Semi-Supervised and Unsupervised Extreme Learning Machines

Gao Huang, Shiji Song, Jatinder N. D. Gupta, and Cheng Wu

***Abstract*—Extreme learning machines (ELMs) have proven to be efficient and effective learning mechanisms for pattern classification and regression. However, ELMs are primarily applied to supervised learning problems. Only a few existing research papers have used ELMs to explore unlabeled data. In this paper, we extend ELMs for both semi-supervised and unsupervised tasks based on the manifold regularization, thus greatly expanding the applicability of ELMs. The key advantages of the proposed algorithms are as follows: 1) both the semi-supervised ELM (SS-ELM) and the unsupervised ELM (US-ELM) exhibit learning capability and computational efficiency of ELMs; 2) both algorithms naturally handle multiclass classification or multicluster clustering; and 3) both algorithms are inductive and can handle unseen data at test time directly. Moreover, it is shown in this paper that all the supervised, semi-supervised, and unsupervised ELMs can actually be put into a unified framework. This provides new perspectives for understanding the mechanism of random feature mapping, which is the key concept in ELM theory. Empirical study on a wide range of data sets demonstrates that the proposed algorithms are competitive with the state-of-the-art semi-supervised or unsupervised learning algorithms in terms of accuracy and efficiency.**

***Index Terms*—Clustering, embedding, extreme learning machine (ELM), manifold regularization, semi-supervised learning, unsupervised learning.**

I. Introduction

SINGLE LAYER feedforward networks (SLFNs) have been intensively studied during the past several decades. Most of the existing learning algorithms for training SLFNs, such as the famous back-propagation algorithm [1] and the Levenberg–Marquardt algorithm [2], adopt gradient methods to optimize the weights in the network. Some existing papers also use forward selection or backward elimination approaches to construct network dynamically during the training process [3]–[7]. However, neither the gradient based methods nor the grow/prune methods guarantee a global optimal solution. Although various methods, such as the generic and evolutionary algorithms, have been proposed to handle the local minimum problem, they basically introduce high computational cost. One of the most successful algorithms for training SLFNs is the support vector machines (SVMs) [8], [9], which is a maximal margin classifier derived under the framework of structural risk minimization (SRM). The dual problem of SVMs is a quadratic programming problem which can be solved conveniently. Due to its simplicity and stable generalization performance, SVMs have been widely studied and applied to various domains [10]–[14].

Recently, Huang *et al*. [15], [16] proposed the extreme learning machines (ELMs) for training SLFNs. In contrast to most of the existing approaches, ELMs only update the output weights between the hidden layer and the output layer, while the parameters, i.e., the input weights and biases of the hidden layer, are randomly generated. By adopting the squared loss of the prediction error, the training of output weights turns into a regularized least squares (or ridge regression) problem which can be solved efficiently in closed form. It has been shown that even without updating the parameters of the hidden layer, the SLFN with randomly generated hidden neurons and tunable output weights maintains its universal approximation capability [17]–[19]. Compared to gradient based algorithms, ELMs are much more efficient and usually lead to better generalization performance [20]–[22]. Solving the regularized least squares problem in ELMs is also faster than solving the quadratic programming problem in standard SVMs. Recent papers show that the predicting accuracy achieved by ELMs is comparable with or even higher than that of SVMs [16], [22]–[24]. Moreover, ELMs provide an elegant and unified model for binary classification, multiclass classification and regression. The differences and similarities between ELMs and SVMs are discussed in [25] and [26], and new algorithms are proposed by combining the advantages of both models. In [25], an extreme SVM (ESVM) model is proposed by combining ELMs and the proximal SVM (PSVM). The ESVM algorithm is shown to be more accurate than the basic ELMs model due to the introduced regularization technique. The ESVM is also much more efficient than SVMs since there is no kernel matrix multiplication in ESVM. In [26], the traditional RBF kernel

Manuscript received May 12, 2013; revised November 20, 2013; accepted February 7, 2014. Date of publication March 12, 2014; date of current version November 13, 2014. This work was supported in part by the National Natural Science Foundation of China under Grant 61273233, in part by the Research Fund for the Doctoral Program of Higher Education under Grant 20120002110035 and Grant 20130002130010, in part by the National Key Technology Research and Development Program under Grant 2012BAF01B03, in part by the Project of China Ocean Association under Grant DY125-25-02, and in part by the Tsinghua University Initiative Scientific Research Program under Grant 2011THZ07132. This paper was recommended by Associate Editor J. Basak.

G. Huang, S. Song, and C. Wu are with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: huang-g09@mails.tsinghua.edu.cn; shijis@mail.tsinghua.edu.cn; wuc@tsinghua.edu.cn).

J. N. D. Gupta is with the College of Business Administration, University of Alabama in Huntsville, Huntsville, AL 35899 USA (e-mail: guptaj@uah.edu).

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

Digital Object Identifier 10.1109/TCYB.2014.2307349

is replaced by ELM kernel, leading to an efficient algorithm with matched accuracy of SVMs.

In the past years, researchers from various fields have made substantial contribution to ELM theories and applications. For example, the universal approximation ability of ELMs has been further studied in a classification context [23]. The generalization error bound of ELMs has been investigated from the perspective of the Vapnik–Chervonenkis (VC) dimension theory and the initial localized generalization error model (LGEM) [27], [28]. Various extensions have been made to the basic ELMs to make it more efficient and more suitable for specific problems, such as ELMs for online sequential data [29]–[31], ELMs for noisy/missing data [32]–[34], ELMs for imbalanced data [35], and so on. From the implementation aspect, ELMs have recently been implemented using parallel techniques [36], [37], and realized on hardware [38], which made ELMs feasible for large data sets and real time reasoning.

Though ELMs have become popular in a wide range of domains, they are primarily used for supervised learning tasks such as classification and regression that greatly limits their applicability. In some cases, such as text classification, information retrieval, and fault diagnosis, obtaining labels for fully supervised learning is time consuming and expensive, while a multitude of unlabeled data are easy and cheap to collect.

To overcome the disadvantage of supervised learning algorithms that they cannot make use of unlabeled data, semi-supervised learning (SSL) has been proposed to leverage both labeled and unlabeled data [39], [40]. The SSL algorithms assume that the input patterns from both labeled and unlabeled data are drawn from the same marginal distribution. Therefore, the unlabeled data naturally provide useful information for exploring the data structure in the input space. By assuming that the input data follows some cluster structure or manifold in the input space, SSL algorithms can incorporate both labeled and unlabeled data into the learning process. Since SSL requires less effort to collect labeled data and can offer higher accuracy, it has been applied to various domains [41]–[43]. In some other cases where no labeled data are available, people may be interested in exploring the underlying structure of the data. To this end, unsupervised learning (USL) techniques, such as clustering, dimension reduction, or data representation, are widely used to fulfill these tasks.

In this paper, we extend ELMs to handle both semi-supervised and unsupervised learning problems by introducing the manifold regularization framework. Both the proposed semi-supervised ELM (SS-ELM) and unsupervised ELM (US-ELM) inherit the computational efficiency and the learning capability of traditional ELMs. Compared with existing algorithms, SS-ELM and US-ELM are not only inductive (straightforward extension for out-of-sample examples at test time), but also can be used for multiclass classification or multicluster clustering directly. We test our algorithms on a variety of data sets, and make comparisons with other related algorithms. The results show that the proposed algorithms are competitive with the state-of-the-art algorithms in terms of both accuracy and efficiency.

It is worth mentioning that all the supervised, semi-supervised, and unsupervised ELMs can actually be put into a unified framework in which all the algorithms consist of two stages: 1) random feature mapping and 2) output weights solving. The first stage is to construct the hidden layer using randomly generated hidden neurons. This key concept in ELM theory differs from that of many existing feature learning methods. Generating feature mapping randomly enables ELMs for fast nonlinear feature learning and alleviates the problem of overfitting. The second stage is to solve the weights between the hidden layer and the output layer. This is where the main difference among supervised, semi-supervised, and unsupervised ELMs lies. We believe that the unified framework for the three types of ELMs might provide us a new perspective to understand the underlying behavior of random feature mapping in ELMs.

The rest of the paper is organized as follows. In Section II, we give a brief review of related existing literature on semi-supervised and unsupervised learning. Sections III and IV introduce the basic formulation of ELMs and the manifold regularization framework, respectively. We present the proposed SS-ELM and US-ELM algorithms in Sections V and VI. Experimental results are given in Section VII. Section VIII concludes the paper.

II. Related Works

Only a few existing research studies on ELMs have dealt with the problem of semi-supervised learning or unsupervised learning. In [44] and [45], the manifold regularization framework was introduced into the ELMs model to leverage both labeled and unlabeled data, thus extending ELMs for semi-supervised learning. However, these two approaches are limited to binary classification problems, thus they haven't explored the full power of ELMs. Moreover, both algorithms are only effective when the number of training patterns is larger than the number of hidden neurons. Unfortunately, this condition is usually violated in semi-supervised learning, since the training data is relatively scarce compared to the hidden neurons, whose number is commonly set to several hundreds or several thousands. Recently, a co-training approach has been proposed to train ELMs in a semi-supervised setting [46]. In this algorithm, the labeled training sets are augmented gradually by moving a small set of most confidently predicted unlabeled data to the labeled set at each loop, and ELMs are trained repeatedly on the pseudo-labeled set. Since the algorithm needs to train ELMs repeatedly, it introduces high computational cost.

The proposed SS-ELM is related to a few other manifold assumption based semi-supervised learning algorithms, such as the Laplacian support vector machines (LapSVMs) [47], the Laplacian regularized least squares (LapRLS) [47], semi-supervised neural networks (SSNNs) [48], and semi-supervised deep embedding [49]. It has been shown in these papers that manifold regularization is effective in semi-supervised learning, and usually leads to the state-of-the-art performances on a wide range of data sets.

