Neurocomputing 261 (2017) 242–252

Contents lists available at ScienceDirect

Neurocomputing

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

Discriminative extreme learning machine with supervised sparsity preserving for image classification

Yong Peng[a,b,*], Bao-Liang Lu[c]

[a] MoE Key Laboratory of Complex Systems Modeling and Simulation, School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China
[b] Jiangsu Key Laboratory of Image and Video Understanding for Social Safety, Nanjing University of Science and Technology, Nanjing 210094, China
[c] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

ARTICLE INFO

Article history:
Received 17 September 2015
Revised 10 May 2016
Accepted 13 May 2016
Available online 7 February 2017

Keywords:
Extreme learning machine
Sparse representation
Group sparsity
Sparsity preserving
Image classification

ABSTRACT

In order to seek non-propagation method to train generalized single-hidden layer feed forward neural networks, extreme learning machine was proposed, which has been proven to be an effective and efficient model for both multi-class classification and regression. Different from most of existing studies which consider extreme learning machine as a classifier, we make improvements on it to let it become a feature extraction model in this paper. Specifically, a discriminative extreme learning machine with supervised sparsity preserving (SPELM) model is proposed. From the hidden layer to output layer, SPELM performs as a subspace learning method by considering the discriminative as well as sparsity information of data. The sparsity information of data is identified by solving a supervised sparse representation objective. Experiments are conducted on four widely used image benchmark data sets and the classification results demonstrate the effectiveness of the proposed SPELM model.

1. Introduction

Extreme learning machine (ELM) proposed by Huang et al. [1,2] is an efficient and effective method to train single hidden layer feed neural networks (SLFNs), providing us a unified framework for both multi-class classification and regression. The basic ELM model can be simply seen as a random feature mapping followed by a least square formula based linear regression. The main contribution of ELM to general SLFNs is that the input weights between the input layer and hidden layer as well as the biases of hidden units can be randomly generated, which leads to the analytical determination of the output weights between the hidden layer and output layer. Such improvement greatly alleviates the burden of weight tuning caused by the widely used back-propagation algorithms [3] and thus guarantees the fast learning speed of ELM. As an variant of SLFNs, though the mathematical formula of ELM is simple, the universal approximation capacity [4,5] can be also hold. Furthermore, the rationality of the randomly generated input weights is analyzed by some recent studies [6,7]. ELM research has been a hotspot and studies are conducted from diverse aspects such as theoretical investigation [6,7], model improvements [8,9] and various applications [10,11]. Some of the recent progresses were reviewed in [12].

In most of existing studies, ELM was treated as a classifier to various applications such as face recognition [13] and EEG signal classification [14]. In this paper, we make improvements on ELM from the feature extraction perspective. The underlying idea is to consider the output layer of ELM as a subspace of the ELM feature space, and we expect to learn a meaningful representation of data in such subspace. Therefore, our central task is to impose appropriate constraints on the output weights, which is also the general problem in designing an effective subspace learning algorithm. In machine learning and computer vision communities, sparse representation (SR) has been a promising method for statistical signal modeling. Identifying the sparsity information of data by solving an ℓ_1-norm regularized least square formula has become efficient [15] and thus many related applications based on SR are put forward such as face recognition [16], spectral clustering [17] and subspace learning [18,19]. The sparsity information of data usually presents us some excellent properties such as the self-expression capacity and discriminative information of data. Therefore, we naturally expect that the sparsity could be preserved by the ELM output weights. However, the difference lies in that what we are trying to preserve is the group sparsity [20] other than the generally used flat sparsity. By (1) modifying the indicator matrix in original ELM definition to make it proper for discriminant analysis; and (2)

* Corresponding author.
E-mail addresses: stany.peng@gmail.com, yongpeng@hdu.edu.cn (Y. Peng).

http://dx.doi.org/10.1016/j.neucom.2016.05.113
0925-2312/© 2017 Elsevier B.V. All rights reserved.

incorporating the sparsity preserving term into ELM objective as a regularizer, we formulate the objective function of the proposed SPELM model.

The remainder of this paper is organized as follows. In Section 2, we briefly review extreme learning machine, sparse representation and its supervised variant. The proposed sparsity preserving extreme learning machine including the model formulation of SPELM and some practical considerations is presented in Section 3. Experiments are conducted in Section 4 to evaluate the effectiveness of the proposed model. Section 5 concludes the whole paper.

Notations

Throughout the whole paper, we use bold upper case letter to denote matrices, bold lower case letter to denote vectors. For any matrix **M**, $\mathbf{m}_i$ means the ith column vector of **M**, $\mathbf{m}^i$ means the ith row vector of **M** and m_{ij} denotes the (i, j)th element of **M**. Tr(**M**) is the trace of **M** if it is a square matrix. $\mathbf{M}^T$ denotes the transpose of **M**. For $p \geq 1$, the ℓ_p-norm of a vector $\mathbf{m} \in \mathbb{R}^d$ is defined as $\|\mathbf{m}\|_p = (\sum_{i=1}^d |m_i|^p)^{1/p}$. The ℓ_0-norm of **m** counts its number of non-zero elements. The Frobenius norm of $\mathbf{M} \in \mathbb{R}^{d\times n}$ is $\|\mathbf{M}\|_F^2 = \sum_{i=1}^d \sum_{j=1}^n m_{ij}^2 = \mathrm{Tr}(\mathbf{M}^T\mathbf{M})$.

2. Preliminaries

2.1. Extreme learning machine

Considering the data collection matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$, the output function of ELM for generalized SLFNs is

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

where $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_L]^T$ is the output weight matrix between the hidden layer and output layer, L is the number of hidden units, and $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \ldots, h_L(\mathbf{x})]$ is the ELM nonlinear feature mapping, e.g., the output row vector of the hidden layer with respect to the input **x**. $h_i(\mathbf{x})$, $i = 1, 2, \ldots, L$ is the output of the ith hidden unit.

Basically, training SLFN based on ELM contains two stages: (1) calculating the hidden layer representation; (2) estimating the output weight. In the first stage, given a randomly generated input weight matrix $\mathbf{W} \in \mathbb{R}^{d\times L}$ and bias vector of hidden units $\mathbf{b} \in \mathbb{R}^{L\times 1}$, the hidden layer representation **H** can be obtained via

$$\mathbf{H} = \varphi(\mathbf{X}^T\mathbf{W} + \mathbf{b}), \tag{2}$$

where $\varphi(\cdot)$ is a nonlinear piecewise continuous function. The hidden layer is also called *ELM feature space*. In the second stage of ELM learning, the output weight between hidden layer and output layer, denoted by $\boldsymbol{\beta} \in \mathbb{R}^{L\times c}$, can be estimated by solving the following least square regression formula

$$\min_{\boldsymbol{\beta}} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}\|_2^2. \tag{3}$$

The hidden layer output matrix **H** is

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_n) \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}_1) & \ldots & h_L(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ h_1(\mathbf{x}_n) & \ldots & h_L(\mathbf{x}_n) \end{bmatrix} \tag{4}$$

To simplify the notation in following sections, we denote $\mathbf{h}^i$ as $\mathbf{h}(\mathbf{x}_i)$, which is the ith row vector in **H**. The training data response matrix **Y** is defined as

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_n \end{bmatrix} = \begin{bmatrix} y_{11} & \ldots & y_{1c} \\ \vdots & \vdots & \vdots \\ y_{n1} & \ldots & y_{nc} \end{bmatrix}. \tag{5}$$

Obviously, the solution to (3) is

$$\boldsymbol{\beta}^*_{\mathrm{ELM}} = \mathbf{H}^\dagger\mathbf{Y}, \tag{6}$$

where **H**† denotes the Moore–Penrose generalized inverse of matrix **H**.

To avoid the singularity problem when calculating the inverse and enhance the generalization performance of ELM, the regularized ELM [21] can be reached via shrinking the values of $\boldsymbol{\beta}$ as

$$\min_{\boldsymbol{\beta}} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2. \tag{7}$$

Similarly, the closed form solution to regularized ELM is

$$\boldsymbol{\beta}^*_{\mathrm{RELM}} = \left(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I}\right)^{-1}\mathbf{H}^T\mathbf{Y}, \tag{8}$$

where $\mathbf{I} \in \mathbb{R}^{L\times L}$ is an identity matrix.

