Journal of Classification (2020) 37:656–670
https://doi.org/10.1007/s00357-019-09352-2

An Optimal Weight Semi-Supervised Learning Machine for Neural Networks with Time Delay

Chengbo Lu[1] · Ying Mei[1]

Published online: 10 December 2019

Abstract
In this paper, an optimal weight semi-supervised learning machine for a single-hidden layer feedforward network (SLFN) with time delay is developed. Both input weights and output weights of the SLFN are globally optimized with manifold regularization. By feature mapping, input vectors can be placed at the prescribed positions in the feature space in the sense that the separability of all nonlinearly separable patterns can be maximized, unlabeled data can be leveraged to improve the classification accuracy when labeled data are scarce, and a high degree of recognition accuracy can be achieved with a small number of hidden nodes in the SLFN. Some simulation examples are presented to show the excellent performance of the proposed algorithm.

Keywords Neural networks · Optimal weight learning · Semi-supervised learning · Manifold regularization · Time delay

1 Introduction

Feedforward neural networks (FNNs) have been extensively used in classification applications and regressions (Bishop 1995). One of FNNs single-hidden layer feedforward networks (SLFNs) play an important role. Recently, an interesting learning algorithm named extreme learning machine (ELM) (Huang et al. 2006) is proposed for SLFNs, where the input weights of an SLFN are uniformly randomly selected, and the output weights are trained with the batch learning type of least squares.

Most ELMs have been concerned with supervised task. However, in many practical applications such as text classification, information retrieval, and fault diagnosis, obtaining labels for fully supervised learning is time consuming and hard to get, while unlabeled data are easy and cheap to collect. In (Liu et al. 2011), a semi-supervised learning method SELM (semi-supervised extreme learning machine) by manifold regularization framework was

✉ Chengbo Lu
lu.chengbo@aliyun.com

[1] Department of Mathematics, Lishui University, Lishui, Zhejiang 323000, China

proposed. There are two advantages of the proposed SELM. First, it employs graph Laplacian regularization to import large number of unlabeled samples which can dramatically reduce labeled calibration samples. Based on SELM, another semi-supervised learning method named SS-ELM was proposed in (Huang et al. 2014), which further consider structural risk further, and introduce a regularization term which can control the complexity of the model. However, we still hope that the robustness and classification recognition accuracy of both the SELM and SSELM should be improved further. For example, when the input weights and the hidden layer biases of an SLFN are randomly assigned, the changes of the hidden layer output matrix of the SLFN sometimes may be very large. This in turn will result in the large changes of the output weight matrix. According to the statistical learning theory (Man et al. 2013; Vapnik 1998; Devroye et al. 1996), the large changes of the output weight matrix will lead to the increase of both the structural and empirical risks of the SLFNs, which will in turn degenerate the robustness property of SLFNs with respect to the input disturbances. Therefore, we consider designing the optimal input weights such that the hidden layer of SLFNs can improve the robustness with respect to disturbances, and reduce both empirical risks of labeled training samples and structural risks, and thus further improve the classification accuracy.

In this paper, we develop an optimal weight semi-supervised learning (OSSL) with both the linear nodes and the input tapped delay line to handle semi-supervised problems by the manifold regularization framework. Borrowing the concept of model reference control from control engineering (Duda and Hart 1973), we use the SLFN classifier with the nonlinear hidden nodes, trained with the ELM in (Huang et al. 2006), as the reference classifier that provides the reference feature vectors for the outputs of the hidden layer of the proposed SLFN classifier to follow. The input weights are then optimized to assign the feature vectors to the "desired reference positions" defined by the reference feature vectors, and maximize the separability of all feature vectors in the feature space. In terms of the optimization of the output weights, the recognition accuracy at the output layer of the SLFN classifier can be further improved.

The rest of this paper is organized as follows: in section 2, the basics of the ELM are briefly described. In section 3, firstly the SLFN classifier with both linear nodes and a tapped delay line is formulated, then the optimizations of the input weights and output weights with the manifold regularization theory are explored in detail. In section 4, the SLFN classifiers with the ELM, the SELM, and the SS-ELM are compared to show the performance of the new SLFN classifier with the OSSL on some semi-supervised tasks. Section 5 gives the conclusions.

2 Brief of ELM

2.1 Nomenclature

A_{ij} The weight connects vertex i and j
c Positive parameter
$\boldsymbol{C}$ A $(l+u)\times(l+u)$ diagonal matrix with its first l diagonal elements 1 and the rest equal to 0
$\boldsymbol{D}^*$ A diagonal matrix
$\boldsymbol{e}$ The matrix of the errors between $\boldsymbol{T}$ and $\boldsymbol{O}$
$\boldsymbol{H}$ The hidden layer output matrix
$\boldsymbol{I}$ The unit matrix

$G(\boldsymbol{w}, b, \boldsymbol{x})$The activation of the ELM
l The number of input labeled samples
$\boldsymbol{L}$ The graph Laplacian
m The dimension of $\boldsymbol{t}_i$
M The number of input samples
n The dimension of $\boldsymbol{x}_i$
$\widetilde{N}$ The number of hidden nodes
$\boldsymbol{o}_i$ The output vector of the SLFN with respect to the input vector $\boldsymbol{x}_i$
$\boldsymbol{O}$ The output of a SLFN with respect to $\boldsymbol{X}$
$S(f)$ The smoothness function
$\boldsymbol{T}$ The training target data matrix
$\boldsymbol{T}_l$ The labeled output data matrix
u The number of input unlabeled samples
$\boldsymbol{w}_i$, b_i The learning parameters generated randomly of the ith hidden node
$\boldsymbol{W}$ The inner weight matrix
$\boldsymbol{W}_0$Randomly input weight matrix
$(\boldsymbol{x}_i, \ \boldsymbol{t}_i)$ The ith training sample
$\boldsymbol{X}_l$ The input labeled pattern matrix
$\boldsymbol{X}_u$ The input unlabeled pattern matrix
$\boldsymbol{X}$ The input pattern matrix
$\boldsymbol{Y}^*$ Desired reference feature matrix

2.2 Greek Symbols

$\boldsymbol{\beta}_i$ The weight vector connecting the ith hidden node to the output nodes
$\boldsymbol{\beta}$ The output weight matrix
$\boldsymbol{\lambda}$ The Lagrange multiplier matrix
ε The matrix of the errors between the reference feature vectors and the feature vectors
$\mathcal{T}$The test set
$\mathcal{V}$ The validation set
θ_1, γ_1, μ_1, θ_2, γ_2, μ_2 The positive real regularization parameters

The basics of the ELM are briefly introduced as follows:

Given M distinct training samples $(\boldsymbol{x}_i, \ \boldsymbol{t}_i) \in R^n \times R^m$ $(i = 1, 2, \ldots, M)$, the output of a SLFN with $\widetilde{N}$ hidden nodes can be represented by

$$\boldsymbol{o}_j = \sum_{i=1}^{\widetilde{N}} \boldsymbol{\beta}_i G\left(\boldsymbol{w}_i, b_i, \boldsymbol{x}_j\right), \quad \boldsymbol{w}_i \in R^n, \quad \boldsymbol{\beta}_i \in R^m, \tag{2.1}$$

where $\boldsymbol{o}_j \in R^m$ is the output vector of the SLFN with respect to the input vector $\boldsymbol{x}_j$.

