

where $G(\mathbf{a}, b, \mathbf{x})$ is a nonlinear piecewise continuous function satisfying ELM universal approximation capability theorems [4–6] and $\{(\mathbf{a}_i,b_i)\}_{i=1}^{L}$ are randomly generated according to any continuous probability distribution.

Thus, we have

$$\text{Minimize}: \quad L_D=\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} t_i t_j K_{\mathrm{ELM}}(\mathbf{x}_i,\mathbf{x}_j)\alpha_i \alpha_j - \sum_{i=1}^{N}\alpha_i$$
$$\text{Subject to}: \quad 0 \le \alpha_i \le C, \quad i=1,\ldots,N \tag{20}$$

The decision function of ELM is

$$f(\mathbf{x}) = \mathrm{sign}\left(\sum_{s=1}^{N_s} \alpha_s t_s K_{\mathrm{ELM}}(\mathbf{x},\mathbf{x}_s)\right) \tag{21}$$

4.3. ELM primal and dual networks

Seen from ELM output functions (10) and (21), after applying the optimization method to ELM, we have $\boldsymbol{\beta}=\sum_{s=1}^{N_s}\alpha_s t_s h(\mathbf{x}_s)$, where N_s is the number of the support vectors $\mathbf{x}_s$, and thus we have the primal ELM network as shown in Fig. 1(a).

On the other hand, seen from the ELM network output function (21), after applying the standard optimization method to ELM, we can also obtain a dual network (the so-called support vector network) for ELM (cf. Fig. 1(b)) which has N_s number of hidden nodes each with ELM kernel $K_{\mathrm{ELM}}(\mathbf{x}, \mathbf{x}_s)$ while the output weight linking the s-th hidden layer to the output node is $\alpha_s t_s$.

4.4. Karush–Kuhn–Tucker conditions of ELM

Based on the Karush–Kuhn–Tucker (KKT) theorem [18], the necessary conditions (often referred to as KKT conditions) of the primal ELM optimization (14) should be:

Primal feasibility

$$t_i\boldsymbol{\beta}\cdot h(\mathbf{x}_i)-(1-\xi_i)\ge 0, \quad \forall i \tag{22}$$

$$\xi_i \ge 0, \quad \forall i \tag{23}$$

Dual feasibility

$$\alpha_i \ge 0, \quad \forall i \tag{24}$$

$$\mu_i \ge 0, \quad \forall i \tag{25}$$

Complementary slackness

$$\alpha_i(t_i\boldsymbol{\beta}\cdot h(\mathbf{x}_i)-(1-\xi_i))=0, \quad \forall i \tag{26}$$

$$\mu_i\xi_i=0, \quad \forall i \tag{27}$$

Furthermore,

(1) if $\alpha_i=0$: from (17) and (27) we have $\xi_i=0$. Thus, from (22) we have
$$t_i\boldsymbol{\beta}\cdot h(\mathbf{x}_i)\ge 1 \tag{28}$$
(2) if $0<\alpha_i<C$: from (17) and (27) we have $\xi_i=0$. Thus, from (26) we have
$$t_i\boldsymbol{\beta}\cdot h(\mathbf{x}_i)= 1 \tag{29}$$
(3) if $\alpha_i=C$: from (17) and (27) we have $\xi_i\ge 0$. Thus, from (26) we have
$$t_i\boldsymbol{\beta}\cdot h(\mathbf{x}_i)\le 1 \tag{30}$$

Thus, the QP problem is solved when for all i:

$$\alpha_i=0 \implies t_i\boldsymbol{\beta}\cdot h(\mathbf{x}_i)\ge 1$$
$$0<\alpha_i<C \implies t_i\boldsymbol{\beta}\cdot h(\mathbf{x}_i)= 1$$
$$\alpha_i=C \implies t_i\boldsymbol{\beta}\cdot h(\mathbf{x}_i)\le 1 \tag{31}$$

5. Discussions

In essence it has different physical meanings of introducing the training error in the optimization constraints of both SVM and ELM: the reason why SVM permits the training errors is that some training data may not be linearly separable in the conventional SVM feature space. According to the ELM theory [2–6] all the training data are linearly separable in the ELM feature space, the reason why ELM permits the training errors is to eliminate the possible overfitting and thus to minimize the testing errors and to further improve the generalization performance. According to ELM theories [2–6] widespread nonlinear piecewise continuous functions can be used as ELM feature mapping $h(\mathbf{x})$. Similar to the normalized ELM kernel [15] the ELM kernel function $K_{\mathrm{ELM}}(\mathbf{x}, \mathbf{x}_i)$ always satisfies the Mercer's condition.