2.2. Sparse representation

Given a data point $\mathbf{y} \in \mathbb{R}^{d\times 1}$ as well as an over-complete dictionary matrix $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m) \in \mathbb{R}^{d\times m}$, the goal of sparse representation is to represent **y** by using as few entries in **A** as possible. This can be expressed as follows

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_0, \quad s.t. \quad \mathbf{y} = \mathbf{A}\boldsymbol{\alpha}, \tag{9}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{m\times 1}$ is the representation coefficient. The above problem is NP-hard due to the non-convex and discrete nature of the ℓ_0-norm. However, it will become tractable by using the closest convex surrogate ℓ_1-norm instead. The corresponding objective is

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_1, \quad s.t. \quad \mathbf{y} = \mathbf{A}\boldsymbol{\alpha}. \tag{10}$$

Due to the noisy environment in real-world applications, the equality constraint is hard to satisfy. Thus, by considering the coding residual and taking the sparse constraint on coefficient as regularizer, the reformulated objective of sparse representation is

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\|\mathbf{y} - \mathbf{A}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1, \tag{11}$$

where λ is the regularization parameter. This is an ℓ_1-norm regularized least square problem which can be solved by many efficient algorithms [15]. Sometimes, the training samples are stacked together to form the dictionary **A**.

2.3. Supervised sparse representation

Recently, several studies have shown that the ℓ_1-norm induced sparse models perform well only in low-correlation settings [22,23]. Considering using the data collection matrix as dictionary, if samples from the same class are highly correlated, the ℓ_1-norm will encounter the stability problem and it generally tends to select a single representative sample randomly and ignore other correlated ones. This leads to a sparse solution but misses the correlation in data and often causes suboptimal performance. Specifically, for face recognition task in uncontrolled environment, the variation information (e.g., illumination and expression) may be more significant than the identity. In this case, it is possible that face images from different subjects with similar variations could have higher correlation than those from the same subject but with different variations. Therefore, considering the label information of training samples and emphasizing the sparsity of the number of classes instead of the number of samples are of great necessity, which leads the group sparsity. Below is the method to identify the group sparsity.

Assume that there are total n training images from c classes, each class has n_k images for $k = 1, \ldots, c$. Based on the standard sparse representation-based classification (SRC) [16], we are given a test image $\mathbf{y}$ and then aim to optimize the following supervised sparse representation objective by using training samples as columns of the dictionary matrix

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\|\mathbf{y}-\mathbf{A}\boldsymbol{\alpha}\|_2^2+\lambda\|\boldsymbol{\alpha}\|_g, \tag{12}$$

where λ is a regularization parameter, $\|\boldsymbol{\alpha}\|_g=\sum_{k=1}^{c}\|\boldsymbol{\alpha}_k\|_2$ and

$$\begin{aligned}&\mathbf{A}=(\mathbf{A}_1,\ldots,\mathbf{A}_c),\quad \mathbf{A}_k=(\mathbf{x}_{k1},\ldots,\mathbf{x}_{kn_k});\\&\boldsymbol{\alpha}=(\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_c)^T,\quad \boldsymbol{\alpha}_k=(\alpha_{k1},\ldots,\alpha_{kn_k})^T.\end{aligned} \tag{13}$$

Currently, we need to investigate the sparsity information and self-expression capacity of the training samples. Therefore, $\mathbf{y}$ will be replaced by each sample $\mathbf{x}_i, i=1,2,\ldots,n,$ and the corresponding objective is to reconstruct $\mathbf{x}_i$ by

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\|\mathbf{x}_i-\mathbf{A}_{\backslash i}\boldsymbol{\alpha}_{\backslash i}\|_2^2+\lambda\|\boldsymbol{\alpha}_{\backslash i}\|_g, \tag{14}$$

where $\mathbf{A}_{\backslash i}$ is obtained by removing $\mathbf{x}_i$ from $\mathbf{A}$, that is, $\mathbf{A}-\mathbf{x}_i$; and $\boldsymbol{\alpha}_{\backslash i}$ is the corresponding representation coefficient on $\mathbf{A}_{\backslash i}$.

The detailed implementation to (14) based on the $\ell_{2,1}$-norm optimization method [24] is given in Algorithm 1, where we simplify the notations of $\mathbf{A}_{\backslash i}$ and $\boldsymbol{\alpha}_{\backslash i}$ respectively as $\mathbf{A}$ and $\boldsymbol{\theta}$ to enhance the readability. If $\|\boldsymbol{\theta}_k^{(t+1)}\|_2, k=1,\ldots,c$ is close to zero, we usually use $\|\boldsymbol{\theta}_k^{(t+1)}\|_2+\epsilon, k=1,\ldots,c$ instead based on regularization technique, where ϵ is a small constant.

Algorithm 1. Optimization to (14).

```
Input: Training samples $\mathbf{X}=\{\mathbf{x}_i\}_{i=1}^n$ to construct the dictionary
    matrix $\mathbf{A}$, and regularization parameter $\lambda$;
Output: The representation coefficient $\boldsymbol{\theta}$.
1: Initialization. Set $t=0$ and $\mathbf{D}^{(t)}=\mathbf{I}\in\mathbb{R}^{n\times n}$;
2: while not converged do
3:     Calculate $\boldsymbol{\theta}$ as $\boldsymbol{\theta}^{(t+1)}=(\mathbf{A}^T\mathbf{A}+\lambda\mathbf{D}^{(t)})^{-1}\mathbf{A}^T\mathbf{x}_i$;
4:     Update the diagonal matrix $\mathbf{D}^{(t+1)}\in\mathbb{R}^{n\times n}$ as
           $$\mathbf{D}^{(t+1)}=\begin{bmatrix}\frac{1}{2\|\boldsymbol{\theta}_1^{(t+1)}\|_2}\mathbf{I}_{n_1} & \cdots & \mathbf{0}\\ \vdots & \ddots & \vdots\\ \mathbf{0} & \cdots & \frac{1}{2\|\boldsymbol{\theta}_c^{(t+1)}\|_2}\mathbf{I}_{n_c}\end{bmatrix}; \tag{15}$$
5:     Update $t=t+1$;
6: end while
```

Once $\boldsymbol{\theta}$ is obtained, we can construct the sparse coefficient vector $\mathbf{s}_i$ corresponding to $\mathbf{x}_i$ by adding a zero value in the ith position of $\boldsymbol{\theta}$, that is,

$$\mathbf{s}_i=(\theta_1,\ldots,\theta_{i-1},0,\theta_i,\ldots,\theta_{n-1})\in\mathbb{R}^{n\times 1}. \tag{16}$$

By this way, the learned coefficient $\mathbf{s}_i$ can explore the sparse structure as well as the label information of training samples.

3. Sparsity preserving extreme learning machine

In this section, we will give the formulation of sparsity preserving extreme learning machine (SPELM) in detail including the strategy of sparsity preserving, the objective of SPELM and some discussions between SPELM and related ELMs.

3.1. Sparsity preserving method

We expect that the group sparsity structure in the original data space to be preserved from the ELM feature space to the projected subspace, which can be achieved by minimizing the following objective

$$\mathcal{J}(\boldsymbol{\beta})=\min_{\boldsymbol{\beta}}\sum_{i=1}^{n}\|\boldsymbol{\beta}^T\mathbf{h}^{iT}-\boldsymbol{\beta}^T\mathbf{H}^T\mathbf{s}_i\|_2^2. \tag{17}$$

The above objective is the total sparse representation error of all training samples in the projected subspace; therefore, minimizing this term can ensure the sparse representation structure to be preserved in the projected subspace. Accordingly, it is employed to regularize the objective of extreme learning machine. With some algebraic transformation, we have

$$\begin{aligned}&\sum_{i=1}^{n}\|\boldsymbol{\beta}^T\mathbf{h}^{iT}-\boldsymbol{\beta}^T\mathbf{H}^T\mathbf{s}_i\|_2^2\\&=\mathrm{Tr}\left[\boldsymbol{\beta}^T\left(\sum_{i=1}^{n}\left(\mathbf{h}^{iT}-\mathbf{H}^T\mathbf{s}_i\right)\left(\mathbf{h}^{iT}-\mathbf{H}^T\mathbf{s}_i\right)^T\right)\boldsymbol{\beta}\right]\\&=\mathrm{Tr}\left[\boldsymbol{\beta}^T\left(\sum_{i=1}^{n}(\mathbf{H}^T\mathbf{e}_i-\mathbf{H}^T\mathbf{s}_i)(\mathbf{H}^T\mathbf{e}_i-\mathbf{H}^T\mathbf{s}_i)^T\right)\boldsymbol{\beta}\right]\\&=\mathrm{Tr}\left[\boldsymbol{\beta}^T\mathbf{H}^T\left(\sum_{i=1}^{n}\mathbf{e}_i\mathbf{e}_i^T-\mathbf{s}_i\mathbf{e}_i^T-\mathbf{e}_i\mathbf{s}_i^T+\mathbf{s}_i\mathbf{s}_i^T\right)\mathbf{H}\boldsymbol{\beta}\right]\\&=\mathrm{Tr}\left(\boldsymbol{\beta}^T\mathbf{H}^T(\mathbf{I}-\mathbf{S}-\mathbf{S}^T+\mathbf{S}^T\mathbf{S})\mathbf{H}\boldsymbol{\beta}\right),\end{aligned} \tag{18}$$

where $\mathbf{e}_i$ is an nth dimensional unit vector with the ith element 1 and the others 0.