$\boldsymbol{w}_i = [w_{i1}, w_{i2}, \cdots, w_{in}]^T$and $b_i \in R$ $\left(i = 1, 2, \ldots, \widetilde{N}\right)$ are learning parameters generated randomly of the ith hidden node. $\boldsymbol{W} = \left[\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{\widetilde{N}}\right]$ is referred as inner weight matrix. $\boldsymbol{\beta}_i = [\beta_{i1}, \beta_{i2}, \cdots, \beta_{im}]^T$ is weight vector connecting the ith hidden node to the output nodes, $\boldsymbol{\beta} = \left[\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_{\widetilde{N}}\right]$ is referred as output weight matrix, and $G(\boldsymbol{w}_i, b_i, \boldsymbol{x}_j)$ is the activation of the ELM.

Set $\boldsymbol{w}_i \cdot \boldsymbol{x}_j$ be the inner product of $\boldsymbol{w}_i$ and $\boldsymbol{x}_j$. Equation (2.1) can be written compactly as

$$\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{O} \tag{2.2}$$

with

$$\boldsymbol{H} = \begin{pmatrix} G(\boldsymbol{w}_1 \cdot \boldsymbol{x}_1 + \boldsymbol{b}_1) & \cdots & G\left(\underset{\sim}{\boldsymbol{w}}_N \cdot \boldsymbol{x}_1 + \underset{\sim}{b}_N\right) \\ \vdots & \ddots & \vdots \\ G(\boldsymbol{w}_1 \cdot \boldsymbol{x}_M + b_1) & \cdots & G\left(\underset{\sim}{\boldsymbol{w}}_N \cdot \boldsymbol{x}_M + \underset{\sim}{b}_N\right) \end{pmatrix},$$

$$\boldsymbol{O} = [\boldsymbol{o}_1, \boldsymbol{o}_2, \cdots, \boldsymbol{o}_M]^T.$$

Let $\boldsymbol{T} = [\boldsymbol{t}_1, \boldsymbol{t}_2, \cdots, \boldsymbol{t}_M]^T$, to minimize the network cost function $\|\boldsymbol{O} - \boldsymbol{T}\|$, Eq.. (2.2) can be written as

$$\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{T}. \tag{2.3}$$

Solving Eq. (2.3), we obtain the following output weight matrix:

$$\boldsymbol{\beta} = \boldsymbol{H}^{+}\boldsymbol{T},$$

where $\boldsymbol{H}^{+}$ is the Moore–Penrose generalized inverse of the matrix $\boldsymbol{H}$.

The orthogonal projection method can be efficiently used in ELM: $\boldsymbol{H}^{+} = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T$ if $\boldsymbol{H}^T\boldsymbol{H}$ is nonsingular or $\boldsymbol{H}^{+} = \boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{H}^T)^{-1}\boldsymbol{H}^T$ else. In terms of the ridge regression theory, it was suggested that a positive parameter $\frac{1}{c}$ is added to the diagonal $\boldsymbol{H}^T\boldsymbol{H}$ or $\boldsymbol{H}\boldsymbol{H}^T$in the computation of the output weights $\boldsymbol{\beta}$. The resultant solution is more stable and tends to have better generalization performance. That means, in order to improve the stability of ELM, we can have

$$\boldsymbol{\beta} = \left(\frac{1}{c}\boldsymbol{I} + \boldsymbol{H}^T\boldsymbol{H}\right)^{-1}\boldsymbol{H}^T\boldsymbol{H},$$

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 are more than the number of the hidden neurons. Or we can have

$$\boldsymbol{\beta} = \boldsymbol{H}^T\left(\frac{1}{c}\boldsymbol{I} + \boldsymbol{H}\boldsymbol{H}^T\right)^{-1}\boldsymbol{T}.$$

3 Optimal Weight Semi-Supervised ELM

The time delay neural network proposed by Waibel has been frequently used on phoneme recognition and trajectory recognition (Waibel et al. 1989). case where the number of training patternssuch applications as speech recognition and are highly competitive with existing technologies. In these years, researchers began to pay attention to recurrent neural networks with time delay

(Wu et al. 2010; Chen et al. 2008; Wu et al. 2012; Chen et al. 2014). Figure 1 shows a single-hidden layer feedforward network with linear nodes and an input tapped delay line (Man et al. 2011), where the output layer has m linear nodes; the hidden layer has $\widetilde{N}$ linear nodes; w_{ij} $(i = 1, n, j = 1, \widetilde{N})$ are the input weights; β_{ij} $(i = 1, \widetilde{N}, j = 1, m)$ are the output weights; y_i $(i = 1, \widetilde{N})$ are the outputs of the hidden nodes; a string of Ds, seen at the input layer, are $n-1$ unit delay elements that ensure the input sequence x(k), x(k − 1), ⋯, x(k − n + 1), represent a time series type of data, and consist of both the present and the past observations of the process.

Since $n-1$ unit delay elements are added to the input layer, every hidden node in the SLFN performs as an nth-order finite impulse response (FIR) filter that is often used to approximate a linear or a nonlinear function by properly choosing the input weights in signal processing and control engineering. Also, in practice, the design and analysis of the SLFNs with linear nodes are much easier than the ones of the SLFNs with nonlinear nodes (Man et al. 2013).

Denote the labeled data in the training set as $\{\boldsymbol{X}_l, \boldsymbol{T}_l\} = \{\boldsymbol{x}_i(k), \boldsymbol{t}_i\}_{i=1}^{l}$ and unlabeled data as $\boldsymbol{X}_u = \{\boldsymbol{x}_i(k)\}_{i=1}^{u}$ (here $\boldsymbol{x}_i(k) = [\mathrm{x_i(k)}, \mathrm{x_i(k-1)}, \cdots, \mathrm{x_i(k-n+1)}]^T \in R^n$, l and u are the number of labeled and unlabeled samples, respectively). The $M = l + u$ linear output equations of the SLFN in Fig. can be obtained as

$$\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{O},$$

with

$$\boldsymbol{H} = [\boldsymbol{x}_1(k)\ \ \boldsymbol{x}_2(k)\cdots\ \boldsymbol{x}_l(k)\cdots\ \boldsymbol{x}_{l+u}(k)]^T\boldsymbol{W}$$