5.1. Differences between SVM's separating hyperplane and ELM's separating hyperplane

Liu et al. [14] and Frénay and Verleysen [15] provide an efficient extreme learning approach to SVM by simply replacing the SVM kernels with ELM kernels (or normalized ELM kernels). Generally speaking, their work adopts the same optimization constrains as the conventional SVM.

The essence of SVM is: after the training data are mapped into the SVM feature space and there exists a hyperplane which can separate these data with a maximal margin. It is reasonable to think that such separating hyperplane in SVM may not necessarily pass through the origin in the SVM feature space and thus a bias b is required in the optimization constraint of SVM: $t_i(\mathbf{w}\cdot\phi(\mathbf{x}_i)+b)\ge 1-\xi_i$. The SVM with ELM kernels or normalized ELM kernels proposed by Liu et al. [14] and Frénay and Verleysen [15] has the same optimization constraint assumption as the conventional SVM. The corresponding Lagrange function of the primal SVM

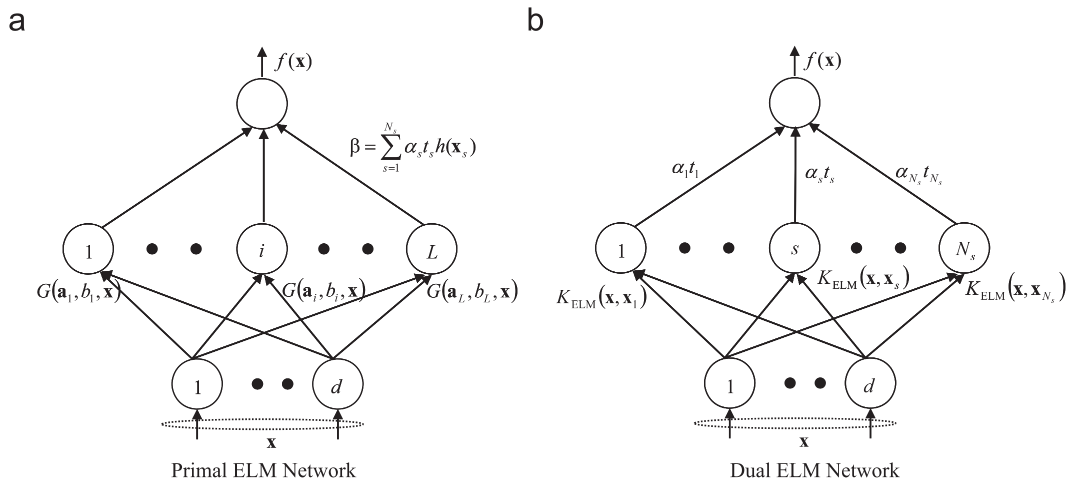
a
b
$f(\mathbf{x})$
$\beta = \sum_{s=1}^{N_s} \alpha_s t_s h(\mathbf{x}_s)$
$G(\mathbf{a}_1, b_1, \mathbf{x})$
$G(\mathbf{a}_i, b_i, \mathbf{x})$
$G(\mathbf{a}_L, b_L, \mathbf{x})$
x
Primal ELM Network
$\alpha_1 t_1$
$\alpha_s t_s$
$\alpha_{N_s} t_{N_s}$
$K_{\text{ELM}}(\mathbf{x}, \mathbf{x}_1)$
$K_{\text{ELM}}(\mathbf{x}, \mathbf{x}_s)$
$K_{\text{ELM}}(\mathbf{x}, \mathbf{x}_{N_s})$
Dual ELM Network

Fig. 1. (a) Primal ELM network: (1) L hidden nodes; (2) the output of the i-th hidden node is $G(\mathbf{a}_i, b_i, \mathbf{x}_i)$; (3) the output weight vector linking the hidden layer to the output node is $\boldsymbol{\beta} = \sum_{s=1}^{N_s} \alpha_s t_s K_{\text{ELM}}(\mathbf{x},\mathbf{x}_s)$. (b) Dual ELM Network: (1) N_s hidden nodes where N_s is the number of support vectors; (2) The output of the s-th hidden node is $K_{\text{ELM}}(\mathbf{x},\mathbf{x}_s)$: $K_{\text{ELM}}(\mathbf{x},\mathbf{x}_s) = [G(\mathbf{a}_1,b_1,\mathbf{x}),\ldots,G(\mathbf{a}_L,b_L,\mathbf{x})]^T \cdot [G(\mathbf{a}_1,b_1,\mathbf{x}_s),\ldots,G(\mathbf{a}_L,b_L,\mathbf{x}_s)]^T$; (3) the output weight linking the s-th hidden layer to the output node is $\alpha_s t_s$.