3.2. Objective of SPELM

Considering the supervised sparsity preserving term $\mathcal{J}(\boldsymbol{\beta})$ shown in (17) as a regularizer, the formulated discriminative extreme learning machine with sparsity preserving (SPELM) has the following objective

$$\min_{\boldsymbol{\beta}}\|\mathbf{H}\boldsymbol{\beta}-\mathbf{Y}\|_2^2+\lambda_1\mathrm{Tr}\left(\boldsymbol{\beta}^T\mathbf{H}^T\mathbf{M}\mathbf{H}\boldsymbol{\beta}\right)+\lambda_2\|\boldsymbol{\beta}\|_2^2, \tag{19}$$

where $\mathbf{M}=\mathbf{I}-\mathbf{S}-\mathbf{S}^T+\mathbf{S}^T\mathbf{S}$, $\mathbf{Y}$ is a class indicator matrix defined as follows [25]

$$y_{ij}=\begin{cases}\sqrt{\frac{n}{n_j}}-\sqrt{\frac{n_j}{n}}, & \text{if } y_i=j\\ -\sqrt{\frac{n_j}{n}}, & \text{otherwise.}\end{cases} \tag{20}$$

$\mathbf{y}^i$ is the class label of $\mathbf{x}_i$, n_j is the sample size of the jth class, and n is the total sample size of data set. λ_1 controls the impact of sparsity preserving by $\boldsymbol{\beta}$ and λ_2 controls the effect of value shrinkage of $\boldsymbol{\beta}$.

Obviously, it is easy to verify that the optimal solution to (19) can be reached when $\boldsymbol{\beta}$ has the following estimation

$$\boldsymbol{\beta}^*=\left(\mathbf{H}^T(\mathbf{I}+\lambda_1\mathbf{M})\mathbf{H}+\lambda_2\mathbf{I}\right)^{-1}\mathbf{H}^T\mathbf{Y}. \tag{21}$$

This is an analytical solution to output weights. Except the process of sparsity identification, the computational complexity of SPELM is similar to that of GELM [26].

We summarize the whole framework of SPELM in Algorithm 2.

3.3. Efficient model selection

Parameters λ_1 and λ_2 play important roles in the implementation of SPELM, which are usually selected from some candidate values via cross-validation. In ELM setting, the dimension of hidden layer representation, which equals to the number of hidden units, are usually large. Therefore, the sizes of $\mathbf{H}^T\mathbf{M}\mathbf{H}$ and $\mathbf{H}^T\mathbf{H}$ are

Algorithm 2. Sparsity preserving extreme learning machine.

```
Input: Training samples $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$, and related parameters $\lambda$, $\lambda_1$, $\lambda_2$;
Output: The output weight matrix $\boldsymbol{\beta}$.
1: Randomly generate the input weight matrix $\mathbf{W}$ and bias vector $\mathbf{b}$;
2: Calculate the hidden layer output $\mathbf{H}$ by (2);
3: Centralize $\mathbf{H}$ by $\mathbf{H} \triangleq \mathbf{H}(\mathbf{I} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T)$;
4: Identify the sparsity information $\mathbf{S}$ via Algorithm 1;
5: Construct the indicator matrix $\mathbf{Y}$ based on (20);
6: Estimate the output weights $\boldsymbol{\beta}$ based on (21);
```

large and then the cross-validation process is computationally expensive. Following the pipeline in [27], we give an efficient method below to do model selection.

According to (21), we have

$$\begin{aligned}\boldsymbol{\beta}^* &= \left(\mathbf{H}^T\mathbf{H} + \lambda_1\mathbf{H}^T\mathbf{M}\mathbf{H} + \lambda_2\mathbf{I}\right)^{-1}\mathbf{H}^T\mathbf{Y} \\ &= \left[\lambda_2\left(\frac{1}{\lambda_2}\mathbf{H}^T\mathbf{H} + \frac{\lambda_1}{\lambda_2}\mathbf{H}^T\mathbf{M}\mathbf{H} + \mathbf{I}\right)\right]^{-1}\mathbf{H}^T\mathbf{Y} \\ &= \frac{1}{\lambda_2}\left(\frac{1}{\lambda_2}\mathbf{H}^T\mathbf{H} + \frac{\lambda_1}{\lambda_2}\mathbf{H}^T\mathbf{M}\mathbf{H} + \mathbf{I}\right)^{-1}\mathbf{H}^T\mathbf{Y}.\end{aligned} \tag{22}$$

Define $\gamma = \frac{\lambda_1}{\lambda_2}$ and $\mathbf{D}_{\lambda_2}$ as a diagonal matrix with each element on the diagonal as $(\mathbf{D}_{\lambda_2})_{ii} = \frac{1}{\lambda_2}$, $i = 1, 2, \ldots, n$, then we can rewrite the above equation as

$$\boldsymbol{\beta}^* = \frac{1}{\lambda_2}\left[\mathbf{H}^T(\mathbf{D}_{\lambda_2} + \gamma\mathbf{M})\mathbf{H} + \mathbf{I}\right]^{-1}\mathbf{H}^T\mathbf{Y}. \tag{23}$$

Since $(\mathbf{I} + \mathbf{AB})^{-1} = \mathbf{I} - \mathbf{A}(\mathbf{I} + \mathbf{BA})^{-1}\mathbf{B}$ [28], we have

$$\left[\mathbf{H}^T(\mathbf{D}_{\lambda_2} + \gamma\mathbf{M})\mathbf{H} + \mathbf{I}\right]^{-1} = \mathbf{I} - \mathbf{H}^T\tilde{\mathbf{M}}(\mathbf{I} + \mathbf{H}\mathbf{H}^T\tilde{\mathbf{M}})^{-1}\mathbf{H}, \tag{24}$$

where $\tilde{\mathbf{M}} = \mathbf{D}_{\lambda_2} + \gamma\mathbf{M}$. Therefore,

$$\boldsymbol{\beta}^* = \frac{1}{\lambda_2}\left[\mathbf{I} - \mathbf{H}^T\tilde{\mathbf{M}}(\mathbf{I} + \mathbf{H}\mathbf{H}^T\tilde{\mathbf{M}})^{-1}\mathbf{H}\right]\mathbf{H}^T\mathbf{Y}. \tag{25}$$

In the case when the number of ELM hidden units L is much larger than the sample size n, the size of matrix

$$\mathbf{I} + \mathbf{H}\mathbf{H}^T(\mathbf{D}_{\lambda_2} + \gamma\mathbf{M}) \in \mathbb{R}^{n\times n} \tag{26}$$

is much smaller than that of matrix

$$\mathbf{H}^T\mathbf{H} + \lambda_1\mathbf{H}^T\mathbf{M}\mathbf{H} + \lambda_2\mathbf{I} \in \mathbb{R}^{L\times L}, \tag{27}$$

the computational cost will be dramatically reduced. When the sample size is much larger than the number of ELM hidden units, the original formulation is more efficient.

3.4. Discussions

In this section, we give discussions on connections between SPELM and two closely related ELMs which are unsupervised ELM (US-ELM) [8] and graph regularized ELM (GELM) [26].

US-ELM is also a feature extraction model, which can be simply viewed as a random feature mapping followed by a locality preserving projection (LPP) [29]. Equivalently, we can think it as doing LPP in the ELM feature space. Basically, there are three different aspects between SPELM and US-ELM:

- US-ELM is an unsupervised subspace learning method and thus the output representation is more applicable to data clustering, while SPELM is a supervised learning method and thus the learned subspace representation is used to do supervised tasks such as classification;
- From the hidden layer to output layer, US-ELM expects the output weights to preserve the locality information of data. However, SPELM aims to preserve the global information based on the specially designed indicator matrix and the least square discriminant analysis [25];
- The dimension of the output representation in US-ELM is arbitrary, which leaves the dimension of the subspace a free parameter. For SPELM, we can consider it as an enhanced nonlinear discriminant analysis. Then, the maximal dimension of the learned subspace is $c-1$ (c is the number of classes).