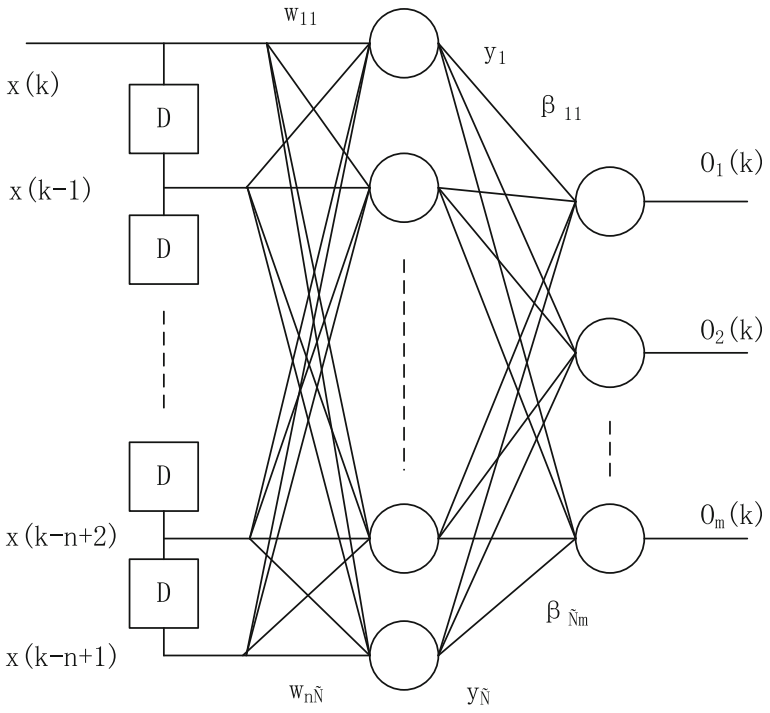

Fig. 1 A single-hidden layer neural network with linear nodes and an input tapped delay line

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{x}_1(k)^T \boldsymbol{w}_1 & \cdots & \boldsymbol{x}_1(k)^T \underset{\sim}{w}_N \\ \vdots & \ddots & \vdots \\ \boldsymbol{x}_{l+u}(k)^T \boldsymbol{w}_1 & \cdots & \boldsymbol{x}_{l+u}(k)^T \underset{\sim}{\boldsymbol{w}}_N \end{pmatrix} = \boldsymbol{X}^T \boldsymbol{W}, \tag{3.1}$$

where $\boldsymbol{X} = [\boldsymbol{x}_1(k)\ \ \boldsymbol{x}_2(k)\cdots\ \boldsymbol{x}_l(k)\cdots\ \boldsymbol{x}_{l+u}(k)]$.

The matrix $\boldsymbol{H}$ in Eq. (3.1) contains all feature vectors $\boldsymbol{Y}_1, \boldsymbol{Y}_2, \cdots, \boldsymbol{Y}_{l+u}$, corresponding to the input data vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{l+u}$, respectively. For example, for the ith given input pattern vector $\boldsymbol{x}_i(k)$, the corresponding feature vector is the ith row of the matrix $\boldsymbol{H}$

$$\boldsymbol{Y}_i = \left[\boldsymbol{x}_i{}^T(k)\boldsymbol{w}_1\ \ \boldsymbol{x}_i{}^T(k)\boldsymbol{w}_2\ \ \cdots \boldsymbol{x}_i{}^T(k)\underset{\sim}{\boldsymbol{w}}_N \right]^T = \boldsymbol{W}^T \boldsymbol{x}_i(k).$$

Then we have

$$\boldsymbol{Y} = \boldsymbol{W}^T \boldsymbol{X}$$

with

$$\boldsymbol{Y} = [\boldsymbol{Y}_1\ \ \boldsymbol{Y}_2 \cdots \boldsymbol{Y}_M].$$

Belkin et al. (Belkin et al. 2004) constructed a semi-labeled graph $\boldsymbol{G}$ based on labeled and unlabeled samples. Each sample $\boldsymbol{x}_j$ represents one vertex j, and if j is one of the neighbors of i ($i \sim j$), an edge with the weight A_{ij} connects them. Furthermore, a function f is defined on the graph $\boldsymbol{G}$, and the label values of the training samples are regarded as the observed function values on the corresponding vertices. Assume that both the label data $\boldsymbol{X}_l$ and the unlabeled data $\boldsymbol{X}_u$ are drawn from the same marginal distribution, and if two input patterns $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are close to each other, then $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$ should be close as well. To enforce this assumption on the data, it can be formalized by the smoothness function:

$$S(f) = \sum_{i \sim j} A_{ij} \left\| f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j) \right\|^2 \tag{3.2}$$

where "$i \sim j$" means the vertex j is one of the neighbors of i. f is the vector of the n values of $f(x)$ on training data, $f = [f(\boldsymbol{x}_i), \boldsymbol{x}_i \in \boldsymbol{X}_l \cup \boldsymbol{X}_u]$.

Let $f(\boldsymbol{x}) = \boldsymbol{W}^T \boldsymbol{x}$, Eq. (3.2) can be simplified in a matrix form

$$S(f) = Tr\left(\boldsymbol{Y}\boldsymbol{L}\boldsymbol{Y}^T\right),$$

where $Tr(\cdot)$ is the trace of a matrix, $\boldsymbol{L}$ is the graph Laplacian which can be computed as: $\boldsymbol{L} = \boldsymbol{D}^* - \boldsymbol{A}$, where $\boldsymbol{D}^*$ is a diagonal matrix given by $D^*_{ii} = \sum_{j=1}^{l+u} A_{ij}$. There are some types of weights $\boldsymbol{A}$; Gaussian function $e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$ is usually used to compute the weights $\boldsymbol{A}$.

Intuitively, $S(f)$ penalizes large variation in $f(\boldsymbol{x})$ when input data $\boldsymbol{x}$ has a small change. This requires that $f(\boldsymbol{x})$ vary smoothly with $\boldsymbol{x}$.

Now, let the M reference feature vectors described in section 1 be

$$\boldsymbol{Y}^* = \left[\boldsymbol{Y}^*{}_1\ \ \boldsymbol{Y}^*{}_2 \cdots \boldsymbol{Y}^*{}_M\right] \tag{3.3}$$

Conventionally, the selection of the N desired feature vectors in Eq. (3.3) is mainly based on one's understanding of the characteristics of the input patterns. However, in this

paper, we will use the ELM feature vectors as the "desired reference feature vectors" for the ones generated by the outputs of the SLFN in Fig. to follow. Through optimizing the input weights of the SLFN in Fig. , we can place the feature vectors of the SLFN in Fig. at the "desired position" in feature space. The purpose of such feature vectors' assignment is to further maximize the separability of the feature vectors so that the classification or recognition accuracy, seen from the output layer of the SLFN, can be greatly improved, compared with the SLFN classifiers in Fig. , trained with the ELM, SELM, and the SS-ELM.