The US-ELM proposed in this paper are related to the Laplacian eigenmaps (LE) [50] and spectral clustering (SC) [51] in that they both use spectral techniques for embedding

and clustering. In all these algorithms, an affinity matrix is first built from the input patterns. The SC performs eigen-decomposition on the normalized affinity matrix, and then embeds the original data into a d-dimensional space using the first d eigenvectors (each row is normalized to have unit length and represents a point in the embedded space) corresponding to the d largest eigenvalues. The LE algorithm performs generalized eigen-decomposition on the graph Laplacian, and uses the d eigenvectors corresponding to the second through the $(d+1)$th smallest eigenvalues for embedding. When LE and SC are used for clustering, then k-means is adopted to cluster the data in the embedded space. Similar to LE and SC, the US-ELM is also based on the affinity matrix, and is converted to solving a generalized eigen-decomposition problem. However, the eigenvectors obtained in US-ELM are not used for data representation directly, but are used as the parameters of the network, i.e., the output weights. Note that once the US-ELM model is trained, it can be applied to any presented data in the original input space. In this way, US-ELM provide a straightforward way for handling new patterns without recomputing eigenvectors as in LE and SC.

III. ELMs

Consider a supervised learning problem where we have a training set with N samples, $\{\boldsymbol{X}, \boldsymbol{Y}\} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$. Here $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$, $\boldsymbol{y}_i$ is a n_o-dimensional binary vector with only one entry (correspond to the class that $\boldsymbol{x}_i$ belongs to) equal to one for multiclassification tasks, or $\boldsymbol{y}_i \in \mathbb{R}^{n_o}$ for regression tasks, where n_i and n_o are the dimensions of input and output, respectively. ELMs aim to learn a decision rule or an approximation function based on the training data.

In general, the training of ELMs consists of two stages. The first stage is to construct the hidden layer using a fixed number of randomly generated mapping neurons, which can be any nonlinear piecewise continuous functions, such as the Sigmoid function (1) and Gaussian function (2)

$$g(\boldsymbol{x};\boldsymbol{\theta}) = \frac{1}{1+\exp(-(\boldsymbol{a}^T\boldsymbol{x}+b))} \tag{1}$$

$$g(\boldsymbol{x};\boldsymbol{\theta}) = \exp(-b\|\boldsymbol{x}-\boldsymbol{a}\|) \tag{2}$$

where $\theta = \{\boldsymbol{a}, b\}$ are the parameters of the mapping function and $\|\cdot\|$ denotes the Euclidean norm.

A notable feature of ELMs is that the parameters of the hidden mapping functions can be randomly generated according to any continuous probability distribution, e.g., the uniform distribution on $(-1, 1)$. This makes ELMs distinct from the traditional feedforward neural networks and SVMs. The only free parameters that need to be optimized in the training process are the output weights between the hidden neurons and the output nodes. By doing so, training ELMs is equivalent to solving a regularized least squares problem, which is considerately more efficient than training SVMs or learning with backpropagation.

In the first stage, a number of hidden neurons that map the data from the input space into a n_h-dimensional feature space (n_h is the number of hidden neurons) are randomly generated. We denote by $\boldsymbol{h}(\boldsymbol{x}_i) \in \mathbb{R}^{1\times n_h}$ the output vector of the hidden layer with respect to $\boldsymbol{x}_i$, and $\boldsymbol{\beta} \in \mathbb{R}^{n_h\times n_o}$ the output weights that connect the hidden layer with the output layer. Then, the outputs of the network are given by

$$\boldsymbol{f}(\boldsymbol{x}_i) = \boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta}, \quad i = 1, \ldots, N. \tag{3}$$

In the second stage, ELMs aim to solve the output weights by minimizing the sum of the squared losses of the prediction errors, which leads to the following formulation:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{n_h\times n_o}} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{C}{2}\sum_{i=1}^{N}\|\boldsymbol{e}_i\|^2$$
$$\text{s.t.} \quad \boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta} = \boldsymbol{y}_i^T - \boldsymbol{e}_i^T, \quad i = 1, \ldots, N \tag{4}$$

where the first term in the objective function is a regularization term against over-fitting, $\boldsymbol{e}_i \in \mathbb{R}^{n_o}$ is the error vector with respect to the ith training pattern, and C is a penalty coefficient on the training errors.

By substituting the constraints into the objective function, we obtain the following equivalent unconstrained optimization problem:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{n_h\times n_o}} L_{ELM} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{C}{2}\|\boldsymbol{Y}-\boldsymbol{H}\beta\|^2 \tag{5}$$

where $\boldsymbol{H} = [\boldsymbol{h}(\boldsymbol{x}_1)^T, \ldots, \boldsymbol{h}(\boldsymbol{x}_N)^T]^T \in \mathbb{R}^{N\times n_h}$.

The above problem is widely known as the ridge regression or regularized least squares. By setting the gradient of L_{ELM} with respect to $\boldsymbol{\beta}$ to zero, we have

$$\nabla L_{ELM} = \boldsymbol{\beta} + C\boldsymbol{H}^T(\boldsymbol{Y}-\boldsymbol{H}\boldsymbol{\beta}) = 0. \tag{6}$$

If $\boldsymbol{H}$ has more rows than columns and is of full column rank, which is usually the case where the number of training patterns is larger than the number of the hidden neurons, the above equation is overdetermined, and we have the following closed form solution for (5):

$$\boldsymbol{\beta}^* = \left(\boldsymbol{H}^T\boldsymbol{H} + \frac{\boldsymbol{I}_{n_h}}{C}\right)^{-1}\boldsymbol{H}^T\boldsymbol{Y} \tag{7}$$

where $\boldsymbol{I}_{n_h}$ is an identity matrix of dimension n_h.

Note that in practice, rather than explicitly inverting the $n_h \times n_h$ matrix in the above expression, we can obtain $\boldsymbol{\beta}^*$ by solving a linear system in a more efficient and numerically stable manner.

If the number of training patterns is smaller than the number of hidden neurons, then $\boldsymbol{H}$ will have more columns than rows, which often leads to an underdetermined least squares problem. In this case, $\boldsymbol{\beta}$ may have infinite number of solutions. To handle this problem, we restrict $\boldsymbol{\beta}$ to be a linear combination of the rows of $\boldsymbol{H}$: $\boldsymbol{\beta} = \boldsymbol{H}^T\boldsymbol{\alpha}$ ($\boldsymbol{\alpha} \in \mathbb{R}^{N\times n_o}$). Notice that when $\boldsymbol{H}$ has more columns than rows and is of full row rank, then $\boldsymbol{H}\boldsymbol{H}^T$ is invertible. Multiplying both side of (6) by $(\boldsymbol{H}\boldsymbol{H}^T)^{-1}\boldsymbol{H}$, we get

$$\boldsymbol{\alpha} + C\left(\boldsymbol{Y} - \boldsymbol{H}\boldsymbol{H}^T\boldsymbol{\alpha}\right) = 0. \tag{8}$$

This yields

$$\boldsymbol{\beta}^* = \boldsymbol{H}^T\boldsymbol{\alpha}^* = \boldsymbol{H}^T\left(\boldsymbol{H}\boldsymbol{H}^T + \frac{\boldsymbol{I}_N}{C}\right)^{-1}\boldsymbol{Y} \tag{9}$$

where $\boldsymbol{I}_N$ is an identity matrix of dimension N.

Therefore, in the case where training patterns are plentiful compared to the hidden neurons, we use (7) to compute the output weights, otherwise we use (9).

IV. Manifold Regularization Framework

Semi-supervised learning is built on the following two assumptions: (1) both the labeled data X_l and the unlabeled data X_u are drawn from the same marginal distribution $\mathcal{P}_X$ and (2) if two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are close to each other, then the conditional probabilities $P(\boldsymbol{y}|\boldsymbol{x}_1)$ and $P(\boldsymbol{y}|\boldsymbol{x}_2)$ should be similar as well. The latter assumption is widely known as the smoothness assumption in machine learning. To enforce this assumption on the data, the manifold regularization framework proposes to minimize the following cost function:

$$L_m = \frac{1}{2}\sum_{i,j} w_{ij}\|P(\boldsymbol{y}|\boldsymbol{x}_i) - P(\boldsymbol{y}|\boldsymbol{x}_j)\|^2 \tag{10}$$

where w_{ij} is the pair-wise similarity between two patterns $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

Note that the similarity matrix $\boldsymbol{W} = [w_{ij}]$ is usually sparse, since we only place a nonzero weight between two patterns $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ if they are close, e.g., $\boldsymbol{x}_i$ is among the k nearest neighbors of $\boldsymbol{x}_j$ or $\boldsymbol{x}_j$ is among the k nearest neighbors of $\boldsymbol{x}_i$. The nonzero weights are usually computed using Gaussian function $\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma^2)$, or simply fixed to 1.

Intuitively, the formulation (10) penalizes large variation in the conditional probability $P(\boldsymbol{y}|\boldsymbol{x})$ when $\boldsymbol{x}$ has a small change. This requires that $P(\boldsymbol{y}|\boldsymbol{x})$ vary smoothly along the geodesics in $\mathcal{P}(\boldsymbol{x})$.

Since it is difficult to compute the conditional probability, we can approximate (10) with the following expression:

$$\hat{L}_m = \frac{1}{2}\sum_{i,j} w_{ij}\|\hat{\boldsymbol{y}}_i - \hat{\boldsymbol{y}}_j\|^2 \tag{11}$$

where $\hat{\boldsymbol{y}}_i$ and $\hat{\boldsymbol{y}}_j$ are the predictions with respect to pattern $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, respectively.

It is straightforward to simplify the above expression in a matrix form

$$\hat{L}_m = \mathrm{Tr}\left(\hat{\boldsymbol{Y}}^T\boldsymbol{L}\hat{\boldsymbol{Y}}\right) \tag{12}$$

where $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix, $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$ is known as graph Laplacian, and $\boldsymbol{D}$ is a diagonal matrix with its diagonal elements $D_{ii} = \sum_{j=1}^{l+u}\boldsymbol{w}_{i,j}$. As discussed in [52], instead of using $\boldsymbol{L}$ directly, we can normalize it by $\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}}$ or replace it by $\boldsymbol{L}^p$ (p is an integer) based on some prior knowledge.