Though the general objective of SPELM is similar to that of GELM, there are at least three different aspects between these two models:

- GELM is an enhanced ELM and of course a classifier, while SPELM is essentially a feature extraction method which can be theoretically suffixed by any classifier for pattern recognition;
- The structure information of training samples in GELM is obtained based on their label information. Therefore, the corresponding affinity matrix and graph Laplacian can be directly constructed, while in SPELM the sparsity information matrix $\mathbf{S}$ (can be also seen as one type of similarity measure) is obtained by solving a sparse learning objective;
- Specifically, GELM tries to minimize $\boldsymbol{\beta}^T\mathbf{H}^T\mathbf{L}\mathbf{H}\boldsymbol{\beta}$, while SPELM tries to minimize $\boldsymbol{\beta}^T\mathbf{H}^T\mathbf{M}\mathbf{H}\boldsymbol{\beta}$. Note that, a vector $\mathbf{f} = (f_1, f_2, \ldots, f_n)$ can be thought as a function defined on the graph such that f_i is the map of the ith node. Thus, a matrix can be thought as an operator acting on functions on the graph. In [30], Belkin and Niyogi show that under certain conditions,

$$\mathbf{M}\mathbf{f} \approx \frac{1}{2}\mathbf{L}^2\mathbf{f}. \tag{28}$$

Also, from spectral graph theory [31], we know that $\mathbf{L}$ provides a discrete approximation to the Laplace Beltrami operator $\mathcal{L}$ on the manifold. Therefore, the matrix $\mathbf{M}$ provides a discrete approximation to $\mathcal{L}^2$. This indicates that these two regularizers provide two different ways to approximate the eigenfunctions of Laplace Beltrami operator.

Table 1
The summary of data sets in our experiments. c is the number of classes, d is the number of feature dimensions, and n is the number of data points.

Dataset	c	d	n
COIL20	20	1024	1440
Yale-B	38	1024	2414
USPS	10	256	9298
UMIST	20	644	575

Table 2
Classification accuracy with deviation of different algorithms (%) on the COIL20 data set.

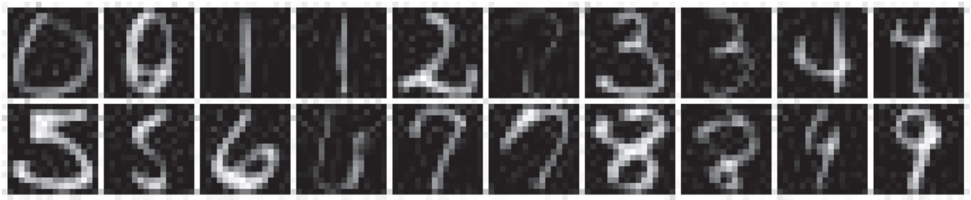
COIL20	2 train	4 train	6 train	8 train
LDA	68.05 ± 1.80	77.61 ± 1.69	81.03 ± 2.29	83.92 ± 1.57
LPP	68.94 ± 1.60	78.45 ± 1.87	81.58 ± 2.42	84.20 ± 1.73
NPE	68.44 ± 1.69	78.46 ± 2.00	81.44 ± 2.29	84.26 ± 1.47
SSC	68.54 ± 1.64	80.33 ± 1.75	84.08 ± 1.88	87.67 ± 1.71
USELM	61.68 ± 1.79	72.22 ± 1.71	77.26 ± 2.41	79.89 ± 1.70
RELM	70.54 ± 1.95	81.99 ± 1.94	86.35 ± 1.84	90.43 ± 1.59
GELM	71.02 ± 1.73	82.91 ± 1.86	87.13 ± 1.69	91.29 ± 1.55
SPELM	$\mathbf{74.61} \pm \mathbf{1.90}$	$\mathbf{85.23} \pm \mathbf{1.96}$	$\mathbf{89.00} \pm \mathbf{1.54}$	$\mathbf{92.07} \pm \mathbf{1.71}$

4. Experimental studies

In this section, we evaluate the proposed method, *i.e.*, SPELM, by comparing it with closely related subspace learning methods and classifiers.

(a) COIL20

(b) Extended Yale B


(c) USPS

(d) UMIST

Fig. 1. Some sample images from the four data sets used in our experiments.

Table 3
Classification accuracy with deviation of different algorithms (%) on the Extended Yale B data set.

Ext. Yale B	5 train	10 train	20 train	30 train
LDA	76.09 ± 1.66	86.85 ± 1.13	89.69 ± 0.70	86.96 ± 1.30
LPP	73.62 ± 1.61	84.89 ± 0.96	87.84 ± 0.84	85.55 ± 1.25
NPE	73.36 ± 1.99	84.57 ± 1.42	87.74 ± 0.96	81.79 ± 1.29
SSC	71.49 ± 1.68	86.55 ± 1.27	94.52 ± 0.76	97.03 ± 0.35
USELM	42.80 ± 2.80	55.71 ± 2.03	69.78 ± 1.69	78.84 ± 1.36
RELM	76.49 ± 1.72	88.13 ± 1.15	95.13 ± 0.76	97.72 ± 0.45
GELM	$\mathbf{78.18 \pm 1.61}$	89.08 ± 1.10	95.52 ± 0.71	98.04 ± 0.43
SPELM	77.67 ± 1.63	$\mathbf{89.45 \pm 1.15}$	$\mathbf{95.68 \pm 0.87}$	$\mathbf{98.13} \pm 0.26$

Table 4
Classification accuracy with deviation of different algorithms (%) on the USPS data set.

USPS	3 train	5 train	10 train	15 train
LDA	65.07 ± 4.20	65.19 ± 2.88	60.16 ± 3.07	55.40 ± 3.85
LPP	64.24 ± 4.11	65.46 ± 2.75	60.71 ± 2.81	57.32 ± 3.77
NPE	65.60 ± 4.64	66.34 ± 2.99	62.30 ± 2.53	56.90 ± 3.57
SSC	73.41 ± 3.25	80.18 ± 1.99	$\mathbf{87.44 \pm 0.98}$	$\mathbf{89.43 \pm 1.20}$
USELM	62.12 ± 3.48	64.78 ± 3.15	78.15 ± 1.56	80.30 ± 2.53
RELM	74.55 ± 2.89	80.29 ± 2.19	86.52 ± 1.10	88.66 ± 1.10
GELM	75.19 ± 2.93	80.87 ± 2.29	86.64 ± 1.12	88.73 ± 0.93
SPELM	$\mathbf{76.08 \pm 2.62}$	$\mathbf{81.17 \pm 2.47}$	86.76 ± 1.20	88.77 ± 1.20

Table 5
Classification accuracy with deviation of different algorithms (%) on the UMIST data set.

UMIST	3 train	5 train	7 train	9 train
LDA	74.84 ± 3.08	85.27 ± 2.59	89.37 ± 1.80	92.85 ± 1.94
LPP	75.57 ± 3.00	86.07 ± 2.58	90.08 ± 1.90	93.42 ± 1.84
NPE	75.81 ± 3.14	85.94 ± 2.57	89.83 ± 1.82	93.09 ± 1.91
SSC	75.83 ± 3.04	86.88 ± 2.84	91.63 ± 1.96	95.08 ± 1.47
USELM	61.08 ± 3.25	74.47 ± 2.42	81.75 ± 1.59	88.75 ± 1.08
RELM	75.93 ± 2.97	86.03 ± 2.65	90.56 ± 1.96	93.87 ± 1.47
GELM	80.11 ± 3.04	90.37 ± 2.68	94.08 ± 1.65	96.52 ± 1.23
SPELM	$\mathbf{81.47 \pm 3.03}$	$\mathbf{91.27 \pm 2.82}$	$\mathbf{95.64 \pm 1.66}$	$\mathbf{98.05 \pm 1.06}$

4.1. Data sets

Four widely used image data sets are employed in our experiments. The statistics of these data sets are summarized below (see also Table 1.)