In the following, we will use the manifold regularization technique and the batch training methodology to design an optimal weight semi-supervised learning (OSSL) which optimizes both the input weight matrix $\boldsymbol{W}$ and the output weight matrix $\boldsymbol{\beta}$ in the sense that (i) the error between the reference feature vectors and the feature vectors generated by the hidden layer of the SLFN classifier in Fig. can be minimized; (ii) the error between the desired output pattern and the actual output pattern with respect to labeled input data is minimized; (iii) unlabeled data are leveraged to improve the classification accuracy when labeled data are scarce.

Then the design of the optimal input weights of the SLFN can be formulated by the following optimization problem

$$\text{Minimize} \quad \frac{\theta_1}{2}\|\boldsymbol{\varepsilon}\|^2 + \frac{\gamma_1}{2}Tr\left(\boldsymbol{YLY}^T\right) + \frac{\mu_1}{2}\|\boldsymbol{W}\|^2 , \tag{3.4}$$

$$\text{subject to } \boldsymbol{\varepsilon} = \boldsymbol{Y}^* - \boldsymbol{Y} = \boldsymbol{Y}^* - \boldsymbol{W}^T\boldsymbol{X}, \tag{3.5}$$

where $\boldsymbol{L} \in R^{M\times M}$ is the graph Laplacian built from all data; μ_1, γ_1, and θ_1 are the positive real regularization parameters; $\|\boldsymbol{W}\|^2$ is the regularizer which, through the proper choice of the regularization parameters μ_1, θ_1,and γ_1, is used to smooth the cost function at the singular point of the correlation matrix of the feature vectors to avoid the ill-posed inverse of the data matrix and improve the robustness of the SLFN with the respect to noisy environment.

The optimization problem in Eq. (3.4) with the constraint in Eq. (3.5) can be solved by using the method of Lagrange multipliers, with the following Lagrange functions:

$$L_{ij} = \frac{\theta_1}{2}\sum_{i=1}^{N}\sum_{j=1}^{\widetilde{N}}\varepsilon_{ij}{}^2 + \frac{\gamma_1}{2}\sum_{j=1}^{N}\sum_{i=1}^{n}w_{ij}\sum_{t=1}^{n}s_{it}w_{tj}$$

$$+\frac{\mu_1}{2}\sum_{i=1}^{n}\sum_{j=1}^{\widetilde{N}}w_{ij}{}^2 - \sum_{k=1}^{\widetilde{N}}\sum_{p=1}^{N}\lambda_{kp}\left(Y_{kp} - \sum_{v=1}^{n}w_{vk}x_{pv} - \varepsilon_{kp}\right),$$

where w_{ij} is the ijth element of the input weight matrix $\boldsymbol{W}$, ε_{ij} is the ijth element of the error matrix $\boldsymbol{\varepsilon}$ defined in Eq. (3.5), Y_{ij} is the ijth element of the reference feature vector matrix $\boldsymbol{Y}^*$ defined in Eq. (3.3), s_{ij} is the ijth element of matrix $\boldsymbol{S} = \boldsymbol{XLX}^T$, and λ_{ij} is the ijth Lagrange multiplier.

Differentiating L_1 with respect to w_{ij}, we have

$$\frac{\partial L_{ij}}{\partial w_{ij}} = \mu_1 w_{ij} + \gamma_1 \sum_{t=1}^{n} s_{it} w_{tj} + \frac{\partial}{\partial w_{ij}} \left(\sum_{k=1}^{\widetilde{N}} \sum_{p=1}^{N} \lambda_{kp} \left(\sum_{v=1}^{n} w_{vk} x_{pv} + \varepsilon_{kp} \right) \right)$$

$$= \mu_1 w_{ij} + \gamma_1 \sum_{t=1}^{n} s_{it} w_{tj} + \sum_{p=1}^{N} \lambda_{jp} x_{pi} \quad ,$$

Letting $\frac{\partial L_{ij}}{\partial w_{ij}} = 0$ lead

$$\mu_1 w_{ij} = -\gamma_1 \sum_{t=1}^{n} s_{it} w_{tj} - \sum_{p=1}^{N} \lambda_{jp} x_{pi}$$

$$= -\gamma_1 \begin{pmatrix} w_{1j} & w_{2j} & \cdots & w_{nj} \end{pmatrix} \begin{pmatrix} s_{i1} \\ s_{i2} \\ \vdots \\ s_{in} \end{pmatrix} - \begin{pmatrix} \lambda_{j1} & \lambda_{j2} & \cdots & \lambda_{jN} \end{pmatrix} \begin{pmatrix} x_{1i} \\ x_{2i} \\ \vdots \\ x_{Ni} \end{pmatrix},$$

and

$$\mu_1 \begin{pmatrix} w_{11} & w_{21} & \cdots & w_{n1} \end{pmatrix}$$

$$= -\gamma_1 \begin{pmatrix} w_{11} & w_{21} & \cdots & w_{n1} \end{pmatrix} \begin{pmatrix} s_{11} & \cdots & s_{n1} \\ \vdots & \ddots & \vdots \\ s_{1n} & \cdots & s_{nn} \end{pmatrix} - \begin{pmatrix} \lambda_{11} & \lambda_{12} & \cdots & \lambda_{1N} \end{pmatrix} \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{Nn} \end{pmatrix} ,$$

Then

$$\mu_1 \begin{pmatrix} w_{11} & \cdots & w_{n1} \\ \vdots & \ddots & \vdots \\ w_{1\widetilde{N}} & \cdots & w_{n\widetilde{N}} \end{pmatrix} = -\gamma_1 \begin{pmatrix} w_{11} & \cdots & w_{n1} \\ \vdots & \ddots & \vdots \\ w_{1\widetilde{N}} & \cdots & w_{n\widetilde{N}} \end{pmatrix} \begin{pmatrix} s_{11} & \cdots & s_{n1} \\ \vdots & \ddots & \vdots \\ s_{1n} & \cdots & s_{nn} \end{pmatrix}$$

$$- \begin{pmatrix} \lambda_{11} & \cdots & \lambda_{1N} \\ \vdots & \ddots & \vdots \\ \lambda_{\widetilde{N}1} & \cdots & \lambda_{\widetilde{N}N} \end{pmatrix} \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{Nn} \end{pmatrix} ,$$

or

$$\mu_1 \boldsymbol{W} = -\gamma_1 \boldsymbol{S}\boldsymbol{W} - \boldsymbol{X}\boldsymbol{\lambda}^T.$$

Since $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^T$, then

$$\mu_1 \boldsymbol{W} = -\gamma_1 \boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^T \boldsymbol{W} - \boldsymbol{X}\boldsymbol{\lambda}^T. \tag{3.6}$$

Then, differentiating L_{ij} with respect to ε_{ij}, we have

$$\frac{\partial L_{ij}}{\partial \varepsilon_{ij}} = \theta_1 \varepsilon_{ij} + \lambda_{ij} \quad .$$