optimization (4) is

$$L_{\text{SVM}}(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\mu},b) = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(t_i(\mathbf{w}\cdot\phi(\mathbf{x}_i)+b) - (1-\xi_i)) - \sum_{i=1}^{N}\mu_i\xi_i \tag{32}$$

In order to find the optimal solutions of (32) one of the KKT conditions (necessary conditions) for the conventional SVM (as well as for SVM with ELM kernels [14,15]) is

$$\frac{\partial L_{\text{SVM}}(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\mu},b)}{\partial b} = 0 \implies \sum_{i=1}^{N}\alpha_i t_i = 0 \tag{33}$$

This is the reason why SVM learning needs to satisfy both optimization conditions (5): $\sum_{i=1}^{N}\alpha_i t_i = 0$ and $0 \le \alpha_i \le C, \forall i$.

According to ELM theories [2–6] the separating hyperplane of ELM basically passes through the origin in the SVM feature space, there is no bias b in the optimization constraint of ELM: $t_i\boldsymbol{\beta}\cdot h(\mathbf{x}_i) \ge 1-\xi_i$ and thus, different from SVM, ELM learning does not need to satisfy the condition: $\sum_{i=1}^{N}\alpha_i t_i = 0$.

SVM and ELM have similar optimization objective functions and SVM learning needs to satisfy both optimization conditions (5): $\sum_{i=1}^{N}\alpha_i t_i = 0$ and $0 \le \alpha_i \le C, \forall i$ while ELM learning only needs to satisfy the loose condition: $0 \le \alpha_i \le C, \forall i$, thus, obviously SVM tends to find a solution which is sub-optimal to ELM's solution.

This view can also become clear from a different observation aspect. ELM finds the optimal solution in the entire cube $[0, C]^N$ of the ELM feature space without any other constraints (cf. Fig. 2(a)). However, SVM always searches for the optimal solution in the hyperplane $\sum_{i=1}^{N}\alpha_i t_i = 0$ within the cube $[0, C]^N$ of the SVM feature space (cf. Fig. 2(b)). In this sense, SVM's search area depends more on the target output vector $[t_1,\ldots,t_N]^T$ instead of the combination of $(\mathbf{x}_i,t_i)$. In other words, given two training datasets $\{(\mathbf{x}_i^{(1)}, t_i^{(1)})\}_{i=1}^N$ and $\{(\mathbf{x}_i^{(2)}, t_i^{(2)})\}_{i=1}^N$ and $\{(\mathbf{x}_i^{(1)})\}_{i=1}^N$ and $\{(\mathbf{x}_i^{(2)})\}_{i=1}^N$ are totally irrelevant/independent, if $[t_1^{(1)},\ldots,t_N^{(1)}]^T$ is similar or close to $[t_1^{(2)},\ldots,t_N^{(2)}]^T$ (although this rarely happens in applications) SVM may have similar search areas of the cube $[0, C]^N$ for two different cases, which actually implies that the bias b should not be used in SVMs.[3]