- **COIL20 database**[1] is a data set of gray-scale images of 20 objects. The objects were placed on a motorized turntable against a background. The turntable was rotated through 360° to vary

[1] http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

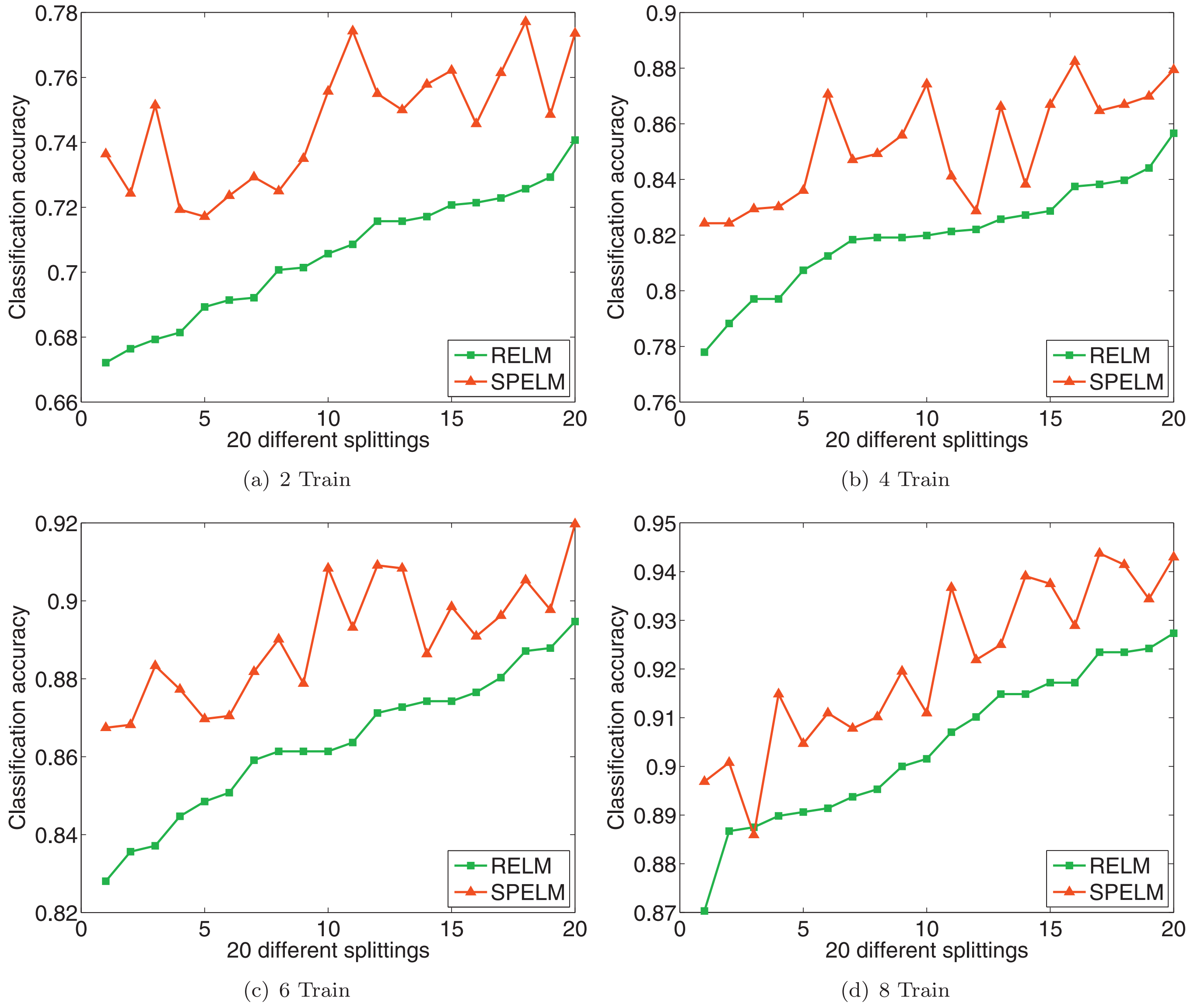


Fig. 2. Pairwise comparison between RELM and SPELM on the COIL20 data set. The *x*-axis denotes 20 different splittings of training and testing sets. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

the object poses with respect to a fixed camera. Images of the objects were taken at pose intervals of 5°, which corresponds to 72 images per object. In following experiments, we have resized each of the original 1440 images down to 32 × 32 pixels.

- **Extended Yale B database**[2] contains 16128 face images of 38 human subjects under 9 pose and 64 illumination conditions. In our experiment, we choose the frontal pose and use all the images under different illumination, thus we get 2414 images in total. All the face images are manually aligned and cropped. They are resize to 32 × 32 pixels, with 256 gray levels per pixel. Thus each face image is represented as a 1024-dimensional vector.
- **USPS database** consists of gray-scale handwritten digit images. We use a popular subset which contains 9298 handwritten digit images in total provided by Deng Cai[3]. The size of each image is 16 × 16 pixels with 256 gray levels.
- **UMIST database**[4] contains 20 subjects and totally 575 face images. Each covers a range of poses from profile to frontal views. Subjects cover a range of race/sex/appearance. All images are cropped and resized into 28 × 23 pixels per image.

Several sample images from these four data sets are shown in Fig. 1.

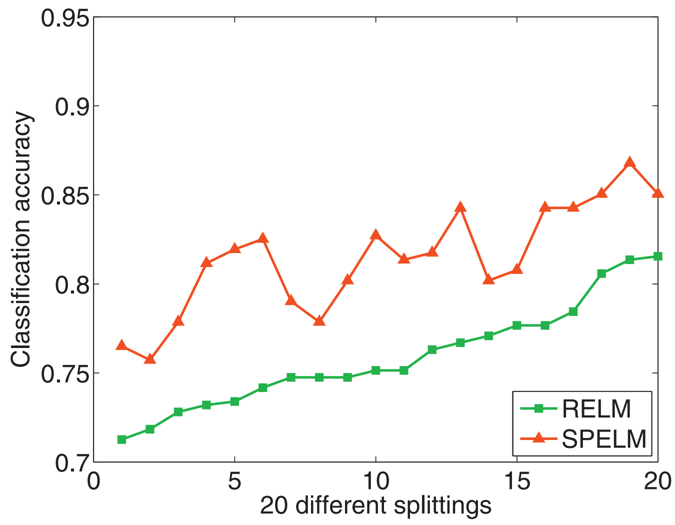
4.2. Experimental settings

In the following experiments, we compare SPELM with some closely related subspace learning methods such as linear discriminant analysis (LDA) [32], locality preserving projection (LPP) [29], and neighborhood preserving embedding (NPE) [33] and classifiers such as supervised sparse coding (SSC) [34] and ℓ_2-norm regularized extreme learning machine (RELM) [2]. Additionally, we also include the results of US-ELM [1] which can be seen as a regularized locality preserving projection from ELM feature space to

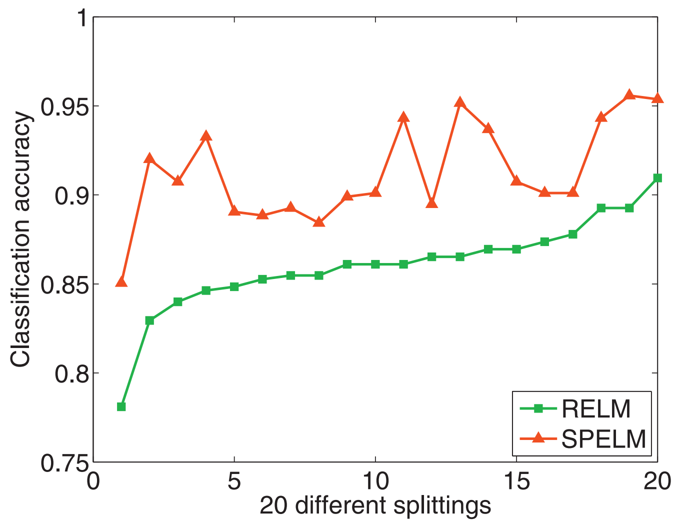
[2] http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html
[3] http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html
[4] http://images.ee.umist.ac.uk/danny/database.html

Table 6
Results of paired students' *t*-tests on the "> " relationship between the two accuracies on the four data sets.