Solving $\frac{\partial L_{ij}}{\partial \varepsilon_{ij}} = 0$, we have

$$\lambda_{ij} \quad = -\theta_1 \varepsilon_{ij},$$

or

$$\boldsymbol{\lambda} = -\theta_1 \boldsymbol{\varepsilon}. \tag{3.7}$$

Considering the constraint in Eq. (3.5), we can express Eq. (3.7) as

$$\boldsymbol{\lambda} = -\theta_1 \left(\boldsymbol{Y}^* - \boldsymbol{W}^T \boldsymbol{X} \right), \tag{3.8}$$

and using Eq. (3.8) in Eq. (3.6) leads to

$$\mu_1 \boldsymbol{W} = -\gamma_1 \boldsymbol{X} \boldsymbol{L} \boldsymbol{X}^T \boldsymbol{W} + \theta_1 \boldsymbol{X} \boldsymbol{Y}^{*T} - \theta_1 \boldsymbol{X} \boldsymbol{X}^T W. \tag{3.9}$$

Then, we can derive the optimal input weight matrix $\boldsymbol{W}$ as follows:

$$\boldsymbol{W} = \left(\frac{\mu_1}{\theta_1} \boldsymbol{I}_n + \boldsymbol{X} \boldsymbol{X}^T + \frac{\gamma_1}{\theta_1} \boldsymbol{X} \boldsymbol{L} \boldsymbol{X}^T \right)^{-1} \boldsymbol{X} \boldsymbol{Y}^{*T}. \tag{3.10}$$

If the number of labeled data M is fewer than the length of the input vectors n, then $\boldsymbol{X}$ will have more rows than columns, which often leads to an underdetermined least squares problem. To solve this problem, we premultiply both sides of Eq. (3.9) by $\boldsymbol{X}^T(\boldsymbol{X}^T\boldsymbol{X})^{-1}$, then

$$\boldsymbol{W} = \boldsymbol{X} \left(\frac{\mu_1}{\theta_1} \boldsymbol{I}_M + \boldsymbol{X}^T \boldsymbol{X} + \frac{\gamma_1}{\theta_1} \boldsymbol{L} \boldsymbol{X}^T \boldsymbol{X} \right)^{-1} \boldsymbol{Y}^{*T}, \tag{3.11}$$

where $\boldsymbol{I}_M$ is an unit matrix with order M.

Remark 1 The Lagrange multiplier matrix $\boldsymbol{\lambda}$in Eq. (3.8) describes the sensitivity of the objective function in Eq. (3.4) with respect to the constraint in Eq. (3.5), that is, how tight the constraint in Eq. (3.5) is binding at the optimal point with the optimal input weights in Eqs. (3.10) and (3.11). It is seen from Eq. (3.8) that as the value of the regularization parameter θ_1 becomes smaller, the cost function in Eq. (3.4) is less sensitive with respect to the change of the constraint in Eq. (3.5). Thus, the Lagrange multiplier matrix $\boldsymbol{\lambda}$ plays the role of qualitatively estimating the effects of the input disturbance, the structural risk, and the empirical risk on the robustness of the SLFN classifier with the OSSL, seen from the output of the hidden layer.

Remark 2 The proper choice of the regularization parameters μ_1 and θ_1 makes the augmented cost function in Eq. (3.4) smoother around the minimum point (or vertex) of the cost function in the input weight space. However, the dynamic behaviors of the hidden layer of the SLFN are also related to the slope of the augmented cost function in Eq. (3.4) that is adjusted by the regularization parameters θ_1 and γ_1; this point can be indirectly seen from Eqs. (3.10) and (3.11). Now, we further consider the effects of the regularization parameters θ_1 and γ_1 on the robustness of the output of the hidden layer. It is seen from Eq. (3.4) that the value of the regularization parameters θ_1 and γ_1 affects the width or the steepness of the regularized cost function in Eq. (3.4). For instance, if the value of the regularization parameter θ_1 is very large, the slope of the cost function in Eq. (3.4) will be very steep. The hidden layer of the SLFN is thus very sensitive to the change of the input weights and input disturbances. However, if the value of the regularization parameter θ_1 is very small, the changing rate of the cost function in Eq. (3.4) will be very small, and the hidden layer of the SLFN is thus insensitive to the changes of the input weights and input disturbances. If γ_1 is very large, then the hidden layer of the

SLFN is very sensitive to the differences of feature vectors with respect to similar input samples; similarly, if θ_1 is very small, the changing rate of the cost function in Eq. (3.4) will be very small too. Therefore, it is necessary to properly choose the regularization parameters μ_1 and θ_1 so that the robustness of the hidden layer can be guaranteed.

Similarly, the design of the optimal output weights of the SLFN can be formulated by the following optimization problems:

$$\text{Minimize} \quad \frac{\theta_2}{2}\|\boldsymbol{e}\|^2 + \frac{\gamma_2}{2}Tr\left(\boldsymbol{O}^T\boldsymbol{L}\boldsymbol{O}\right) + \frac{\mu_2}{2}\|\boldsymbol{\beta}\|^2 \tag{3.12}$$

$$\text{subject to } \boldsymbol{e} = \boldsymbol{C}(\boldsymbol{T}-\boldsymbol{H}\boldsymbol{\beta}), \tag{3.13}$$

where μ_2, θ_2, and γ_2 are the positive real regularization parameters; $\boldsymbol{O}$ is the output matrix with respect to M input vectors; $\boldsymbol{C}$ is a $M \times M$ diagonal matrix with its first l diagonal elements 1 and the rest equal to 0; $\boldsymbol{T} \in R^{M\times m}$ is the training target with its first l rows equal to $\boldsymbol{T}_l$ and the rest equal to 0.

Again, the optimization problem in Eq. (3.12) with the constraint in Eq. (3.13) can be solved by using the method of Lagrange multipliers.

$$\boldsymbol{\beta} = \left(\frac{\mu_2}{\theta_2}\boldsymbol{I}_{\widetilde{N}} + \boldsymbol{H}^T\boldsymbol{C}\boldsymbol{H} + \frac{\gamma_2}{\theta_2}\boldsymbol{H}^T\boldsymbol{L}\boldsymbol{H}\right)^{-1}\boldsymbol{H}^T\boldsymbol{C}\boldsymbol{T} \tag{3.14}$$

Similarly, if the number of labeled data M is fewer than the number of hidden neurons $\widetilde{N}$, we get the following alternative solutions:

$$\boldsymbol{\beta} = \boldsymbol{H}^T\left(\frac{\mu_2}{\theta_2}\boldsymbol{I}_M + \boldsymbol{C}\boldsymbol{H}\boldsymbol{H}^T + \frac{\gamma_2}{\theta_2}\boldsymbol{L}\boldsymbol{H}\boldsymbol{H}^T\right)^{-1}\boldsymbol{C}\boldsymbol{T}. \tag{3.15}$$

Remark 3 It is seen from Eqs. (3.14) and (3.15) that the optimal output layer weight matrix $\boldsymbol{\beta}$ depends on the ratio $\frac{\mu_2}{\theta_2}$ and $\frac{\gamma_2}{\theta_2}$ instead of values of μ_2, θ_2, and γ_2. However, the sensitivity of the objective function in Eq. (3.12) with respect to the constraint in Eq. (3.13) depends not only on the ratio $\frac{\mu_2}{\theta_2}$ and $\frac{\gamma_2}{\theta_2}$ but also on the values of μ_2, θ_2, and γ_2. Thus, theses three regularization parameters and two ratios should be chosen to be properly small for ensuring the tradeoff between the sensitivity and the smoothness of the SLFN classifier.