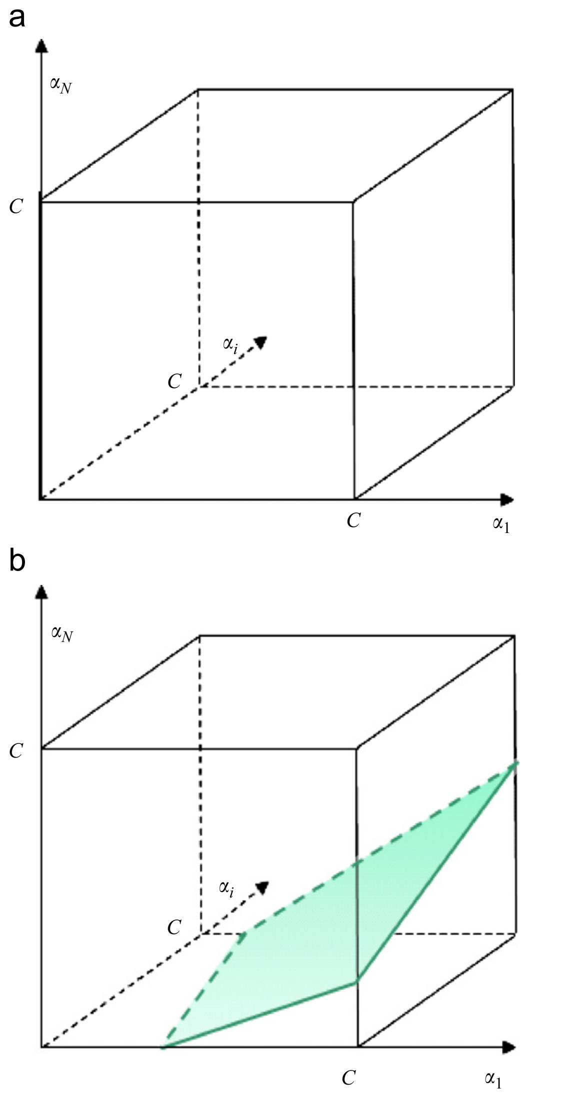

Fig. 2. (a) ELM finds the optimal solution in the entire cube $[0, C]^N$ of the ELM feature space without any other constraints. (b) SVM always searches for the optimal solution in the hyperplane $\sum_{i=1}^{N}\alpha_i t_i = 0$ (shaded area) within the cube $[0, C]^N$ of the SVM feature space.

[3] Poggio et al. [19] studies the role of the bias b in SVMs and shows that the bias b may not be required in some cases. However, different from our work, Poggio et al. [19] does not show that from the cross-application point of view the bias b should not be used in SVMs. A detail comparison of SVMs with and without the bias b is beyond the scope of this paper and will be reported in a separate work.

5.2. Maximal separating margin and minimal norm of output weights

The aim of ELM is to minimize the norm of the output weights $\boldsymbol{\beta}$: $\min\|\boldsymbol{\beta}\|$. However, the distance of the separating boundaries of the two classes in the ELM feature $2/\|\boldsymbol{\beta}\|$, thus to minimize the norm of the output weights $\boldsymbol{\beta}$ is actually to maximize the separating margin.

6. Performance verification

In this section, the performance of ELM for classification is compared with SVM on some benchmark binary classification problems: 11 UCI datasets [20] and 2 Gene expression datasets (leukemia and colon [21]) (cf Table 1). The training and testing data of the first 7 datasets of Table 1 are reshuffled at each trial of simulation while the training and testing data of the last 6 datasets remain fixed for all trials of simulations.

All the simulations for the SVM, SVM with ELM kernel and ELM for classification algorithms are carried out in MATLAB 2007 environment running in a Pentium 4, 2.53 GHZ CPU. The simulations for SVM are carried out using Matlab-coded SVM package: SVM and Kernel Methods Matlab Toolbox [22]. Although it is fully written in Matlab, it can perform fast due to the optimization of QP solver.

6.1. User specified parameters

The popular Gaussian kernel function $K(\mathbf{u},\mathbf{v})=\exp(-\gamma\|\mathbf{u}-\mathbf{v}\|^2)$ is used in SVM. Sigmoid type of ELM kernels are used in ELM and ELM for SVM: $K_{\text{ELM}}(\mathbf{x},\mathbf{x}_s)=[G(\mathbf{a}_1,b_1,\mathbf{x}),\ldots,G(\mathbf{a}_L,b_L,\mathbf{x})]^T\cdot[G(\mathbf{a}_1,b_1,\mathbf{x}_s),\ldots,G(\mathbf{a}_L,b_L,\mathbf{x}_s)]^T$ where $G(\mathbf{a},b,\mathbf{x})=1/(1+\exp(-(\mathbf{a}\cdot\mathbf{x}+b)))$. In order to achieve good generalization performance, the cost parameter C and kernel parameter γ of SVM need to be chosen appropriately. Similar to Ghanty et al. [23] we have tried a wide range of C and γ. For each dataset we have used 15 different values of C and 15 different values of γ resulting in a total of 225 pairs of (C,γ). The 15 different values of C are 0.001, 0.01, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 1000, 10 000 and the 15 different values of γ are 0.0001, 0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10, 20, 100, 1000 and 10 000.

It is known that the generalization performance of SVM usually depends closely on the combination of (C,γ) (see Fig. 3 for an example). The best generalization performance is usually achieved in a narrow range of such combinations. Thus, when SVM applied the best combination of (C,γ) needs to be chosen for each dataset.

Table 1
Specification of tested binary classification problems.