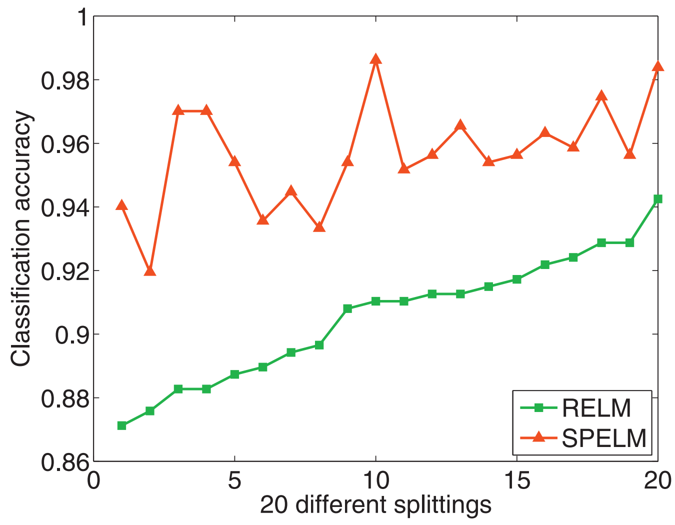
	COIL20-2	COIL2-4	COIL20-6	COIL20-8	YaleB-5	YaleB-10	YaleB-20	YaleB-30
SPELM> LDA	1	1	1	1	1	1	1	1
SPELM> LPP	1	1	1	1	1	1	1	1
SPELM> NPE	1	1	1	1	1	1	1	1
SPELM> SSC	1	1	1	1	1	1	1	1
SPELM> USELM	1	1	1	1	1	1	1	1
SPELM> RELM	1	1	1	1	1	1	1	1
SPELM> GELM	1	1	1	1	0	1	0	0
	USPS-3	USPS-5	USPS-10	USPS-15	UMIST-3	UMIST-5	UMIST-7	UMIST-9
SPELM> LDA	1	1	1	1	1	1	1	1
SPELM> LPP	1	1	1	1	1	1	1	1
SPELM> NPE	1	1	1	1	1	1	1	1
SPELM> SSC	1	1	0	0	1	1	1	1
SPELM> USELM	1	1	1	1	1	1	1	1
SPELM> RELM	1	1	0	0	1	1	1	1
SPELM> GELM	1	0	0	0	1	1	1	1



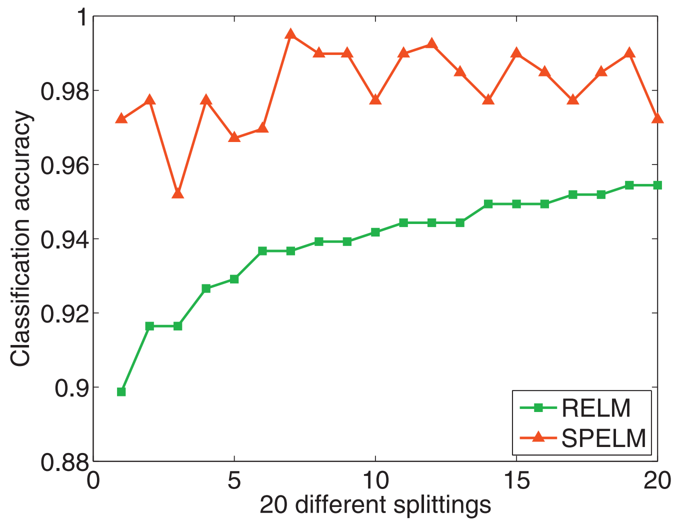
(a) 3 Train



(b) 5 Train

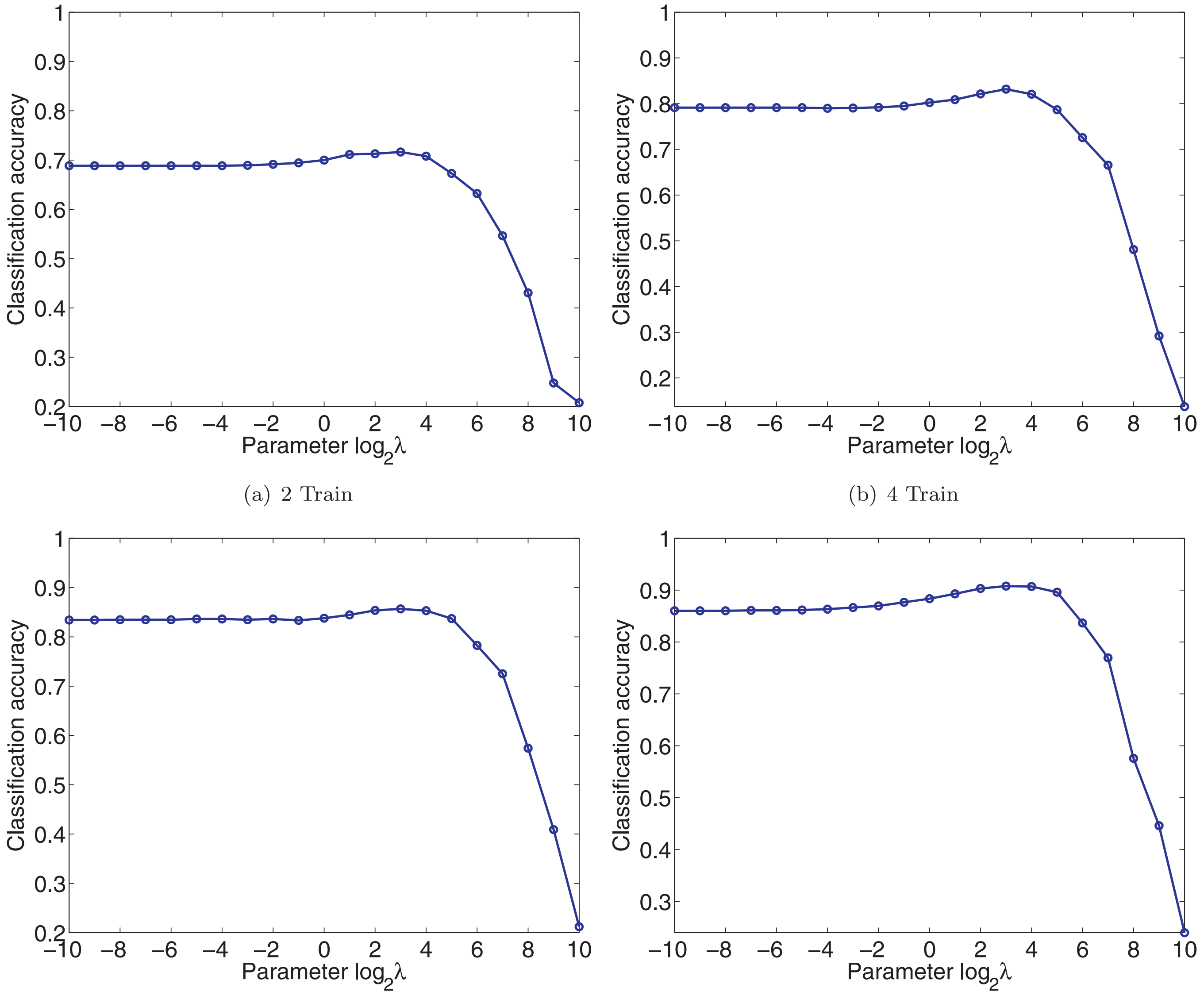


(c) 7 Train



(d) 9 Train

Fig. 3. Pairwise comparison between RELM and SPELM on the UMIST data set. The *x*-axis denotes 20 different splittings of training and testing sets. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)



(a) 2 Train (b) 4 Train (c) 6 Train (d) 8 Train

Fig. 4. Parameter sensitivity to λ on the COIL20 data set on the first splitting of training and testing sets.

learned subspace as well as the GELM [26]. For US-ELM, the regularization parameter is empirically set as 0.1. The two regularization parameters λ_1 and λ_2 are searched from $2^{\{-10,\dots,10\}}$. In our experiment, we consider the supervised learning task and obviously the supervised version of LPP and NPE is used. The SSC is based on the implementation provided by Feiping Nie[5]. For subspace learning methods, the K-nearest neighbor classifier with K=1 is used to do classification. For accelerating the computing, samples are projected onto PCA subspace with 0.98 energy preserved before doing supervised sparse coding.

For COIL20 data set, $p_{\text{COIL20}} = \{2, 4, 6, 8\}$ images from each class were randomly selected as training samples, and the remaining samples were used for testing. Similarly, for the remaining data sets, we respectively set $p_{\text{YaleB}} = \{5, 10, 20, 30\}$, $p_{\text{USPS}} = \{3, 5, 10, 15\}$, and $p_{\text{UMIST}} = \{3, 5, 7, 9\}$. Since the training samples were randomly chosen, we repeated the experiment 20 times and then calculated the average recognition accuracy. The random indices for selecting training samples are kept the same for all compared algorithms. In general, the classification rate varies with the dimensionality of the subspace and thus the best average performance obtained is reported. The number of nearest neighbors in NPE is set as n_{train}-1 if n_{train} is smaller than 5; otherwise, it is set as 5.