Now, the OSSL algorithm can be summarized as follows:

A dataset with labeled samples and unlabeled samples is given: labeled samples $\{(\boldsymbol{x}_i, \boldsymbol{t}_i) | \boldsymbol{x}_i \in R^n, \boldsymbol{t}_i \in R^m, i = 1, 2, \cdots, l\}$; and unlabeled samples $\{(\boldsymbol{x}_i) | \boldsymbol{x}_i \in R^n, i = 1, 2, \cdots, u\}$. And some parameters such as the number of hidden nodes $\widetilde{N}$ and regularization parameters θ_1, γ_1, μ_1, θ_2, γ_2, μ_2 are specified; also, activation function $G(\boldsymbol{x})$ is selected. Then seven steps are as follows:

Step 1: Assign randomly input weight matrix $(\boldsymbol{W}_0)_{n\times\widetilde{N}}$;

Step 2: Compute the reference feature matrix $\boldsymbol{Y}^* = \boldsymbol{W}_0{}^T\boldsymbol{X}$;

Step 3: Construct the graph Laplacian $\boldsymbol{L} = \boldsymbol{D}^* - \boldsymbol{A}$ from both X_l and X_u;

Step 4: Choose the suitable regularization parameters θ_1, γ_1, μ_1,if $M > m$,compute the input weight $\boldsymbol{W}$ by using Eq. (3.10);

else

compute the input weight $\boldsymbol{W}$ by using Eq. (3.11);

Step 5: Compute the hidden layer output matrix $\boldsymbol{H}$ by using Eq. (3.1);
Step 6: Choose the suitable regularization parameters θ_2, γ_2, μ_2,
if $M > \widetilde{N}$compute the output weight $\boldsymbol{\beta}$ by using Eq. (3.14);
else
compute the output weight $\boldsymbol{\beta}$ by using Eq. (3.15);
Step 7: Use the model $f = \boldsymbol{H}\boldsymbol{\beta}$ to make prediction.

4 Experimental Results

In our experiments, all the algorithms are run in such computer environment: (1) operating system Windows 7 Enterprise; (2) CPU i5-3570 (tm); (3) memory 8 GB; (4) simulating software Matlab R2013a.

We selected five popular data sets for our experiment; all of them are available on the web.

G50C is an artificial binary classification data set generated by a 50-dimensional multivariate Gaussian distribution with equal probabilities. The class means are adjusted so that the Bayes error is 5%.

The Columbia Object Image Library (COIL20) is a multiclass image classification data set which consists of 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 obtained from COIL20 by grouping the first 10 objects as class 1, and the last 10 objects as class 2.

The USPST data set is a subset (the testing set) of the well-known handwritten digit recognition data set USPS. It is a collection of handwritten digits from the USPS postal system. Images are acquired at the resolution of 16×16 pixels. USPST refers to the test split of the original data set. The USPST (B) data set is a binary classification task obtained from USPST by grouping the first 5 digits as class 1 and the last 5 digits as class 2.

The details of the described data sets are resumed in Table 1.

All presented results have been obtained by averaging them on different splits of the available data. In particular, a 4-fold cross-validation has been performed, randomizing the fold generation process for 3 times, for a total of 12 splits. Each fold contains the same number of per class examples as in the complete data set. For each split, we have 3 folds that are used for training the classifier and the remaining one that constitutes the test set ($\mathcal{T}$). Training data has been divided in labeled ($\boldsymbol{X}_l$), unlabeled ($\boldsymbol{X}_u$), and validation sets ($\mathcal{V}$), where the last one is only used to tune the classifier parameters. The labeled and validation sets have been randomly selected from the training data such that at least one example per class is assured to be present on each of them, without any additional balancing constraints. When we train a semi-supervised learning algorithm, the labeled data from $\boldsymbol{X}_l$

Table 1 Details of the data sets used for semi-supervised learning

Data set	Class	Dimension	$\lvert\boldsymbol{X}_l\rvert$	$\lvert\boldsymbol{X}_u\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{T}\rvert$
G50C	2	50	50	314	50	136
COLL20(B)	2	1024	40	1000	40	360
USPST(B)	2	256	50	1409	50	498
COLL20	20	1024	40	1000	40	360
USPST	2	256	50	1409	50	498

and the unlabeled data from $\boldsymbol{X}_u$ were used. The validation set which consists of labeled data was only used for finding the optimal parameters in the OSSL algorithm. The parameters θ_1, γ_1, μ_1, θ_2, γ_2, μ_2 were determined by varying them on the grid $\{10^{-6}, 10^{-5}, \cdots, 10^{6}\}$ based on the performance on the validation set $\mathcal{V}$. Five different numbers of nodes for each data set were chosen according to some preliminary experiments.

In this experiment, the supervised algorithm ELM was used as the baseline classifiers. For comparison purposes, we also present the results obtained by two semi-supervised learning algorithms, SELM and SSELM. The recognition accuracies (± standard deviation) on the test set $\mathcal{T}$ are reported in Table 2. It shows that the semi-supervised algorithm OSSL achieves the highest degree of recognition accuracy and the smallest standard deviation compared with the supervised algorithm ELM and the semi-supervised algorithms SELM and SSELM in most cases as the number of hidden nodes is increased from 25 to 200. The smallest standard deviations of the OSSL also show a stronger robustness with respect to such as the changes of the digit writing styles in USPST set, which are equivalent to the bounded input noises. Obviously, the optimization of both the input weights and the output weights of the SLFN with the OSSL plays a very important role of greatly improving the recognition performance.

Figure 2 shows the performance of ELM, SELM, SSELM, and OSSL on G50C, COIL20(B), USPST(B), COIL20, and USPST with different number of labeled and unlabeled data. In this experiment, all the numbers of hidden neurons were fixed to 100 and other settings are the same as the experiment in Table 2, except that we varied the proportion of labeled and unlabeled data in the training set. It is seen that OSSL outperformed in most cases compared with the recognition results with the ELM, SELM, and SSELM.