V. SS-ELM

In semi-supervised setting, we have few labeled data and plenty of unlabeled data. We denote the labeled data in the training set as $\{\boldsymbol{X}_l, \boldsymbol{Y}_l\} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{l}$, and unlabeled data as $\boldsymbol{X}_u = \{\boldsymbol{x}_i\}_{i=1}^{u}$, where l and u are the number of labeled and unlabeled data, respectively.

The proposed SS-ELM incorporates the manifold regularization to leverage unlabeled data to improve the classification accuracy when labeled data are scarce. By modifying the ordinary ELM formulation (4), we give the formulation of SS-ELM as

$$\begin{aligned}\min_{\boldsymbol{\beta}\in\mathbb{R}^{n_h\times n_o}} \ & \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{1}{2}\sum_{i=1}^{l} C_i\|\boldsymbol{e}_i\|^2 + \frac{\lambda}{2}\mathrm{Tr}\left(\boldsymbol{F}^T\boldsymbol{L}\boldsymbol{F}\right) \\ \text{s.t.} \quad & \boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta} = \boldsymbol{y}_i^T - \boldsymbol{e}_i^T, \quad i = 1, \ldots, l \\ & \boldsymbol{f}_i = \boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta}, \quad i = 1, ..., l+u\end{aligned} \tag{13}$$

where $\boldsymbol{L} \in \mathbb{R}^{(l+u)\times(l+u)}$ is the graph Laplacian built from both labeled and unlabeled data, and $\boldsymbol{F} \in \mathbb{R}^{(l+u)\times n_o}$ is the output matrix of the network with its ith row equal to $\boldsymbol{f}(\boldsymbol{x}_i)$, λ is a tradeoff parameter.

Note that similar to the weighted ELM algorithm (W-ELM) introduced in [35], here, we associate different penalty coefficient C_i on the prediction errors with respect to patterns from different classes. This is because we found that when the data is skewed, i.e., some classes have significantly more training patterns than other classes, traditional ELMs tend to fit the classes having the majority of patterns quite well, but fits other classes poorly. This usually leads to poor generalization performance on the testing set. Therefore, we propose to alleviate this problem by reweighting instances from different classes. Suppose that $\boldsymbol{x}_i$ belongs to class t_i, which has N_{t_i} training patterns, then we associate $\boldsymbol{e}_i$ with a penalty of $C_i = C_0/N_{t_i}$, where C_0 is a user-defined parameter as in traditional ELMs. In this way, the patterns from the dominant classes will not be over fitted by the algorithm, and the patterns from a class with less samples will not be neglected.

We substitute the constraints into the objective function, and rewrite the above formulation in a matrix form

$$\begin{aligned}\min_{\boldsymbol{\beta}\in\mathbb{R}^{n_h\times n_o}} \ & \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{1}{2}\|\boldsymbol{C}^{\frac{1}{2}}\left(\widetilde{\boldsymbol{Y}} - \boldsymbol{H}\boldsymbol{\beta}\right)\|^2 \\ & + \frac{\lambda}{2}\mathrm{Tr}\left(\boldsymbol{\beta}^T\boldsymbol{H}^T\boldsymbol{L}\boldsymbol{H}\boldsymbol{\beta}\right)\end{aligned} \tag{14}$$

where $\widetilde{\boldsymbol{Y}} \in \mathbb{R}^{(l+u)\times n_o}$ is the augmented training target with its first l rows equal to $\boldsymbol{Y}_l$ and the rest equal to 0, $\boldsymbol{C}$ is a $(l+u)\times(l+u)$ diagonal matrix with its first l diagonal elements $[\boldsymbol{C}]_{ii} = C_i$ and the rest equal to 0.

Again, we compute the gradient of the objective function with respect to $\boldsymbol{\beta}$

$$\nabla L_{SS-ELM} = \boldsymbol{\beta} + \boldsymbol{H}^T\boldsymbol{C}(\widetilde{\boldsymbol{Y}} - \boldsymbol{H}\boldsymbol{\beta}) + \lambda.\boldsymbol{H}^T\boldsymbol{L}\boldsymbol{H}\boldsymbol{\beta}. \tag{15}$$

By setting the gradient to zero, we obtain the solution to the SS-ELM

$$\boldsymbol{\beta}^* = \left(\boldsymbol{I}_{n_h} + \boldsymbol{H}^T\boldsymbol{C}\boldsymbol{H} + \lambda\boldsymbol{H}^T\boldsymbol{L}\boldsymbol{H}\right)^{-1}\boldsymbol{H}^T\boldsymbol{C}\widetilde{\boldsymbol{Y}}. \tag{16}$$

As in Section III, if the number of labeled data is fewer than the number of hidden neurons, which is common in SSL, we have the following alternative solution:

$$\boldsymbol{\beta}^* = \boldsymbol{H}^T\left(\boldsymbol{I}_{l+u} + \boldsymbol{C}\boldsymbol{H}\boldsymbol{H}^T + \lambda\boldsymbol{L}\boldsymbol{H}\boldsymbol{H}^T\right)^{-1}\boldsymbol{C}\widetilde{\boldsymbol{Y}} \tag{17}$$

where $\boldsymbol{I}_{l+u}$ is an identity matrix of dimension $l+u$.

Algorithm 1 SS-ELM Algorithm

Input:

Labeled patterns, $\{\boldsymbol{X}_l, \boldsymbol{Y}_l\} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{l}$;

Unlabeled patterns, $\boldsymbol{X}_u = \{\boldsymbol{x}_i\}_{i=1}^{u}$;

Output:

The mapping function of SS-ELM: $\boldsymbol{f} : \mathbb{R}^{n_i} \to \mathbb{R}^{n_o}$

Step 1: Construct the graph Laplacian $\boldsymbol{L}$ from both $\boldsymbol{X}_l$ and $\boldsymbol{X}_u$.

Step 2: Initiate an ELM network of n_h hidden neurons with random input weights and biases, and calculate the output matrix of the hidden neurons $\boldsymbol{H} \in \mathbb{R}^{(l+u)\times n_h}$.

Step 3:

Choose the tradeoff parameter C_0 and λ.

Step 4:

1) **If** $n_h \leq N$
   Compute the output weights $\boldsymbol{\beta}$ using (16)
2) **Else**
   Compute the output weights $\boldsymbol{\beta}$ using (17)

return The mapping function $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x})\beta$.

Note that by setting λ to be zero and the diagonal elements of C_i $(i = 1, \ldots, l)$ to be the same constant, (16) and (17) reduce to the solutions of traditional ELMs (7) and (9), respectively.

Based on the above discussion, the SS-ELM algorithm is summarized as Algorithm 1.

VI. Unsupervised ELM

In unsupervised setting, the entire training data $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$ are unlabeled (N is the number of training patterns) and our target is to find the underlying structure of the original data. The formulation of US-ELM follows from the formulation of SS-ELM. When there is no labeled data, (14) is reduced to

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{n_h\times n_o}} \|\boldsymbol{\beta}\|^2 + \lambda \mathrm{Tr}\left(\boldsymbol{\beta}^T\boldsymbol{H}^T\boldsymbol{L}\boldsymbol{H}\boldsymbol{\beta}\right). \tag{18}$$

Note that the above formulation always attains its minimum at $\boldsymbol{\beta} = \boldsymbol{0}$. As suggested in [50], we have to introduce addtional constraints to avoid a degenerated solution. Specifically, the formulation of US-ELM is given by

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{n_h\times n_o}} \|\boldsymbol{\beta}\|^2 + \lambda \mathrm{Tr}\left(\boldsymbol{\beta}^T\boldsymbol{H}^T\boldsymbol{L}\boldsymbol{H}\beta\right) \quad \text{s.t. } (\boldsymbol{H}\boldsymbol{\beta})^T\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{I}_{n_o}. \tag{19}$$

Theorem 1: An optimal solution to (19) is given by choosing $\boldsymbol{\beta}$ as the matrix whose columns are the eigenvectors (normalized to satisfy the constraint) corresponding to the first n_o smallest eigenvalues of the generalized eigenvalue problem

$$\left(\boldsymbol{I}_{n_h} + \lambda\boldsymbol{H}^T\boldsymbol{L}\boldsymbol{H}\right)\boldsymbol{v} = \gamma\boldsymbol{H}^T\boldsymbol{H}\boldsymbol{v}. \tag{20}$$

Proof: We can rewrite (19) as

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{n_h\times n_o}, \boldsymbol{\beta}^T\boldsymbol{B}\boldsymbol{\beta}=\boldsymbol{I}_{n_o}} \mathrm{Tr}\left(\boldsymbol{\beta}^T\boldsymbol{A}\boldsymbol{\beta}\right) \tag{21}$$

where $\boldsymbol{A} = \boldsymbol{I}_{n_h} + \lambda\boldsymbol{H}^T\boldsymbol{L}\boldsymbol{H}$ and $\boldsymbol{B} = \boldsymbol{H}^T\boldsymbol{H}$.

Algorithm 2 US-ELM Algorithm

Input:

The training data: $\boldsymbol{X} \in \mathbb{R}^{N\times n_i}$;

Output:

1) For embedding task:
   The embedding in a n_o-dimensional space: $\boldsymbol{E} \in \mathbb{R}^{N\times n_o}$;
2) For clustering task:
   The label vector of cluster index: $\boldsymbol{y} \in \mathbb{N}_+^{N\times 1}$.

Step 1: Construct the graph Laplacian $\boldsymbol{L}$ from $\boldsymbol{X}$.

Step 2: Initiate an ELM network of n_h hidden neurons with random input weights, and calculate the output matrix of the hidden neurons $\boldsymbol{H} \in \mathbb{R}^{N\times n_h}$.