Datasets	# Attributes	# Training data	# Testing data
Breast-cancer	10	300	383
Liver-disorders	6	200	145
Heart	13	70	200
Ionosphere	34	100	251
Pimadata	8	400	368
Pwlinear	10	100	100
Sonar	60	100	158
Monks Problem 1	6	124	432
Monks Problem 2	6	169	432
Splice	60	1000	2175
A1a	123	1605	30 956
Leukemia	7129	38	34
Colon	2000	30	32

For ELM for classification and SVM with ELM kernel, there are two parameters: the cost parameter C and the number of hidden nodes L. For each problem we have also used 15 different values of C and 10 different values of L resulting in a total of 150 pairs of (C, L). The 15 different values of C are 0.001, 0.01, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 1000, and 10 000 and the 10 different values of L are 20, 30, 50, 80, 100, 200, 500, 1000, 2000, 3000. After extensive simulations, it is found that the performance of ELM and SVM with ELM kernel is actually not so sensitive to the combination of (C,L). Similar to SVM with the normalized ELM kernel [15], SVM with ELM kernel usually can achieve good generalization performance in most cases (although not in all cases) as long as the number of hidden node L is large enough. It is true to ELM as well. As observed from Fig. 4 (which is true to other datasets as well), the generalization performance of ELM tends to monotonically increase with the number of hidden nodes L. In all the simulations for 11 UCI datasets, L is set 1000. One only needs to adjust the cost parameter C although the performance of ELM is not very sensitive to C either. The insensitivity of ELM performance on the user specified parameters makes the implementation of ELM very effectively.

Twenty trials have been conducted for each problem, with training and testing data sets randomly generated for each trial. Simulation results including the average testing accuracy, the corresponding standard deviation (Dev), and the number of resulted support vectors are given in this section.

6.2. Performance comparison on UCI benchmark datasets

As observed from Table 2, SVM with ELM kernels and the conventional SVM achieve similar generalization performance, however, the conventional SVM face the trivial and time-consuming issue in finding the best combination of learning parameters while users can choose the learning parameters for SVM with ELM kernels easily. Among the three learning methods, ELM usually achieves the best generalization performance.

6.3. Performance comparison on DNA microarray datasets

The leukemia dataset was taken from a collection of leukemia patient samples [24]. The dataset consists of 72 samples: 25 samples of AML, and 47 samples of ALL. Each sample is measured over 7129 genes. The colon microarray dataset consists of 22 normal and 40 tumor tissue samples. In this dataset, each sample contains 2000 genes.

The minimum-redundancy-maximum-relevance (MRMR) feature selection method was developed by Ding and Peng et al. [25], and further studied in Ding and Peng [26]. MRMR considers both maximum relevance with the class labels and minimum redundancy among genes. That has six cases, such as MID, MIQ, FCD, FCQ, FDM, FSQ. The MRMR feature selection is among the most powerful methods to select a subset of features from a big pool, or just do dimension reduction. This method has been well cited and used for many different applications. Seen from Table 3, ELM generally achieves the best generalization performance in these two DNA microarray applications.

7. Conclusions

This paper studies ELM for classification with the standard optimization method. It is found that ELM and SVM are actually consistent in some manner: (1) to minimize the norm of output weights in ELM for classification is actually to maximize the distance of the separating margin of two different classes in the