Table 2 Performance comparison of the ELM, SELM, SSELM, and OSSL algorithms

Data set	Nodes	ELM	SELM	SSELM	OSSL
G50C	25	70.55 ± 5.62	72.45±5.19	73.72±5.06	80.16±3.58
	50	74.45±4.65	74.78±4.82	75.26±4.45	80.78±3.83
	100	79.73±4.52	78.10±5.21	77.38±4.39	82.36±3.47
	150	81.32±3.58	80.48±4.32	82.03±4.78	83.87±4.13
	200	83.25±3.14	83.97±4.34	83.12±4.32	85.34±3.35
COLL20(B)	25	66.10±5.09	67.58±4.79	67.09±5.52	82.11±5.02
	50	67.40±5.24	71.65±4.75	72.93±4.65	86.32±4.28
	100	73.65±4.67	73.36±4.87	76.88±5.77	86.78±4.02
	150	77.65±4.56	79.31±4.34	80.67±4.16	87.93±3.86
	200	80.15±4.28	83.44±3.56	82.20±4.21	90.77±2.78
USPST(B)	25	65.93±6.11	66.35±2.37	65.82±1.91	64.13±2.56
	50	67.34±4.23	63.56±2.57	64.02±2.64	69.54±2.37
	100	71.18±4.58	71.45±1.69	72.03±1.78	71.38±3.45
	150	73.98±4.59	74.88±2.14	73.65±2.01	75.47±3.12
	200	75.23±4.09	77.56±2.78	75.78±1.77	79.56±1.49
COLL20	25	36.87±3.72	38.87±4.15	37.97±4.23	64.88±3.45
	50	48.15±4.67	48.68±4.64	50.23±4.86	69.15±4.21
	100	52.31±4.56	57.54±5.67	55.13±4.89	71.07±4.43
	150	58.64±4.31	59.34±4.12	60.12±4.55	72.13±4.16
	200	60.35±4.78	63.55±4.36	62.45±4.87	74.59±3.87
USPST	25	45.61±4.63	50.11±3.98	52.21±4.01	61.77±4.32
	50	55.78±4.84	56.78±4.43	58.06±4.39	65.23±3.82
	100	59.49±5.23	66.43±4.65	65.34±3.95	66.22±3.43
	150	65.88±4.12	66.89±4.74	67.48±3.80	67.05±3.15
	200	68.87±4.45	67.56±4.04	70.43±3.83	72.58±3.56