Step 3:

1) **If** $n_h \leq N$
   Find the generalized eigenvectors $\boldsymbol{v}_2, \boldsymbol{v}_3, \ldots, \boldsymbol{v}_{n_o+1}$ of (20) corresponding to the second through the $n_o + 1$ smallest eigenvalues. Let $\boldsymbol{\beta} = [\widetilde{\boldsymbol{v}}_2, \widetilde{\boldsymbol{v}}_3, \ldots, \widetilde{\boldsymbol{v}}_{n_o+1}]$, where $\widetilde{\boldsymbol{v}}_i = \boldsymbol{v}_i/\|\boldsymbol{H}\boldsymbol{v}_i\|$, $i = 2, \ldots, n_o + 1$.
2) **Else**
   Find the generalized eigenvectors $\boldsymbol{u}_2, \boldsymbol{u}_3, \ldots, \boldsymbol{u}_{n_o+1}$ of (23) corresponding to the second through the $n_o + 1$ smallest eigenvalues. Let $\boldsymbol{\beta} = \boldsymbol{H}^T[\widetilde{\boldsymbol{u}}_2, \widetilde{\boldsymbol{u}}_3, \ldots, \widetilde{\boldsymbol{u}}_{n_o+1}]$, where $\widetilde{\boldsymbol{u}}_i = \boldsymbol{u}_i/\|\boldsymbol{H}\boldsymbol{H}^T\boldsymbol{u}_i\|$, $i = 2, \ldots, n_o + 1$.

Step 4: Calculate the embedding matrix: $\boldsymbol{E} = \boldsymbol{H}\boldsymbol{\beta}$.

Step 5 (For clustering only): Treat each row of $\boldsymbol{E}$ as a point, and cluster the N points into K clusters using the k-means algorithm. Let $\boldsymbol{y}$ be the label vector of cluster index for all the points.

return $\boldsymbol{E}$ (for embedding task) or $\boldsymbol{y}$ (for clustering task);

TABLE I
Details of the Datasets Used for Semi-Supervised Learning

Data set	Class	Dimension	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{T}\rvert$
G50C	2	50	50	314	50	136
COIL20(B)	2	1024	40	1000	40	360
USPST(B)	2	256	50	1409	50	498
COIL20	20	1024	40	1000	40	360
USPST	10	256	50	1409	50	498

It is easy to verify that both $\boldsymbol{A}$ and $\boldsymbol{B}$ are Hermitian matrices. Thus, according to the Rayleigh–Ritz theorem [53], the above trace minimization problem attains its optimum if and only if the column span of $\boldsymbol{\beta}$ is the minimum span of the eigenspace corresponding to the smallest n_o eigenvalues of (20). Therefore, by stacking the normalized eigenvectors of (20) corresponding to the smallest n_o generalized eigenvalues, we obtain an optimal solution to (19). ■

In the algorithm of Laplacian eigenmaps, the first eigenvector is discarded since it is always a constant vector proportional to $\boldsymbol{1}$ (corresponding to the smallest eigenvalue 0) [50]. In the US-ELM algorithm, the first eigenvector of (20) also leads to small variations in embedding and is not useful for data representation. Therefore, we suggest to discard this trivial solution as well.

TABLE II
PERFORMANCE COMPARISON OF THE PROPOSED SS-ELM, PURE SUPERVISED ALGORITHMS AND THE STATE-OF-THE-ART SSL ALGORITHMS

Data set	Subset	SVM	ELM	TSVM	LapRLS	LapSVM	SSELM
G50C	$\mathcal{U}$	9.33 ± 2.00	8.91 ± 2.29	5.43 ± 1.11	6.03 ± 1.32	5.52 ± 1.15	5.24 ± 1.17
	$\mathcal{V}$	9.83 ± 3.46	8.20 ± 3.05	4.67 ± 1.81	6.17 ± 3.66	5.67 ± 2.67	4.07 ± 0.95
	$\mathcal{T}$	10.06 ± 2.80	9.20 ± 2.07	4.96 ± 1.37	6.54 ± 2.11	5.51 ± 1.65	4.96 ± 1.53
COIL20 (B)	$\mathcal{U}$	16.23 ± 2.63	11.28 ± 1.95	13.68 ± 3.09	8.07 ± 2.05	8.31 ± 2.19	6.04 ± 1.26
	$\mathcal{V}$	18.54 ± 6.20	11.30 ± 2.29	12.25 ± 3.99	7.92 ± 3.96	8.13 ± 4.01	5.95 ± 1.68
	$\mathcal{T}$	15.93 ± 3.00	12.22 ± 2.00	15.19 ± 2.52	8.59 ± 1.90	8.68 ± 2.04	7.33 ± 2.15
USPST (B)	$\mathcal{U}$	17.00 ± 2.74	17.49 ± 2.44	22.75 ± 2.68	8.87 ± 1.88	8.84 ± 2.20	9.24 ± 2.79
	$\mathcal{V}$	18.17 ± 5.94	17.40 ± 2.01	21.40 ± 3.78	10.17 ± 4.55	8.67 ± 4.38	9.40 ± 2.07
	$\mathcal{T}$	17.10 ± 3.21	17.98 ± 2.86	22.06 ± 2.52	9.42 ± 2.51	9.68 ± 2.48	10.98 ± 2.99
COIL20	$\mathcal{U}$	29.49 ± 2.24	29.23 ± 1.68	24.61 ± 3.16	10.35 ± 2.30	10.51 ± 2.06	10.92 ± 2.05
	$\mathcal{V}$	31.46 ± 7.79	29.25 ± 2.81	21.25 ± 3.63	9.79 ± 4.94	9.79 ± 4.94	10.75 ± 3.02
	$\mathcal{T}$	28.98 ± 2.74	29.19 ± 3.61	23.83 ± 3.61	11.30 ± 2.17	11.44 ± 2.39	11.16 ± 1.97
USPST	$\mathcal{U}$	23.84 ± 3.26	23.91 ± 2.36	18.92 ± 2.81	15.12 ± 2.90	14.36 ± 2.55	13.51 ± 1.89
	$\mathcal{V}$	24.67 ± 4.54	24.00 ± 2.33	17.53 ± 4.29	14.67 ± 3.94	15.17 ± 4.04	13.47 ± 2.65
	$\mathcal{T}$	23.60 ± 2.32	23.85 ± 2.71	18.92 ± 3.19	16.44 ± 3.53	14.91 ± 2.83	13.85 ± 2.41

TABLE III
TRAINING TIME (IN SECONDS) COMPARISON OF TSVM, LAPRLS, LAPSVM, AND SS-ELM

Data set	TSVM	LapRLS	LapSVM	SS-ELM
G50C	0.324	0.041	0.045	0.035
COIL20(B)	16.82	0.512	0.459	0.516
USPST(B)	68.44	0.921	0.947	1.029
COIL20	18.43	5.841	4.946	0.814
USPST	68.14	7.121	7.259	1.373

Let $\gamma_1, \gamma_2, \ldots, \gamma_{n_o+1}$ $(\gamma_1 \leq \gamma_2 \leq \cdots \leq \gamma_{n_o+1})$ be the $(n_o + 1)$ smallest eigenvalues of (20) and $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{n_o+1}$ be their corresponding eigenvectors. Then, the solution to the output weights $\boldsymbol{\beta}$ is given by

$$\boldsymbol{\beta}^* = [\widetilde{\boldsymbol{v}}_2, \widetilde{\boldsymbol{v}}_3, \ldots, \widetilde{\boldsymbol{v}}_{n_o+1}] \tag{22}$$

where $\widetilde{\boldsymbol{v}}_i = \boldsymbol{v}_i / \|\boldsymbol{H}\boldsymbol{v}_i\|, i = 2, \ldots, n_o + 1$ are the normalized eigenvectors.

If the number of labeled data is smaller than the number of hidden neurons, (20) is underdetermined. In this case, we have the following alternative formulation by using the same trick as in previous sections:

$$\left(\boldsymbol{I}_u + \lambda \boldsymbol{L}\boldsymbol{H}\boldsymbol{H}^T\right)\boldsymbol{u} = \gamma \boldsymbol{H}\boldsymbol{H}^T\boldsymbol{u}. \tag{23}$$

Again, let $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{n_o+1}$ be generalized eigenvectors corresponding to the $(n_o + 1)$ smallest eigenvalues of (23), then the final solution is given by

$$\boldsymbol{\beta}^* = \boldsymbol{H}^T[\widetilde{\boldsymbol{u}}_2, \widetilde{\boldsymbol{u}}_3, \ldots, \widetilde{\boldsymbol{u}}_{n_o+1}] \tag{24}$$

where $\widetilde{\boldsymbol{u}}_i = \boldsymbol{u}_i / \|\boldsymbol{H}\boldsymbol{H}^T\boldsymbol{u}_i\|, i = 2, \ldots, n_o + 1$ are the normalized eigenvectors.

If our task is clustering, then we can adopt the k-means algorithm to perform clustering in the embedded space. We summarize the proposed US-ELM in Algorithm 2.

Remark 2: We can observe that all the supervised, semi-supervised, and unsupervised ELM have two similar stages in the training process, that is the random feature learning stage and the output weights learning stage. Under this two-stage framework, it is easy to analyze the differences and similarities between the three algorithms. Actually, all these algorithms share the same stage of random feature learning, and this is the essence of the ELM theory. This also means that for supervised, semi-supervised, or unsupervised learning problem, we can always follow the same steps to generate the hidden layer. The differences of the three types of ELMs lie in the second stage on how the output weights are computed. In supervised ELM and SS-ELM, the output weights are trained by solving a regularized least squares problem, while the output weights in the US-ELM are obtained by solving a generalized eigenvalue problem. The unified framework for the three types of ELMs might provide new perspectives to further develop the ELM theory.

VII. EXPERIMENTAL RESULTS

We evaluated our algorithms on a wide range of semi-supervised and unsupervised tasks. Comparisons were made with the related state-of-the-art algorithms, e.g., Transductive SVM (TSVM) [54], LapSVM [47], and LapRLS [47] for semi-supervised learning; and Laplacian eigenmaps (LE) [50], spectral clustering (SC) [51], and deep autoencoder (DA) [55] for unsupervised learning. All algorithms were implemented using MATLAB R2012a on a 2.60 GHz machine with 4GB of memory.

A. Semi-Supervised Learning Results

1) *Data Sets:* We tested the SS-ELM on five popular semi-supervised learning benchmarks that have been widely used for evaluating semi-supervised algorithms [52], [56], [57].

a) The G50C is a binary classification data set of which each class is generated by a 50-D multivariate Gaussian distribution. This classification problem is explicitly designed so that the true Bayes error is 5%.