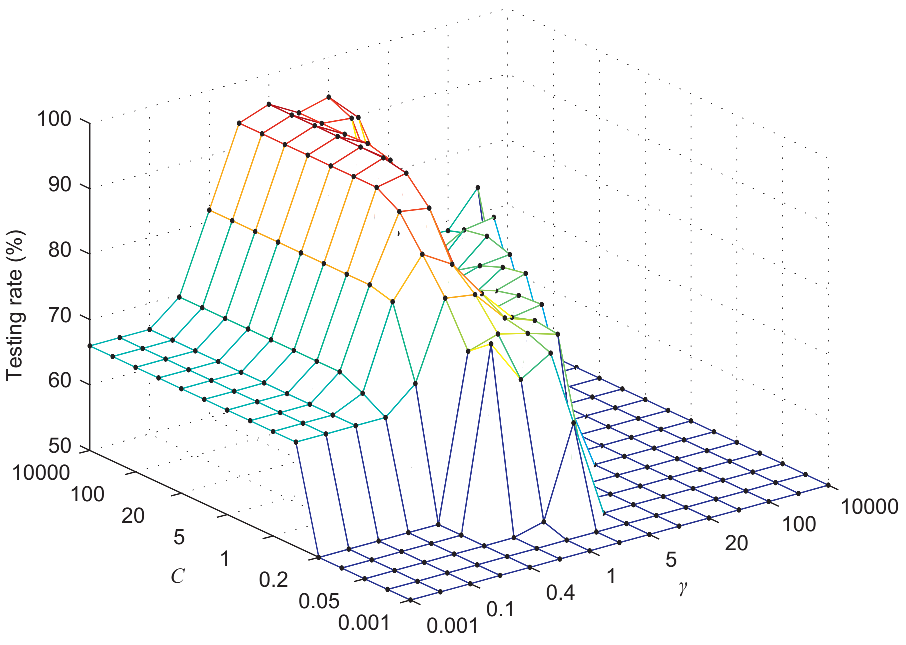

Fig. 3. The performance of SVM is sensitive to the parameters (C,γ): an example on Monks problem 1.

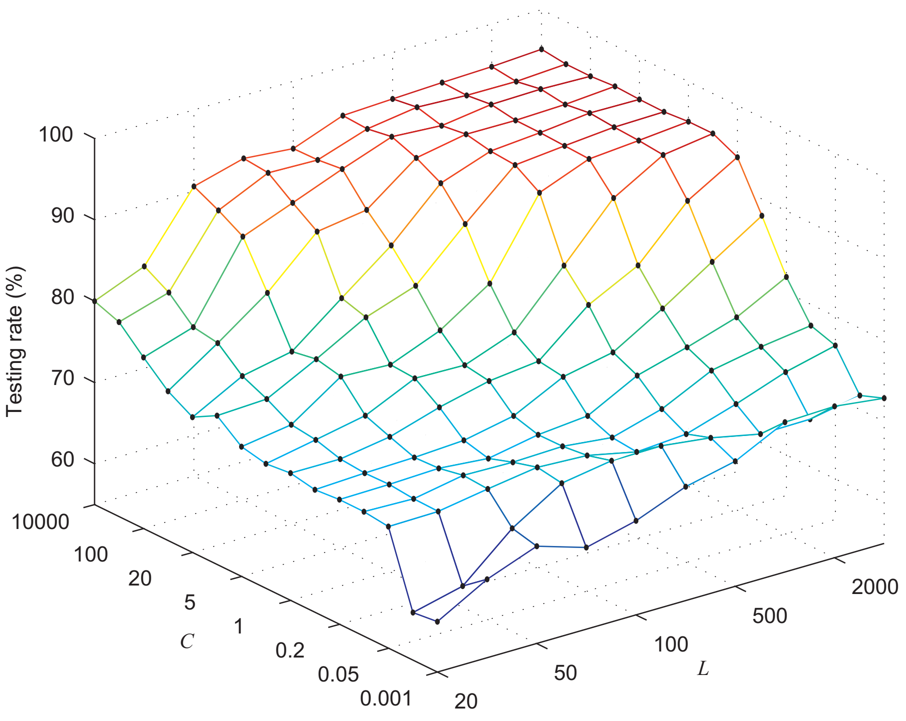

Fig. 4. The performance of ELM is not very sensitive to the parameters (C,L) and a good testing accuracy can be achieved as long as L is large enough: an example on Monks problem 1.

ELM feature space, which is equivalent to SVM's maximal separating margin property; (2) both ELM and SVM are to maximize the separating margin as well as to minimize the training errors.

This paper shows that the separating hyperplane tends to pass through the origin of the ELM feature space, resulting in less optimization constraints and better generalization performance than SVM. It is also pointed out that the generalization performance of ELM is less sensitive to the learning parameters especially the number of hidden nodes. Thus, compared to SVM, users can use ELM easily and effectively by avoiding tedious and time-consuming parameter tuning. From the implementation point of view, such ELM implementation may straightforward be extended to other SVM variants and those extensions are worth studying further in the future.

Similar to the conventional SVM, one may easily apply ELM in multi-label classification applications by using one-against-one (OAO) or one-against-all (OAA) methods.

Table 2
Comparison between the conventional SVM, SVM with ELM kernel and ELM for classification.