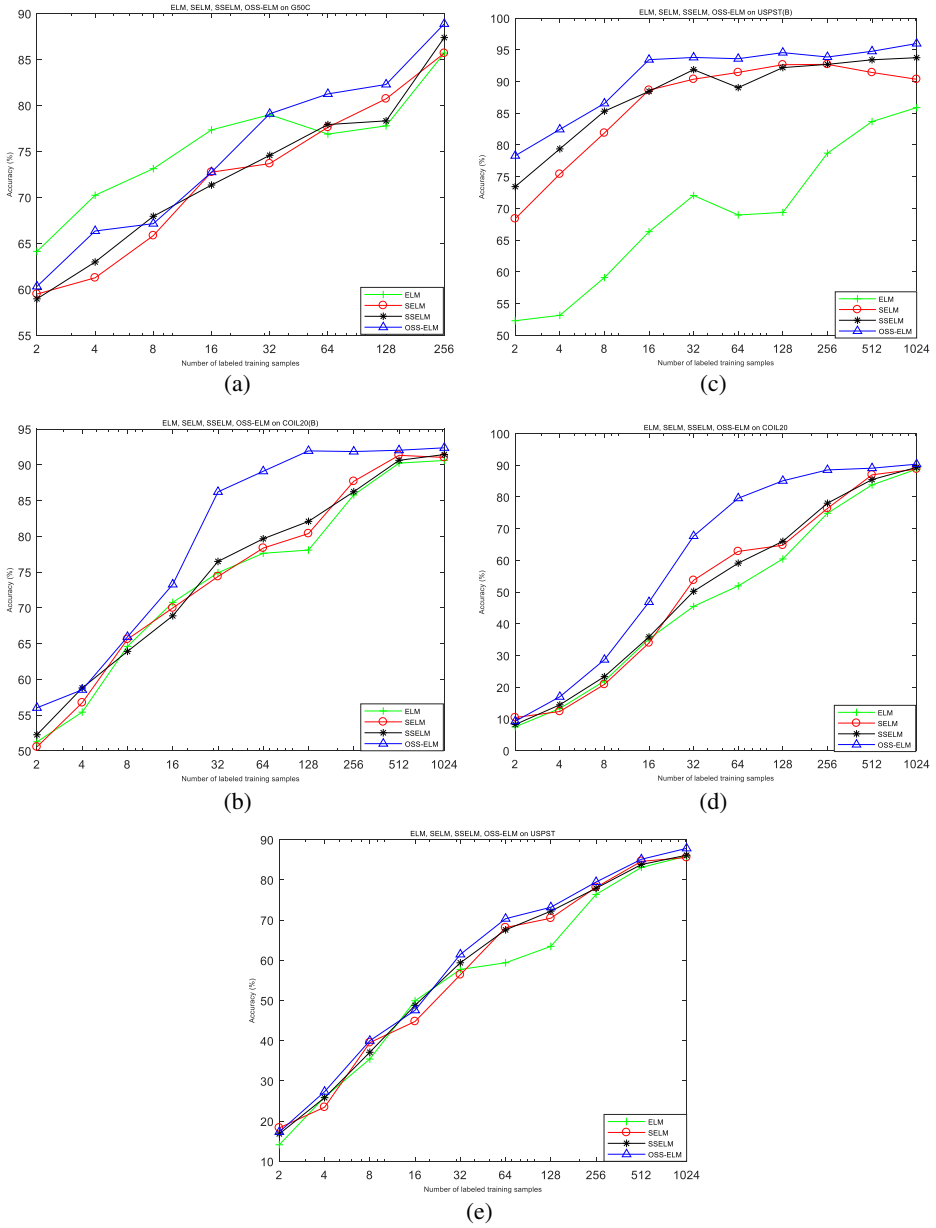

Fig. 2 Accuracy with respect to different number of labeled data

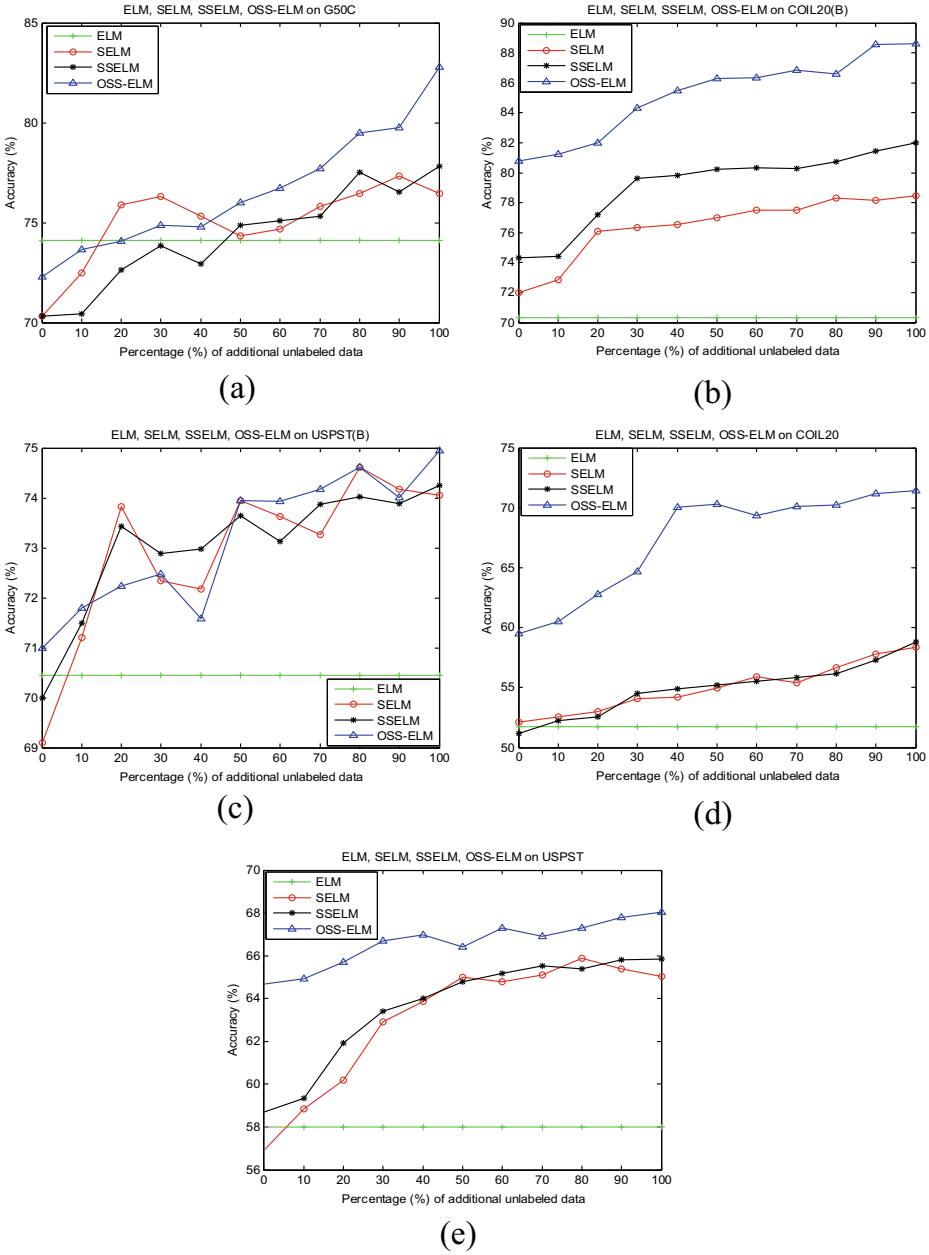

Fig. 3 Accuracy with respect to different number of unlabeled data

Figure 3 shows the performance of OSSL on G50C, COIL20(B), USPST(B), COIL20, and USPST with different number of unlabeled data. In this experiment, all the numbers of hidden neurons were fixed to 100; the labeled set $\boldsymbol{X}_l$, validation set $\mathcal{V}$, and test set $\mathcal{T}$ are the same as before. But we incrementally added the unlabeled patterns to the unlabeled set $\boldsymbol{X}_u$ in a unit of 10%. It is noted again that the classifier with the OSSL performs the best compared with the ones with the ELM, the SELM, and SSELM.

5 Conclusion

In this paper, we have developed an optimal weight semi-supervised extreme learning machine with time delay to extend the original ELM for semi-supervised learning tasks. Because of the optimal designs of both the input and the output weights, the OSSL has demonstrated an excellent performance for a wide range of data sets.

Acknowledgments The authors thank the anonymous referee for his careful reading of the manuscript and his fruitful comments and suggestions.

Funding Information This research was partially supported by Zhejiang Provincial Natural Science Foundation of China under Grant No. LY18F030003 and Science & Technology Program of Lishui City under Grant No. 2017RC01.

References

M. Belkin, I. Matveeva, P. Niyogi, Regularization and semi-supervised learning on large graphs, in: *Proceedings of 17th conference on learning theory (COLT)*, 2004.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. New York: Oxford. University Press Inc.

Chen, Y., Xue, A. K., Lu, R. Q., & Zhou, S. S. (2008). On robustly exponential stability of uncertain neutral systems with time-varying delays and nonlinear perturbations. *Nonlinear Analysis, 68*, 2464–2470.

Chen, Y., Lu, R. Q., Zou, H. B., & Xue, A. K. (2014). Stability analysis for stochastic jump systems with time-varying delay. *Nonlinear Analysis: Hybrid Systems, 14*, 114–125.

L. Devroye, L. Gyorfi, And G. Lugosi, *A probabilistic theory of pattern recognition*. New York: Springer-Verlag, 1996.

Duda, R., & Hart, P. (1973). *Pattern classification and scene analysis*. New York: Wiley.

Huang, G. B., Zhu, Q. Y., & Siew, C. K. (Dec. 2006). Extreme learning machine: theory and applications. *Neurocomputing, 70*(1–3), 489–501.

Huang, G., Song, S., Gupta, J. N. D., & Wu, C. (2014). Semi-supervised and unsupervised extreme learning machines. *IEEE Transactions on Cybernetics, 44*(12), 2405–2417.

Liu, J., Chen, Y., Liu, M., & Zhao, Z. (2011). SELM: semi-supervised ELM with application in sparse calibrated location estimation. *Neurocomputing, 74*(16), 2566–2572.

Man, Z., Lee, K., Wang, D. H., Cao, Z., & Miao, C. (2011). A new robust training algorithm for a class of single hidden layer neural networks. *Neurocomputing, 74*, 2491–2501.

Man, Z., Lee, K., Wang, D., Cao, Z., & Khoo, S. (2013). An optimal weight learning machine for handwritten digit image recognition. *Signal Processing, 93*(6), 1624–1638.

Vapnik, V. (1998). *Statistical learning theory*. New York: Wiley.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., & Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing, 37*(3), 328–339.

Z. G. Wu, H. Su, J Chu, W. N. Zhou. Improved delay-dependent stability condition of discrete recurrent neural networks with time-varying delays, IEEE Transactions on Neural Networks, 21(4): 692–697, 2010.

Wu, Z. G., Lam, J., Su, H. Y., & Chu, J. (2012). Stability and dissipativity analysis of static neural networks with time delay. *IEEE Transactions on Neural Networks and Learning Systems, 23*(2), 199–210.