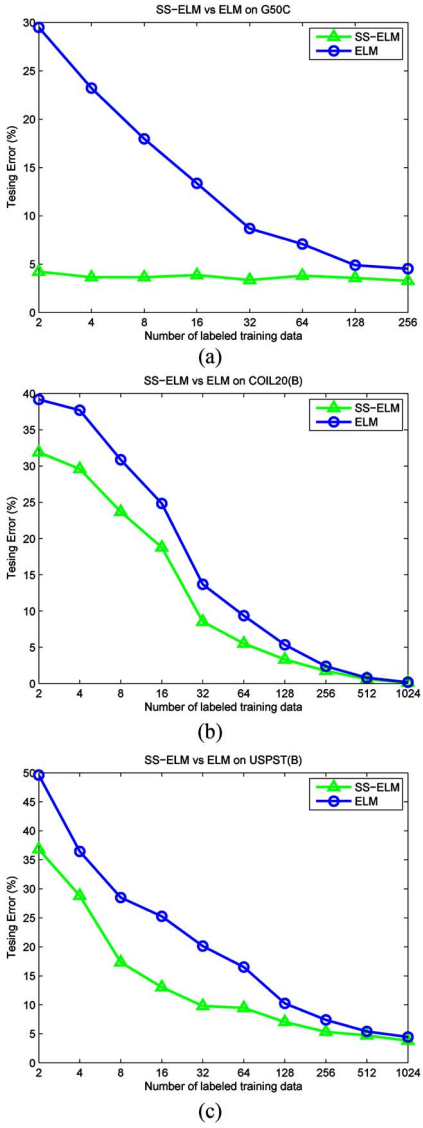


Fig. 1. Testing error with respect to different number of labeled data.

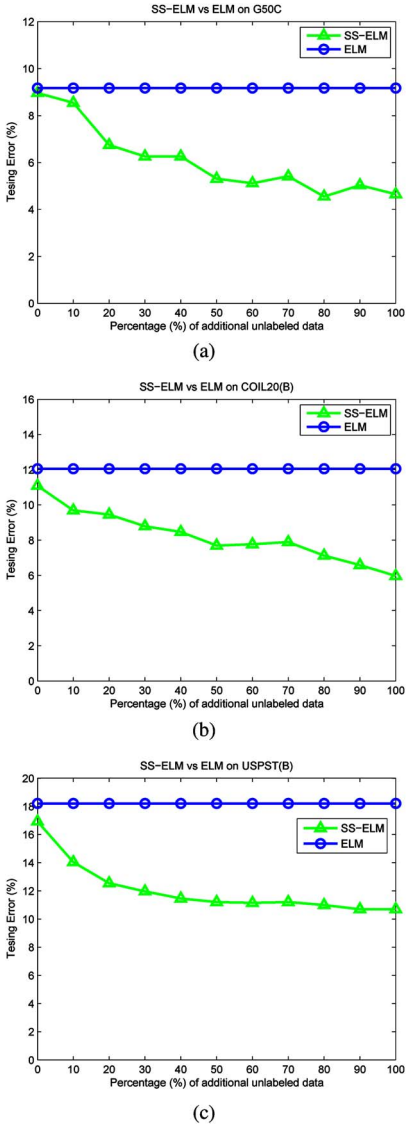


Fig. 2. Testing error with respect to different number of unlabeled data.

b) The Columbia object image library (COIL20) is a multiclass image classification data set which consists 1440 gray-scale images of 20 objects. Each pattern is a 32×32 gray scale image of one object taken from a specific view. The COIL20(B) data set is a binary classification task created from COIL20 by grouping the first ten objects as Class 1 and the last ten objects as Class 2.

c) The USPST data set is a subset (the testing set) of the well-known handwritten digit recognition data set USPS. The USPST(B) data set is a binary classification task created from USPST by grouping the first five digits as Class 1 and the last five digits as Class 2.

2) *Experimental Setup:* We followed the experimental setup in [57] to evaluate the semi-supervised algorithms. Specifically, each of the data sets is split into four folds, one of which was used for testing (denoted by $\mathcal{T}$) and the rest three folds for training. Each of the folds was used as the testing set once (4-fold cross-validation). As in [57], this random fold generation process was repeated three times, resulting in 12 different splits in total. Every training set was further partitioned into a labeled set $\mathcal{L}$, a validation set $\mathcal{V}$, and an unlabeled set $\mathcal{U}$. When we train a semi-supervised learning algorithm, the labeled data from $\mathcal{L}$ and the unlabeled data from $\mathcal{U}$ were used. The validation set which consists of labeled data

TABLE IV
DETAILS OF THE DATASETS USED FOR CLUSTERING

Data sets	Cluster	Dimension	N
IRIS	3	4	150
WINE	3	13	178
SEGMENT	7	19	2310
COIL20	20	1024	1440
USPST	10	256	2007
YALEB	15	1024	165
ORL	40	1024	400

TABLE V
PERFORMANCE COMPARISON OF THE PROPOSED US-ELM

Data set	Accuracy	k-means	DA	SC	LE	USELM
IRIS	Average	82.28 ± 13.93	89.69 ± 14.01	76.16 ± 8.92	80.85 ± 13.82	86.06 ± 15.92
	Best	89.33	97.33	84.00	89.33	97.33
WINE	Average	94.47 ± 3.85	95.24 ± 0.48	93.32 ± 11.36	96.63 ± 0	96.63 ± 0
	Best	96.63	95.51	96.63	96.63	96.63
SEGMENT	Average	60.53 ± 5.86	59.06 ± 4.03	65.64 ± 4.72	62.39 ± 5.03	64.22 ± 5.64
	Best	67.10	62.21	77.10	68.27	74.50
COIL20	Average	57.92 ± 5.44	41.43 ± 1.15	42.21 ± 4.88	64.76 ± 7.82	87.58 ± 3.47
	Best	70.56	44.86	55.21	80.00	90.35
USPST	Average	65.42 ± 3.22	69.04 ± 4.81	69.37 ± 10.39	72.44 ± 6.20	75.78 ± 5.90
	Best	72.45	80.57	84.00	81.32	87.39
YALEB	Average	38.46 ± 3.74	32.24 ± 1.70	42.32 ± 3.37	41.96 ± 3.52	44.08 ± 3.24
	Best	47.27	35.76	47.27	49.09	50.91
ORL	Average	50.86 ± 3.03	35.06 ± 1.36	52.53 ± 2.74	52.94 ± 3.20	59.26 ± 3.54
	Best	58.75	38.00	57.25	61.75	67.50

was only used for model selection, i.e., finding the optimal hyperparameters C_0 and λ in the SS-ELM algorithm. The characteristics of the data sets used in our experiment are summarized in Table I.

The training of SS-ELM consists of two stages: 1) generating the random hidden layer and 2) training the output weights using (16) or (17). In the first stage, we adopted the Sigmoid function for nonlinear mapping, and the input weights and biases were generated according to the uniform distribution on $(-1, 1)$. The number of hidden neurons n_h was fixed to 1000 for G50C, and 2000 for the rest four data sets. In the second stage, we followed the methods discussed in [52] and [57] to build the graph Laplacian $\boldsymbol{L}$, and the hyperparameter settings can be found in [47], [52], and [57]. The tradeoff parameters C and λ were selected from the exponential sequence $\{10^{-6}, 10^{-5}, \ldots, 10^{6}\}$ based on the performances on the validation set $\mathcal{V}$.

3) *Comparisons With Related Algorithms:* In this experiment, two supervised algorithms, SVMs and ELMs were used as the baseline classifiers. For comparison purposes, we also present the results obtained by three state-of-the-art semi-supervised learning algorithms, TSVM, LapRLS, and LapSVM. The classification error rates (± standard deviation) on the unlabeled set $\mathcal{U}$, the validation set $\mathcal{V}$, and the test set $\mathcal{T}$ are reported in Table II.

From the results shown in Table II, it is obvious that SS-ELM outperformed the supervised SVMs and ELMs consistently on all data sets, which demonstrate that SS-ELM is able to explore the unlabeled data effectively to achieve significantly better performance than pure supervised learning algorithms. We can also observe that SS-ELM yielded comparable accuracy with the three state-of-the-art semi-supervised learning algorithms. The results also show that when supervised ELMs and SVMs led to similar results on a certain data set, then SS-ELM and Lap-SVM also yielded comparable performances on this task. When the supervised ELMs outperformed SVMs on a data set, such as the COIL20(B), then SS-ELM gave higher accuracy as well.

4) *Computational Efficiency:* In order to evaluate the computational efficiency of SS-ELM, we present the training time of SS-ELM, TSVM, LapRLS, and LapSVM in Table III. It can be observed that the training time of SS-ELM is similar to that of LapRLS and LapSVM on the three binary classification data sets, G50C, COIL20(B), and USPST(B). However, SS-ELM is several times faster than the other two algorithms on the multiclass data sets COIL20 and USPST. This phenomenon is consistent with the empirical results given in the ELMs literature [23]; the ELM usually exhibits more evident advantage on data sets with higher number of classes.

Note that the traditional supervised ELMs are usually much more efficient than SVMs, even on binary classification problems. However, SS-ELM didn't show evident advantage over LapSVM in terms of efficiency on the binary data sets. This is due to the fact that the matrix multiplication $\boldsymbol{H}^T\boldsymbol{L}\boldsymbol{H}$ or $\boldsymbol{L}\boldsymbol{H}\boldsymbol{H}^T$ is relatively time consuming and dominated the computational cost of both SS-ELM and LapSVM. For example, on the first three binary classification data sets, SS-ELM spent 0.028 s, 0.351 s, and 0.692 s on these matrix multiplications, while it spent only 0.008 s, 0.097 s, and 0.198 s on solving the output weights. As discussed in [57], this inefficiency can be alleviated by solving the output weights using iterative methods, and we leave this for future research.

5) *Performance With Different Number of Labeled and Unlabeled Training Data:* Fig. 1 shows the performance of SS-ELM on G50C, COIL20(B) and USPST(B) with different number of labeled data. In this experiment, all the settings are same as above, except that we varied the proportion of labeled and unlabeled data in the training set. We can observe that when there is a small amount of labeled data, SS-ELM outperformed ELMs significantly.