Datasets	SVM										ELM				
	Gaussian kernel					ELM kernel									
	(C,γ)	Training time (s)	Testing rate (%)	Testing Dev (%)	# SVs	C	Training time (s)	Testing rate (%)	Testing Dev (%)	# SVs	C	Training time (s)	Testing rate (%)	Testing Dev (%)	# SVs
Breast-cancer	(5, 50)	0.1118	94.20	0.87	190	10^{-3}	0.1442	96.28	0.59	65	10^{-3}	0.1423	**96.32**	0.75	66
Liver-disorders	(10, 2)	0.0972	68.24	4.58	158	10	0.1685	71.79	3.02	132	10	0.1734	**72.34**	2.55	131
Heart	(10^4, 5)	0.0382	76.00	3.85	41	50	0.0474	75.32	2.22	37	50	0.0344	**76.25**	2.70	36
Ionosphere	(5, 5)	0.0218	**90.58**	1.22	30	0.1	0.0396	88.78	1.77	25	10^{-3}	0.0359	89.48	1.12	32
Pimadata	(10^3, 50)	0.2049	76.43	1.57	209	0.01	0.2942	76.54	1.89	218	0.01	0.2867	**77.27**	1.33	217
Pwlinear	(10^4,10^3)	0.0357	84.35	3.28	54	10^{-3}	0.0560	84.10	2.88	49	10^{-3}	0.0486	**86.00**	1.92	51
Sonar	(20, 1)	0.0412	**83.33**	3.55	70	10^4	0.0498	78.81	3.75	44	10^4	0.0467	81.53	3.78	50
Monks Problem 1	(10, 1)	0.0424	**95.37**	0	70	10^4	0.0612	94.49	0.51	37	10^4	0.0821	95.19	0.41	40
Monks Problem 2	(10^4, 5)	0.0860	83.80	0	77	10^4	0.1077	**85.52**	0.59	75	10^4	0.0920	85.14	0.57	75
Splice	(2, 20)	2.0683	84.05	0	704	0.01	3.2076	85.34	0.41	507	0.01	3.5912	**85.50**	0.54	509
A1a	(2, 5)	5.6275	84.25	0	680	0.01	5.6017	84.13	10^{-3}	656	0.01	5.4542	**84.36**	0.79	666

Table 3
Comparison between the conventional SVM, SVM with ELM kernel and ELM in gene classification applications.

Datasets	SVM										ELM				
	Gaussian Kernel					ELM Kernel									
	(C,γ)	Training time (s)	Testing rate (%)	Testing Dev (%)	# SVs	(C,L)	Training time (s)	Testing rate (%)	Testing Dev (%)	# SVs	(C,L)	Training time (s)	Testing Rate (%)	Testing Dev (%)	# SVs
Before gene selection															
Leukemia	(10^3, 10^3)	3.2674	**82.35**	0	29	(0.05, 3000)	0.9796	80.15	2.44	35	(0.01, 3000)	0.2878	81.18	2.37	35
Colon	(10^3, 10^3)	0.2391	81.25	0	24	(0.01, 3000)	0.2757	81.87	2.54	28	(0.01, 3000)	0.1023	**82.50**	2.86	28
After gene selection (60 genes obtained for each case)															
Leukemia	(2, 20)	0.0640	**100**	0	12	(0.01, 3000)	0.0316	**100**	0	13	(10, 3000)	0.0199	**100**	0	11
Colon	(2, 50)	0.0166	87.50	0	26	(0,05, 20)	0.0160	82.34	4.22	25	(0,001, 500)	0.0161	**89.06**	2.10	28

References

[1] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagation errors, Nature 323 (1986) 533–536.

[2] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of International Joint Conference on Neural Networks (IJCNN2004), vol. 2, Budapest, Hungary, 25–29 July, 2004, pp. 985–990.

[3] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.

[4] G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Transactions on Neural Networks 17 (4) (2006) 879–892.

[5] G.-B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (2007) 3056–3062.

[6] G.-B. Huang, L. Chen, Enhanced random search based incremental extreme learning machine, Neurocomputing 71 (2008) 3460–3468.

[7] C. Cortes, V. Vapnik, Support vector networks, Machine Learning 20 (1995) 273–297.

[8] G.-B. Huang, Q.-Y. Zhu, K.Z. Mao, C.-K. Siew, P. Saratchandran, N. Sundararajan, Can threshold networks be trained directly? IEEE Transactions on Circuits and Systems II 53 (3) (2006) 187–191.

[9] N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate on-line sequential learning algorithm for feedforward networks, IEEE Transactions on Neural Networks 17 (6) (2006) 1411–1423.

[10] M.-B. Li, G.-B. Huang, P. Saratchandran, N. Sundararajan, Fully complex extreme learning machine, Neurocomputing 68 (2005) 306–314.