The ELM architecture is set as follows: the number of hidden units are consistently set as 2000 and the 'sigmoid' function is used as activation function. Before being fed into ELM network, all samples are projected into PCA subspace with ratio 1. For fair comparison, we record each randomly generated input weight matrix of the 20 experiments in RELM and use them to SPELM.

There are three parameters in SPELM: (1) λ is included in the sparsity identification stage; and (2) λ_1 and λ_2 are involved in the output weights estimation stage. It is usually time-consuming to do cross-validation on these three parameters by three-fold loops. In the following experiments, λ is determined by grid search from candidate values $2^{\{-10,\dots,10\}}$ based on SSC classification performance since λ is the only control parameter in SSC. Then, λ_1

[5] http://escience.cn/people/fpnie/papers.html

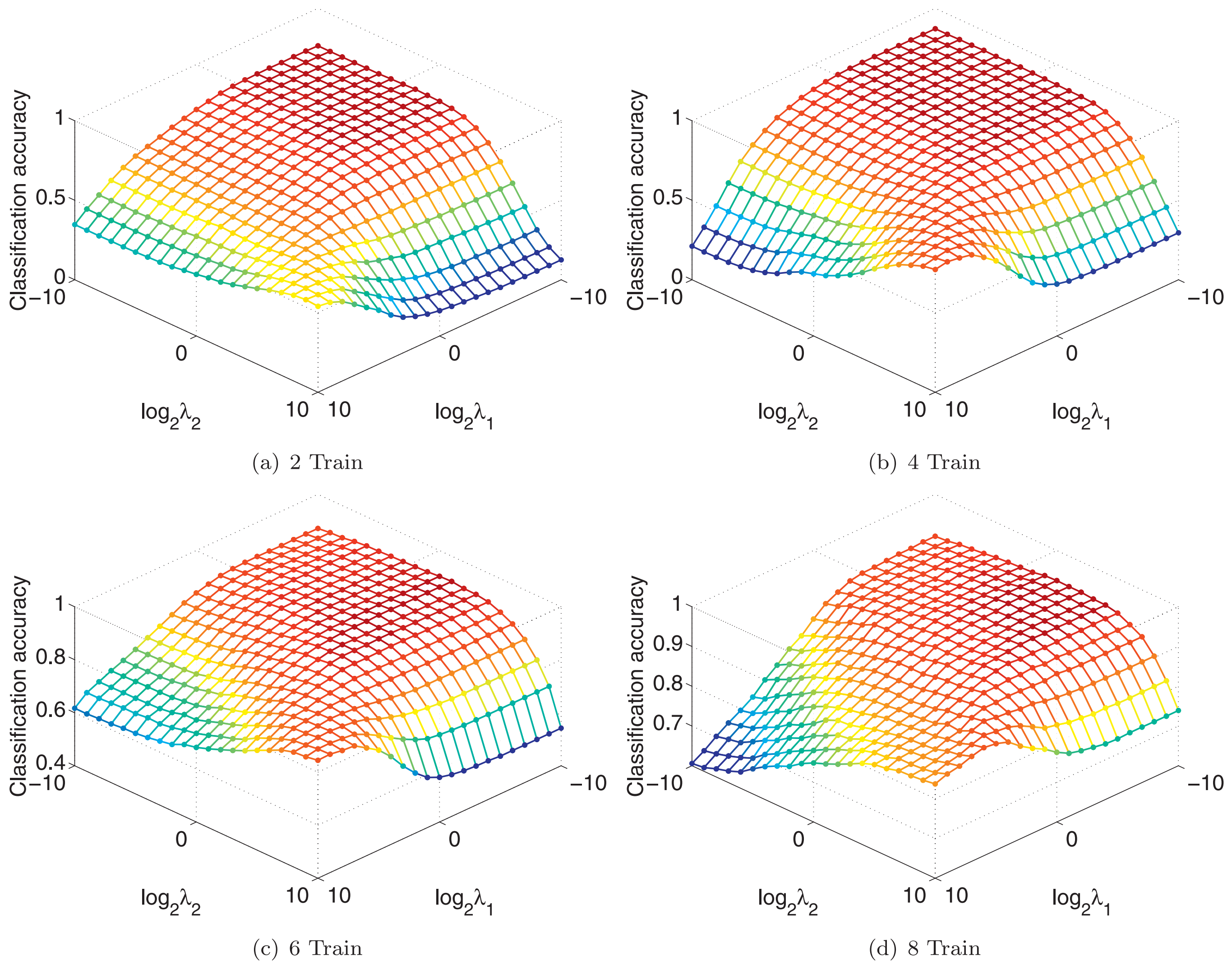


Fig. 5. Parameter sensitivity to λ_1 and λ_2 on the COIL20 data set on the first splitting of training and testing sets.

and λ_2 are also determined by grid search from candidate values $2^{\{-10,\ldots,10\}}$.

4.3. Experimental results

The experimental results are given in Tables 2, 3, 4 and 5 where the best results are shown in boldface. From these four tables, we can find some interesting points below.

- From pairwise comparison results, SPELM is consistently better than RELM on all four data sets. This indicates that the sparsity information can greatly enhance the performance of ELM and thus it is of great necessity to identify the sparsity structure of data.
- Three typical supervised subspace learning methods, LPP, LDA and NPE, have comparable performance across different data sets. Obviously, they are superior to USELM since it is an unsupervised subspace learning method. The only difference between USELM and unsupervised LPP is the random feature mapping of ELM from input layer to hidden layer. LPP in supervised version is similar to LDA, which has been analyzed in detail in [35]. Also, supervised NPE is to preserve the discriminative information in the neighborhood of each sample.
- SSC works well when given more training samples each class. Taking Extended Yale B data set for example, when given 5 training samples each class, SSC can only obtain 71.49% recognition rate, which is obviously worse then the other methods. However, it performs pretty well with accuracy 97.03% when $p_{\text{YaleB}} = 30$, which is comparable with RELM but slightly worse than SPELM. In most cases, SPELM performs better than GELM since SPELM simultaneously emphasizes the differences of iter-class and intra-class samples while GELM only considers the difference of samples on class level.
- Generally, all RELM, GELM and SPELM can achieve excellent performance on all the four data sets no matter how many training samples each class are given. This indirectly implies the rationality of the random feature mapping in ELM and it is a competitive classifier in pattern analysis.

Since Tables 2, 3, 4 and 5 only give the mean accuracies as well as standard deviations of compared algorithms. For more clearly demonstrating the improvement of SPELM with respect to RELM, we give the experimental results on COIL20 and UMIST data sets over the 20 different configurations of training and testing samples, which are respectively shown in Figs. 2 and 3. To make the figures look neater, we first sort the results of RELM over the 20 experiments and record the indices and then rearrange the results of SPELM based on the indices. It is easy to find that for each configuration of training and testing samples, the red curve is always over the green curve which means SPELM actually obtains accuracy improvement by incorporating the supervised sparsity information of data.

To illustrate the statistical difference between our approach and other algorithms, we did the paired students' *t*-test on these data sets. Here, the hypothesis is "the classification (mean) accuracy obtained by SPELM is greater than that obtained by the other (given) method". Each test us run on two accuracy sequences, which are obtained from the 20 splits by our method and the given method. Table 6 reports the results of the statistical tests. In each entity, "1" means that the hypothesis is correct (true) with probability 0.95, and "0" means that "the hypothesis is wrong (false)" with probability 0.95. For example, on the COIL20 data set (see Table 2), the decision "92.07 (SPELM) $>$ 90.43 (RELM)" is correct with probability 0.95. In summary, from Table 6, we see decision that "our algorithm achieves higher classification accuracy" is correct on most data sets.

4.4. Parameter sensitivity analysis

In this section, taking the COIL20 data set as an example, we show the performance sensitivity of SPELM in terms of parameters λ, λ_1 and λ_2. Similar results can be obtained on the other three data sets.

Since we directly use the cross-validation results of λ on SSC objective, we plot the performance of SSC with respect to λ in Fig. 4. As we can see, despite of being given different training samples, the performance of SSC is pretty stable when λ takes values from $2^{\{-10,\ldots,4\}}$. The performance of SPELM with respect to parameter combination (λ_1, λ_2) is shown in Fig. 5. For each number of training samples, there is a large flat area on the mesh where SPELM can get high accuracy, generally when λ_1 and λ_2 respectively takes values from $2^{\{-10,\ldots,0\}}$ and $2^{\{-10,\ldots,5\}}$.