Fig. 2 shows the performance of SS-ELM on G50C, COIL20(B) and USPST(B) with different number of unlabeled data. In this experiment, the labeled set $\mathcal{L}$, validation set $\mathcal{V}$, and

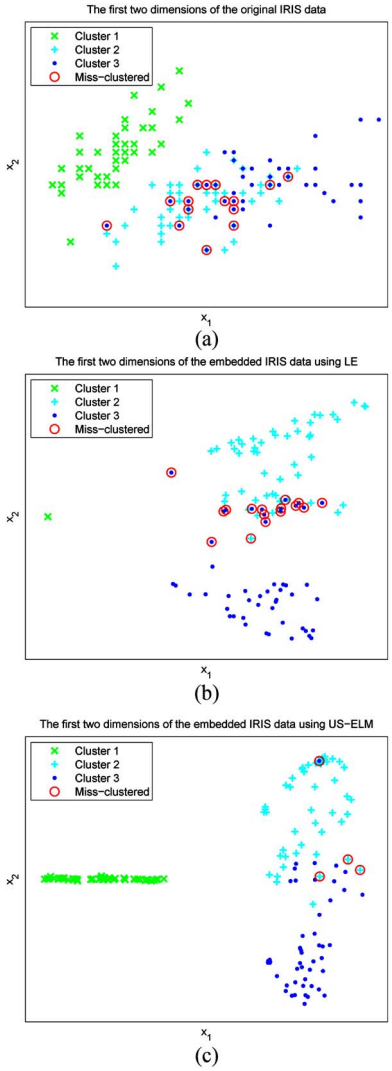


Fig. 3. Visualization of the original IRIS data, and the embedded IRIS data using LE and US-ELM.

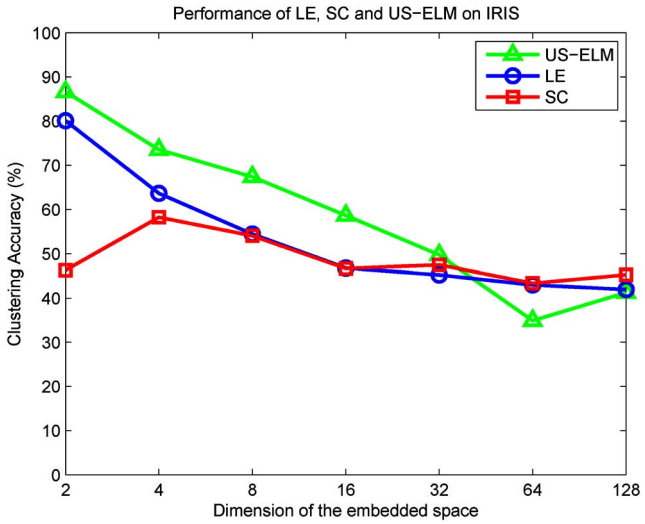


Fig. 4. Clustering accuracy on IRIS as a function of the dimension of the embedded space.

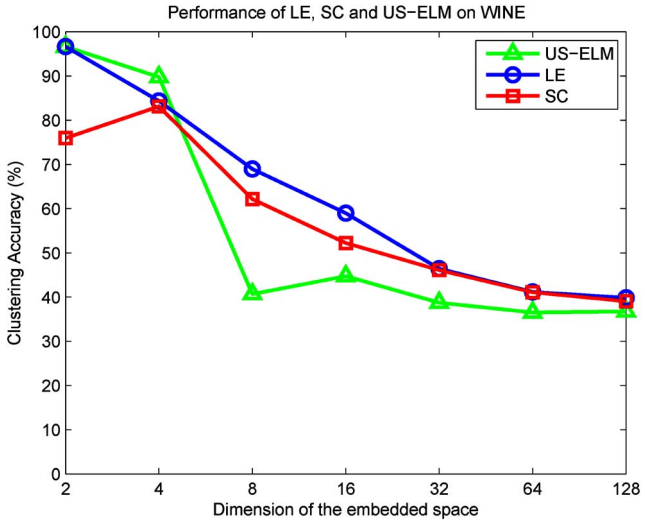


Fig. 5. Clustering accuracy on WINE as a function of the dimension of the embedded space.

test set $\mathcal{T}$ are the same as before. But we incrementally added the unlabeled patterns to the unlabeled set $\mathcal{U}$ in a unit of 10%. From the results in Fig. 2, we can observe an obvious trend that SS-ELM gives lower prediction error with more and more unlabeled data. Even without any unlabeled data, SS-ELM outperformed ELMs on these data sets. This is because that the manifold regularization is also effective for pure supervised learning [47].

B. Unsupervised Learning Results

1) *Datasets:* The data sets used for testing US-ELM include three UCI data sets (IRIS, WINE, and SEGMENT) [58], two face recognition data sets (YALEB and ORL) [59], [60], and two data sets used in Section VII-A (COIL20 and USPST). The characteristics of these data sets are summarized in Table IV.

2) *Experimental Setup:* In our experiment, we applied US-ELM to cluster all the data sets. For comparison purposes, we also report the results obtained by four other clustering algorithms, k-means, deep autoencoder (DA), Laplacian Embedding (LE), and spectral clustering (SC). The criterion for measuring clustering quality is the widely used clustering accuracy (ACC) which is defined as

$$ACC = \frac{\sum_{i=1}^{N} \delta(y_i, map(c_i))}{N} \tag{25}$$

where N is the number of training patterns, y_i and c_i are the true category label and the predicted cluster label of pattern x_i, respectively, $\delta(y_i, c)$ is a function that equals to 1 if $y = c$ or equals to 0 otherwise, $map(\cdot)$ is an optimal permutation function that maps each cluster label to a category label by the Hungarian algorithm [61].

The number of hidden neurons were set to 1000 for the first two data sets (WINE and IRIS), and 2000 for the other data sets. The hyperparameter λ was selected from the exponential sequence $\{10^{-4}, 10^{0}, \ldots, 10^{4}\}$ based on the clustering performance. For US-ELM, LE and SC, the same

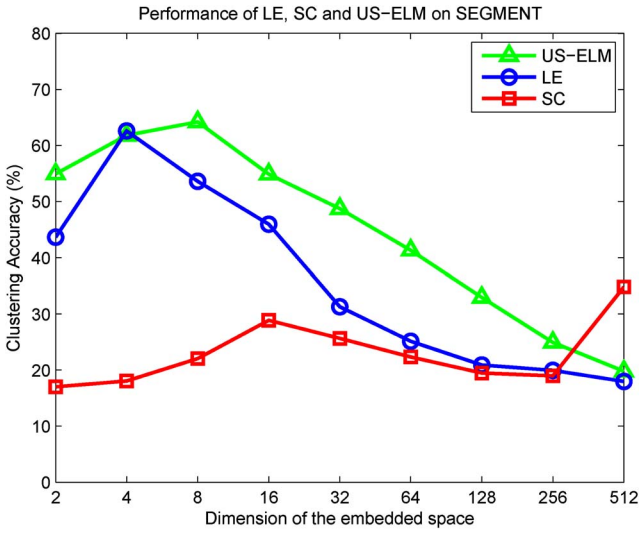


Fig. 6. Clustering accuracy on SEGMENT as a function of the dimension of the embedded space.

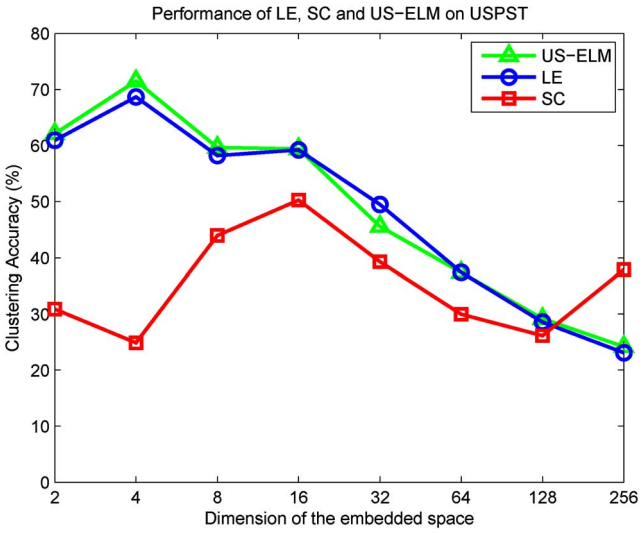


Fig. 8. Clustering accuracy on USPST as a function of the dimension of the embedded space.

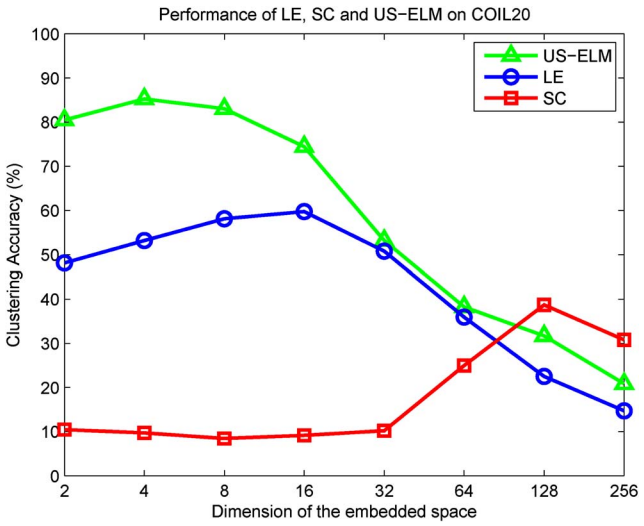


Fig. 7. Clustering accuracy on COIL20 as a function of the dimension of the embedded space.