[11] G. Feng, G.-B. Huang, Q. Lin, R. Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning, IEEE Transactions on Neural Networks 20 (8) (2009) 1352–1357.

[12] H.-J. Rong, G.-B. Huang, N. Sundararajan, P. Saratchandran, Online sequential fuzzy extreme learning machine for function approximation and classification problems, IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 39 (4) (2009) 1067–1072.

[13] 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 Transactions on Information Theory 44 (2) (1998) 525–536.

[14] Q. Liu, Q. He, Z. Shi, Extreme support vector machine classifier, Lecture Notes in Computer Science 5012 (2008) 222–233.

[15] B. Frénay, M. Verleysen, Using SVMs with randomised feature spaces: an extreme learning approach, in: Proceedings of The 18th European Symposium on Artificial Neural Networks (ESANN), Bruges, Belgium, 28–30 April, 2010, pp. 315–320.

[16] G. Fung, O.L. Mangasarian, Proximal support vector machine classifiers, in: International Conference on Knowledge Discovery and Data Mining, San Francisco, California, USA, 2001, pp. 77–86.

[17] F. Rosenblatt, Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms, Spartan Books, New York, 1962.

[18] R. Fletcher, Practical Methods of Optimization: Volume 2 Constrained Optimization, Wiley, New York, 1981.

[19] T. Poggio, S. Mukherjee, R. Rifkin, A. Rakhlin, A. Verri, "*b*," A.I. Memo No. 2001-011, CBCL Memo 198, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 2001.

[20] C.L. Blake, C.J. Merz, UCI repository of machine learning databases, 〈http://www.ics.uci.edu/~mlearn/MLRepository.html〉, Department of Information and Computer Sciences, University of California, Irvine, USA, 1998.

[21] J. Li, H. Liu, Kent ridge bio-medical data set repository, 〈http://levis.tongji.edu.cn/gzli/data/mirror-kentridge.html〉, School of Computer Engineering, Nanyang Technological University, Singapore, 2004.

[22] S. Canu, Y. Grandvalet, V. Guigue, A. Rakotomamonjy, SVM and kernel methods matlab toolbox, 〈http://asi.insa-rouen.fr/enseignants/arakotom/

toolbox/index.html〉, Perception Systems et Information, INSA de Rouen, Rouen, France, 2005.
[23] P. Ghanty, S. Paul, N.R. Pal, NEUROSVM: an architecture to reduce the effect of the choice of kernel on the performance of svm, Journal of Machine Learning Research 10 (2009) 591–622.
[24] 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, E.S. Lander, Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, Science 286 (1999) 531–537.
[25] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (8) (2005) 1226–1238.
[26] C. Ding, H. Peng, Minimum redundancy feature selection from microarray gene expression data, in: Proceedings of the IEEE Computer Society Conference on Bioinformatics, 2003.

Guang-Bin Huang received the B.Sc. degree in applied mathematics and M.Eng. degree in Computer Engineering from Northeastern University, PR China, in 1991 and 1994, respectively, and Ph.D. degree in Electrical Engineering from Nanyang Technological University, Singapore in 1999. During undergraduate period, he also concurrently studied in Applied Mathematics department and Wireless Communication department of Northeastern University, PR China.

From June 1998 to May 2001, he worked as Research Fellow in Singapore Institute of Manufacturing Technology (formerly known as Gintic Institute of Manufacturing Technology) where he has led/implemented several key industrial projects. From May 2001, he has been working as an Assistant Professor and Associate Professor in the School of Electrical and Electronic Engineering, Nanyang Technological University. His current research interests include machine learning, computational intelligence, and extreme learning machine. He serves as an Associate Editor of *Neurocomputing* and *IEEE Transactions on Systems, Man and Cybernetics—Part B*. He is a senior member of IEEE.

Xiaojian Ding received the B.Sc. degree in applied mathematics and the M.Sc. degree in Computer Engineering from Xi'an University of Technology, China, in 2003 and 2006, respectively. He is currently working toward the Ph.D. degree at the school of Electronic and Information Engineering, Xi'an Jiao Tong University, China. He is studying in School of Electrical and Electronic Engineering, Nanyang Technological University under the award of Chinese Scholarship Council (CSC) from China. His research interests include neural networks, pattern recognition, and machine learning.

Hongming Zhou received the B.Eng. degree from Nanyang Technological University, Singapore in 2009. He is currently a Ph.D. student with School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests include neural networks and support vector machines.