5. Conclusion

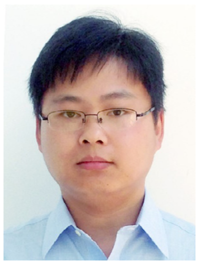
In this paper, we have proposed a novel ELM model, sparsity preserving extreme learning machine (SPELM), by viewing ELM as a feature extraction model instead of a classifier. Basically, SPELM is a two-stage model in which the first stage is to identify the supervised sparsity information by solving a sparse learning formula and the second stage is to estimate the output weights of ELM. Besides the model formulation, we have given some practical considerations on model selection and some discussions on the connection of SPELM and other related ELMs. Extensive experiments on image classification have shown the effectiveness of the proposed SPELM model.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments on this paper. This work was partially supported by Natural Science Foundation of China (61602140), Science and Technology Program of Zhejiang Province (2017C33049), Jiangsu Key Laboratory of Image and Video Understanding for Social Safety, Nanjing University of Science and Technology (30916014107) and Guangxi High School Key Laboratory of Complex Systems and Computational Intelligence.

References

[1] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of IEEE International Joint Conference on Neural Networks, vol. 2, 2004, pp. 985–990.
[2] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1) (2006) 489–501.
[3] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (1986) 533–536.
[4] G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Netw. 17 (4) (2006) 879–892.
[5] R. Zhang, Y. Lan, G.-B. Huang, Z.-B. Xu, Universal approximation of extreme learning machine with adaptive growth of hidden nodes, IEEE Trans. Neural Netw. Learn. Syst. 23 (2) (2012) 365–371.
[6] X. Liu, S. Lin, J. Fang, Z. Xu, Is extreme learning machine feasible? A theoretical assessment (Part I), IEEE Trans. Neural Netw. Learn. Syst. 26 (1) (2015) 7–20.
[7] S. Lin, X. Liu, J. Fang, Z. Xu, Is extreme learning machine feasible? A theoretical assessment (Part II), IEEE Trans. Neural Netw. Learn. Syst. 26 (1) (2015) 21–34.
[8] G. Huang, S. Song, J.N. Gupta, C. Wu, Semi-supervised and unsupervised extreme learning machines, IEEE Trans. Cybern. 44 (12) (2014) 2405–2417.
[9] J. Tang, C. Deng, G.-B. Huang, Extreme learning machine for multilayer perceptron, IEEE Trans. Neural Netw. Learn. Syst. 27 (4) (2016) 809–821.
[10] S. Suresh, R.V. Babu, H. Kim, No-reference image quality assessment using modified extreme learning machine classifier, Appl. Soft Comput. 9 (2) (2009) 541–552.
[11] A. Iosifidis, A. Tefas, I. Pitas, Minimum class variance extreme learning machine for human action recognition, IEEE Trans. Circ. Syst. Video Technol. 23 (11) (2013) 1968–1979.
[12] G. Huang, G.-B. Huang, S. Song, K. You, Trends in extreme learning machines: A review, Neural Netw.s 61 (2015) 32–48.
[13] W. Zong, G.-B. Huang, Face recognition based on extreme learning machine, Neurocomputing 74 (16) (2011) 2541–2551.
[14] N.-Y. Liang, P. Saratchandran, G.-B. Huang, N. Sundararajan, Classification of mental tasks from EEG signals using extreme learning machine, Int. J. Neural Syst. 16 (1) (2006) 29–38.
[15] A.Y. Yang, Z. Zhou, A.G. Balasubramanian, S.S. Sastry, Y. Ma, Fast ℓ_1-minimization algorithms for robust face recognition, IEEE Trans. Image Process. 22 (8) (2013) 3234–3246.
[16] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, Y. Ma, Robust face recognition via sparse representation, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2) (2009) 210–227.
[17] E. Elhamifar, R. Vidal, Sparse subspace clustering: algorithm, theory, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 35 (11) (2013) 2765–2781.
[18] L. Qiao, S. Chen, X. Tan, Sparsity preserving projections with applications to face recognition, Pattern Recog. 43 (1) (2010) 331–341.
[19] F. Yin, L. Jiao, F. Shang, L. Xiong, X. Wang, Sparse regularization discriminant analysis for face recognition, Neurocomputing 128 (2014) 341–362.
[20] M. Yuan, Y. Lin, Model selection and estimation in regression with grouped variables, J. R. Stat. Soc.: Ser. B (Stat. Methodol.) 68 (1) (2006) 49–67.
[21] G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 42 (2) (2012) 513–529.
[22] E. Grave, G.R. Obozinski, F.R. Bach, Trace lasso: a trace norm regularization for correlated designs, in: Proceedings of Advances in Neural Information Processing Systems, 2011, pp. 2187–2195.
[23] Y. Peng, B.-L. Lu, Robust group sparse representation via half-quadratic optimization for face recognition, in: Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering, 2015, pp. 146–151.
[24] F. Nie, H. Huang, X. Cai, C.H. Ding, Efficient and robust feature selection via joint ℓ2,1-norms minimization, in: Proceedings of Advances in Neural Information Processing Systems, 2010, pp. 1813–1821.
[25] J. Ye, Least squares linear discriminant analysis, in: Proceedings of the 24th International Conference on Machine learning, 2007, pp. 1087–1093.
[26] Y. Peng, S. Wang, X. Long, B.-L. Lu, Discriminative graph regularized extreme learning machine and its application to face recognition, Neurocomputing 149 (2015) 340–353.
[27] X. Shu, Y. Gao, H. Lu, Efficient linear discriminant analysis with locality preserving for face recognition, Pattern Recognit. 45 (5) (2012) 1892–1898.
[28] S.R. Searle, Matrix algebra useful for statistics, Wiley, New York, 1982.
[29] X. He, P. Niyogi, Locality preserving projections, in: Proceedings of Advances in Neural Information Processing Systems, vol.16, 2003, pp. 153–160.
[30] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, in: Proceedings of Advances in Neural Information Processing Systems, vol.14, 2001, pp. 585–591.
[31] F.R. Chung, Spectral Graph Theory, vol.92, American Mathematical Society, Providence, R.I., 1997.
[32] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, New York, 1990.
[33] X. He, D. Cai, S. Yan, H.-J. Zhang, Neighborhood preserving embedding, in: Proceedings of IEEE International Conference on Computer Vision, vol.2, 2005, pp. 1208–1213.
[34] J. Huang, F. Nie, H. Huang, C. Ding, Supervised and projected sparse coding for image classification, in: Proceedings of Twenty-Seventh AAAI Conference on Artificial Intelligence, 2013, pp. 438–444.
[35] D. Cai, X. He, J. Han, Using graph model for face analysis, Technical Report, Department of Computer Science, University of Illinois, Urbana-Champaign, Urbana, IL, 2005.



Yong Peng received the B.S. degree from Hefei New Star Research Institute of Applied Technology, the M.S. degree from Graduate University of Chinese Academy of Sciences, and the PhD degree from Shanghai Jiao Tong University, all in computer science, in 2006, 2010 and 2015, respectively. From September 2012 to August 2014, he was a visiting PhD student in the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor. He joined in School of Computer Science and Technology, Hangzhou Dianzi University as an Assistant Professor in June 2015 where he is currently a Research Associate Professor. He was awarded by the Presidential Scholarship, Chinese Academy of Sciences in 2009 and National Scholarship for Graduate Students, Ministry of Education in 2012. His research interests are machine learning, pattern recognition and evolutionary computation.

Bao-Liang Lu received his B.S. degree from Qingdao University of Science and Technology in 1982, the M.S. degree from Northwestern Polytechnical University in 1989 and the Ph.D. degree from Kyoto University in 1994. From 1982 to 1986, he was with the Qingdao University of Science and Technology. From April 1994 to March 1999, he was a Frontier Researcher at the Bio-Mimetic Control Research Center, the Institute of Physical and Chemical Research (RIKEN), Japan. From April 1999 to August 2002, he was a Research Scientist at the RIKEN Brain Science Institute. Since August 2002, he has been a full Professor at the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His research interests include brain-like computing, neural networks, machine learning, pattern recognition, and brain computer interface. He was the past President of the Asia Pacific Neural Network Assembly (APNNA) and the general Chair of ICONIP2011. He serves as Associate Editors of *IEEE Transactions on Cognitive and Developmental Systems* and *Neural Networks* (Elsevier). He is a governing board member of Asia Pacific Neural Network Society (APNNS) and a senior member of IEEE.