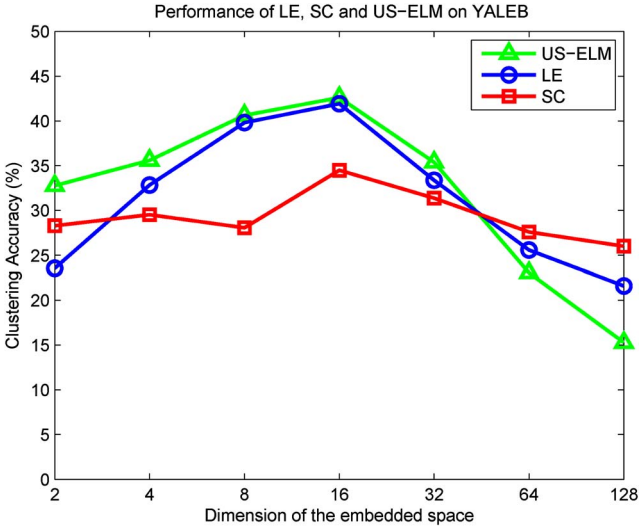


Fig. 9. Clustering accuracy on YALEB as a function of the dimension of the embedded space.

affinity matrix was used, but the dimension of the embedded space was selected independently. We ran k-means algorithm in the original space and the embedded spaces of US-ELM, DA, LE, and SC 100 times independently.

3) *Comparison With Related Algorithms:* The average and the best clustering accuracy of these algorithms are reported in Table V. From the results, we can conclude that US-ELM has obtained satisfying results on all the data sets. Considering the criterion of best clustering accuracy, US-ELM yielded the best results among the five algorithms on six out of the seven data sets. The DA algorithm leads to high accuracy on the first two small data sets, while its performance on the other data sets is not satisfying. In general, US-ELM often leads to similar results as LE due to the use of the same graph Laplacian. However, on some data sets, such as COIL20 and ORL, US-ELM yields significantly better results than LE. This may be due to the fact that US-ELM provides a way of performing regularization on the embedding through the regularization term, where the parameter λ controls the smoothness of the resulting embedding.

To compare the embedding results of US-ELM and LE, we give a visualized example using the IRIS data set. Fig. 3 shows the first 2-D of the original data, the embedded data using LE and the embedded data using US-ELM, respectively. As shown in Fig. 3(a), there are three clusters in total, and the data in the original space are not clustered well. The circles show the wrongly clustered data points by k-means. After performing embedding using LE, it can be observed from Fig. 3(b) that the data structure is more evident than it was in the original data space. However, the embedding of LE is still not satisfying. Though all the data points from Cluster 1 almost shrink to one point, the data from Cluster 2 and Cluster 3 are not well separated. In comparison, the embedding given by US-ELM shown in Fig. 3(c) is less aggressive, but leads to much better

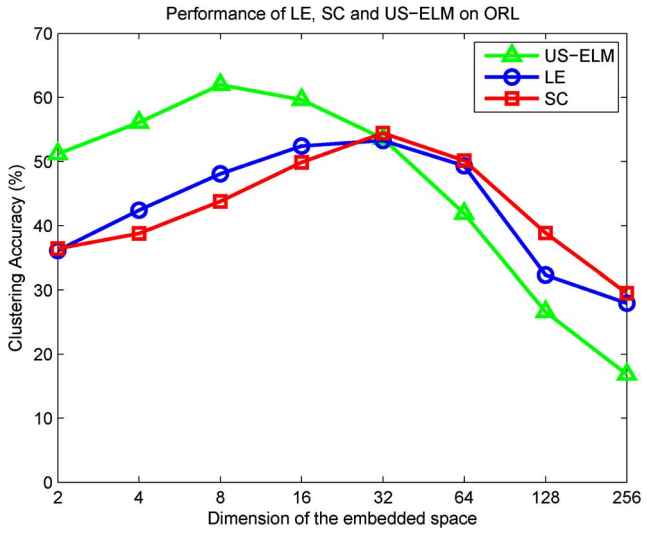


Fig. 10. Clustering accuracy on ORL as a function of the dimension of the embedded space.

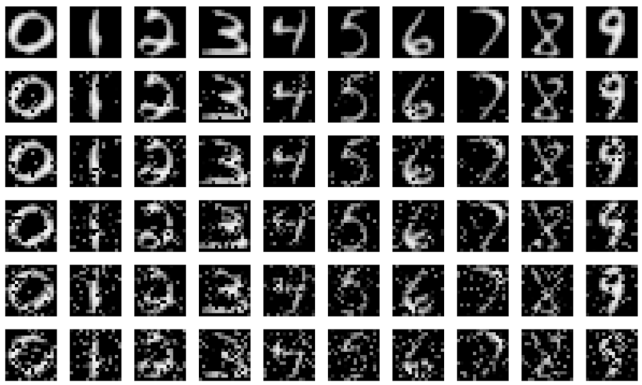


Fig. 11. Example images extracted from the clean and noise USPST data set that we used. The images shown in the first row are the noise free examples. The second through the sixth row are the contaminated images with 10%, 20%, 30%, 40%, and 50% of original pixels replaced by random values.

results. After performing k-means in the embedded space of US-ELM, the best clustering accuracy reached 97.33% (only four miss-clustered points). This example demonstrates the advantages of US-ELM for embedding and clustering.

Figs. 4–10 show the clustering accuracy of LE, SC, and US-ELM in different dimensions of embedded space. From these results, the following observations can be made.

a) The highest clustering accuracy achieved by US-ELM is generally higher than that of SC and LE.
b) The US-ELM outperformed the other two methods in a wide range of embedding dimensions.
c) The US-ELM attains its best performance with a very low dimensional embedding (usually less than 10-D), which demonstrates its effectiveness for data representation. Moveover, this also means it could be more efficient than the other two algorithms since less eigenvectors are required to be computed. If the task is clustering, then we only need to perform k-means in a relatively lower dimensional space.

4) *Computational Efficiency:* Table VI presents the training time spent on clustering using the aforementioned

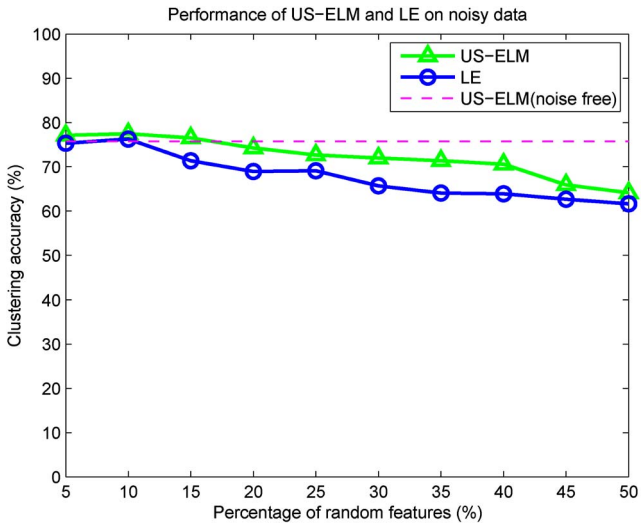


Fig. 12. Clustering accuracy of US-ELM and LE with respect to different levels of noise.

TABLE VI
TRAINING TIME (IN SECONDS) COMPARISON OF DA, SC, LE, AND US-ELM

Data set	k-means	DA	SC	LE	US-ELM
IRIS	0.004	1.624	0.014	0.017	0.040
WINE	0.005	1.452	0.029	0.037	0.041
SEGMENT	0.028	19.67	0.490	0.591	1.796
COIL20	0.139	5681	1.215	0.514	2.201
USPST	0.101	556.2	0.506	0.113	3.246
YALEB	0.021	328.5	0.026	0.087	0.156
ORL	0.035	729.2	0.068	0.148	0.318

algorithms. It is clear the k-means algorithms is the most efficient one. Among the four embedding algorithms, SC and LE are comparable in terms of efficiency, while DA is the most time consuming one. Though US-ELM is not as fast as SC and LE, its training is still efficient. We can observe from Table VI that the training of US-ELM on these data sets took only several seconds at the most.

5) *Robustness Against Noise:* To test the performance of US-ELM on data set perturbed by noise, we performed clustering on a series of data sets with different levels of noise. These contaminated data sets were created from the handwritten digits data set USPST by replacing its original features with uniform random values on (-0.5, 0.5). Fig. 11 shows some examples of the original data and the noisy data. The clustering accuracy of US-ELM and LE are presented in Fig. 12. There is a trend that the performances of both US-ELM and LE were degraded with an increased level of noise. However, the clustering accuracy of US-ELM only decreased approximately 5% when 40% of features are replaced by random values, which demonstrates its robustness against noise. Interestingly, the performance of US-ELM on the data with a low level of noise (less than 15% of corrupted features) is even higher than that on the clean data. This is in accordance with previous papers which found that deliberately corrupting features may help improving learning accuracy [62], [63]. When the level of

noise increased to 50%, both the US-ELM and LE performed significantly worse than that on the clean data. To develop robust version of US-ELM against high levels of noise or outliers is an interesting subject, and we leave it for future research.

VIII. Conclusion

In this paper, we have proposed two algorithms, SS-ELM and US-ELM, to extend the traditional ELMs for semi-supervised and unsupervised learning tasks, respectively. Compared to existing semi-supervised ELM algorithms, the proposed SS-ELM maintains almost all the advantages of ELMs, such as the remarkable training efficiency and straightforward implementation for multiclass classification problems. On a variety of data sets, SS-ELM consistently outperformed pure supervised learning algorithms such as SVMs and ELMs when auxiliary unlabeled data are available. It also led to competitive results with several state-of-the-art semi-supervised learning algorithms, and required significantly less training time on multiclass classification problems. With respect to the unsupervised learning algorithm US-ELM, to the best of our knowledge, it is the first extension of ELMs for embedding and clustering based on spectral techniques. We have evaluated US-ELM on a wide range of real world clustering tasks. Experimental results demonstrate that US-ELM gives favorable performance compared to the state-of-the-art clustering algorithms. These two extensions of ELMs for semi-supervised and unsupervised learning are expected to greatly expand the applicability of ELMs and provide new insights into the extreme learning paradigm.

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[2] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the marquardt algorithm," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 989–993, Nov. 1994.

[3] S. Chen, C. Cowan, and P. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 302–309, Mar. 1991.

[4] S. Chen and J. Wigger, "Fast orthogonal least squares algorithm for efficient subset model selection," *IEEE Trans. Signal Process.*, vol. 43, no. 7, pp. 1713–1715, Jul. 1995.

[5] K. Li, J.-X. Peng, and G. W. Irwin, "A fast nonlinear model identification method," *IEEE Trans. Autom. Control*, vol. 50, no. 8, pp. 1211–1216, Aug. 2005.

[6] E. D. Karnin, "A simple procedure for pruning back-propagation trained neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 2, pp. 239–242, Jun. 1990.

[7] G. Huang, S. Song, and C. Wu, "Orthogonal least squares algorithm for training cascade neural networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 11, pp. 2629–2637, Nov. 2012.

[8] V. Vapnik, *The Nature Of Statistical Learning Theory*. Berlin, Germany: Springer, 1999.

[9] C. Cortes and V. Vapnik, "Support vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[10] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 1999.

[11] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *J. Mach. Learn. Res.*, vol. 2, pp. 45–66, Nov. 2002.

[12] L. Zhang, W. Zhou, and L. Jiao, "Wavelet support vector machine," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 1, pp. 34–39, Feb. 2004.

[13] Y. Xia and J. Wang, "A one-layer recurrent neural network for support vector machine learning," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 2, pp. 1261–1269, Apr. 2004.

[14] G. Huang, S. Song, C. Wu, and K. You, "Robust support vector regression for uncertain input and output data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 11, pp. 1690–1700, Nov. 2012.

[15] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, vol. 2. 2004, pp. 985–990.

[16] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.

[17] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.

[18] G.-B. Huang, Y.-Q. Chen, and H. A. Babri, "Classification ability of single hidden layer feedforward neural networks," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 799–801, May 2000.

[19] R. Zhang, Y. Lan, G.-B. Huang, and Z.-B. Xu, "Universal approximation of extreme learning machine with adaptive growth of hidden nodes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 2, pp. 365–371, Feb. 2012.

[20] Y. Wang, F. Cao, and Y. Yuan, "A study on effectiveness of extreme learning machine," *Neurocomputing*, vol. 74, no. 16, pp. 2483–2490, 2011.

[21] X. Wang, A. Chen, and H. Feng, "Upper integral network with extreme learning mechanism," *Neurocomputing*, vol. 74, no. 16, pp. 2520–2525, 2011.

[22] L.-C. Shi and B.-L. Lu, "EEG-based vigilance estimation using extreme learning machines," *Neurocomputing*, vol. 102, pp. 135–143, Feb. 2013.

[23] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 513–529, Apr. 2012.

[24] H.-J. Rong, G.-B. Huang, N. Sundararajan, and P. Saratchandran, "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 4, pp. 1067–1072, Aug. 2009.

[25] Q. Liu, Q. He, and Z. Shi, "Extreme support vector machine classifier," in *Advances in Knowledge Discovery and Data Mining*. Berlin, Germany: Springer, 2008, pp. 222–233.

[26] B. Frénay and M. Verleysen, "Using SVMs with randomised feature spaces: An extreme learning approach," in *Proc. Eur. Symp. Artif. Neural Netw.*, pp. 315–320, 2010.

[27] X. Y. Liu, C. H. Gao, and P. Li, "A comparative analysis of support vector machines and extreme learning machines," *Neural Netw.*, vol. 33, pp. 58–66, Sep. 2012.

[28] X. Z. Wang, Q. Y. Shao, Q. Miao, and J. H. Zhai, "Architecture selection for networks trained with extreme learning machine using localized generalization error model," *Neurocomputing*, vol. 102, pp. 3–9, Feb. 2013.

[29] J. W. Zhao, Z. H. Wang, and D. S. Park, "Online sequential extreme learning machine with forgetting mechanism," *Neurocomputing*, vol. 87, pp. 79–89, Jun. 2012.

[30] H. J. Rong, G. B. Huang, N. Sundararajan, and P. Saratchandran, "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 4, pp. 1067–1072, Aug. 2009.

[31] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.

[32] P. Horata, S. Chiewchanwattana, and K. Sunat, "Robust extreme learning machine," *Neurocomputing*, vol. 102, pp. 31–44, Feb. 2013.

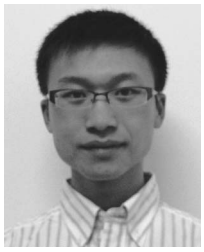
[33] Q. Yu, Y. Miche, E. Eirola, M. van Heeswijk, E. Severin, and A. Lendasse, "Regularized extreme learning machine for regression with missing data," *Neurocomputing*, vol. 102, pp. 45–51, Feb. 2013.

[34] Z. H. Man, K. Lee, D. H. Wang, Z. W. Cao, and S. Y. Khoo, "Robust single-hidden layer feedforward network-based pattern classifier," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 12, pp. 1974–1986, Dec. 2012.

[35] W. Zong, G.-B. Huang, and Y. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229–242, Feb. 2012.

[36] Q. He, C. Du, Q. Wang, F. Zhuang, and Z. Shi, "A parallel incremental extreme SVM classifier," *Neurocomputing*, vol. 74, no. 16, pp. 2532–2540, 2011.

[37] Q. He, C. Y. Du, Q. Wang, F. Z. Zhuang, and Z. Z. Shi, "A parallel incremental extreme SVM classifier," *Neurocomputing*, vol. 74, no. 16, pp. 2532–2540, 2011.
[38] S. Decherchi, P. Gastaldo, A. Leoncini, and R. Zunino, "Efficient digital implementation of extreme learning machines for classification," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 8, pp. 496–500, Aug. 2012.
[39] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*, vol. 2. Cambridge, MA, USA: MIT press, 2006.
[40] X. Zhu, "Semi-supervised learning literature survey," Dept. Comput. Sci., Univ. Wisconsin, Madison, WI, USA, Tech. Rep. 1530, 2007.
[41] O. Chapelle, V. Sindhwani, and S. S. Keerthi, "Optimization techniques for semi-supervised support vector machines," *J. Mach. Learn. Res.*, vol. 9, pp. 203–233, Feb. 2008.
[42] X. Zhu, "Cross-domain semi-supervised learning using feature formulation," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 6, pp. 1627–1638, Dec. 2011.
[43] G. Wang, F. Wang, T. Chen, D.-Y. Yeung, and F. H. Lochovsky, "Solution path for manifold regularized semisupervised classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 308–319, Apr. 2012.
[44] J. Liu, Y. Chen, M. Liu, and Z. Zhao, "SELM: Semi-supervised ELM with application in sparse calibrated location estimation," *Neurocomputing*, vol. 74, no. 16, pp. 2566–2572, 2011.
[45] L. Li, D. Liu, and J. Ouyang, "A new regularization classification method based on extreme learning machine in network data," *J. Inf. Comput. Sci.*, vol. 9, no. 12, pp. 3351–3363, 2012.
[46] K. Li, J. Zhang, H. Xu, S. Luo, and H. Li, "A semi-supervised extreme learning machine method based on co-training," *J. Comput. Inf. Syst.*, vol. 9, no. 1, pp. 207–214, 2013.
[47] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, Nov. 2006.
[48] F. Ratle, G. Camps-Valls, and J. Weston, "Semisupervised neural networks for efficient hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 5, pp. 2271–2282, May 2010.
[49] J. Weston, F. Ratle, and R. Collobert, "Deep learning via semi-supervised embedding," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1168–1175.
[50] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, 2003.
[51] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances Neural Inform. Process. Syst.*, MIT Press., vol. 2, pp. 849–856, 2002.
[52] V. Sindhwani, P. Niyogi, and M. Belkin, "Beyond the point cloud: From transductive to semi-supervised learning," in *Proc. 22nd Int. Conf. Mach. Learn.*, 2005, pp. 824–831.
[53] H. Lütkepohl, *Handbook of Matrices*. New York, NY, USA: Wiley, 1997.
[54] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. Int. Conf. Mach. Learn.*, vol. 99. 1999, pp. 200–209.
[55] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.
[56] V. Sindhwani and D. S. Rosenberg, "An RKHS for multi-view learning and manifold co-regularization," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 976–983.
[57] S. Melacci and M. Belkin, "Laplacian support vector machines trained in the primal," *J. Mach. Learn. Res.*, vol. 12, pp. 1149–1184, Mar. 2011.
[58] A. Frank and A. Asuncion. (2011). *UCI Machine Learning Repository* [Online]. Available: http://archive.ics.uci.edu/ml
[59] K.-C. Lee, J. Ho, and D. J. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 684–698, May 2005.
[60] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proc. 2nd IEEE Workshop Applicat. Comput. Vision*, 1994, pp. 138–142.
[61] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Mineola, NY, USA: Courier Dover Publications, 1998.
[62] C. M. Bishop, "Training with noise is equivalent to Tikhonov regularization," *Neural Comput.*, vol. 7, no. 1, pp. 108–116, 1995.
[63] L. Maaten, M. Chen, S. Tyree, and K. Q. Weinberger, "Learning with marginalized corrupted features," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 410–418.



Gao Huang received the B.S. degree from the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China, and is currently pursuing the Ph.D. degree with the Department of Automation, Tsinghua University, Beijing.

He was a Visiting Research Scholar with the Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO, USA, in 2013. His current research interests include machine learning and pattern recognition, especially in semi-supervised learning and robust learning.

Shiji Song received the Ph.D. degree from the Department of Mathematics, Harbin Institute of Technology, Harbin, China, in 1996.

He is currently a Professor with the Department of Automation, Tsinghua University, Beijing, China. His current research interests include system modeling, control and optimization, computational intelligence, and pattern recognition.

Jatinder N. D. Gupta received the Ph.D. degree in industrial engineering from Texas Tech University, Lubbock, TX, USA.

He is currently an Associate Dean for graduate and sponsored programs. He is also an Eminent Scholar and Professor with the College of Business Administration, and a Professor of industrial and systems engineering and engineering management, University of Alabama in Huntsville, Huntsville, AL, USA. His current research interests include supply chain management, information security, information, and decision technologies, scheduling, organizational learning, and effectiveness.

Cheng Wu received the B.S. and M.S. degrees in electrical engineering from Tsinghua University, Beijing, China.

Since 1967, he has been with Tsinghua University, where he is currently a Professor with the Department of Automation. His current research interests include system integration, modeling, scheduling, and optimization of complex industrial systems.

Mr. Wu is a member of the Chinese Academy of Engineering.