Neurocomputing 000 (2017) 1–10

Contents lists available at ScienceDirect

Neurocomputing

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

# Twin extreme learning machines for pattern classification

Yihe Wan[a,b], Shiji Song[a,*], Gao Huang[a], Shuang Li[a]

[a] Department of Automation, Tsinghua University, Beijing 100084, China
[b] Naval Academy of Armament, Beijing 100161, China

ARTICLE INFO

*Article history:*
Received 27 April 2016
Revised 21 April 2017
Accepted 22 April 2017
Available online xxx

Communicated by Huiyu Zhou

*Keywords:*
Twin extreme learning machine
Pattern classification
Extreme learning machine
Twin support vector machine
Nonparallel separating hyperplane

ABSTRACT

Extreme learning machine (ELM) is an efficient and effective learning algorithm for pattern classification. For binary classification problem, traditional ELM learns only one hyperplane to separate different classes in the feature space. In this paper, we propose a novel *twin extreme learning machine* (TELM) to simultaneously train two ELMs with two nonparallel classification hyperplanes. Specifically, TELM first utilizes the random feature mapping mechanism to construct the feature space, and then two nonparallel separating hyperplanes are learned for the final classification. For each hyperplane, TELM jointly minimizes its distance to one class and requires it to be far away from the other class. TELM incorporates the idea of twin support vector machine (TSVM) into the basic framework of ELM, thus TELM could have the advantages of the both algorithms. Moreover, compared to TSVM, TELM has fewer optimization constraint variables but with better classification performance. We also introduce a successive over-relaxation technique to speed up the training of our algorithm. Comprehensive experimental results on a large number of datasets verify the effectiveness and efficiency of TELM.



## 1. Introduction

Single hidden layer feedforward networks (SLFNs) have been extensively studied, and a large number of algorithms have been proposed based on SLFNs in the past few decades. These algorithms can be roughly divided into three categories. The first category adopts gradient methods to optimize the weights in the network, such as error back-propagation algorithm [1], Levenberg Marquardt algorithm [2], and neuron by neuron algorithm [3]. However, these gradient-based methods usually suffer from slow convergence or local minima issues.

The second one corresponds to the standard optimization based methods, such as the support vector machine (SVM) or its variants [4–8]. Due to its powerful classification and approximation capabilities, SVM is widely adopted in the classification and regression problems. To the specific, SVM constructs two parallel support hyperplanes which can separate the positive and negative data points very well. In order to reduce the generalization errors, SVM aims to find the solution of maximizing the width of two parallel support hyperplanes while minimizing the training errors by solving a quadratic programming problem (QPP).

The third category mainly contains the least square based methods, such as all kinds of extreme learning machines (ELMs) [9–12]. Because of the effectiveness and efficiency, ELM has been extensively studied recently. ELM has two remarkable features: (1) Unlike the conventional function approximation approaches, ELM randomly generates the input weights and hidden layer biases, and fixes them without tuning iteratively. Then the output weights can be determined analytically and efficiently. (2) ELM aims to minimize both training errors and the norm of output weights, which leads ELM to generalize well on the unseen testing data. Compared with the gradient-based algorithms, ELM spends much less time on training and tends to achieve much better generalization performance. Many studies [11] show that ELM can be comparable with or even better than the standard SVM in terms of the prediction accuracy. Moreover, since solving the least squares problem is faster than solving the QPP, ELM is usually much more efficient than SVM.

For binary classification problem, comparing ELM with SVM from the perspective of classification mechanism, ELM and its variants focus on finding a separating hyperplane, which passes through the origin of the ELM random feature space [13]. Whereas SVM aims to learn two parallel support hyperplanes to separate the testing data apart. Though ELM and SVM become more and more popular in a wide range of domains recently. In some cases, it is still difficult to achieve satisfying performance only by exploiting one separating hyperplane

* Corresponding author.
*E-mail addresses:* wanyh12@mails.tsinghua.edu.cn (Y. Wan), shijis@mail.tsinghua.edu.cn (S. Song), huang-g09@mails.tsinghua.edu.cn (G. Huang), l-s12@mails.tsinghua.edu.cn (S. Li).

http://dx.doi.org/10.1016/j.neucom.2017.04.036
0925-2312/

or two parallel separating hyperplanes, for such as the cross data.

To expand the applicability of SVM, a famous twin support vector machine (TSVM) [14] is proposed, which targets at deriving two nonparallel separating hyperplanes. Each hyperplane tends to reach the smallest distance to one of the two classes and makes its distance to the other class as large as possible. By solving two reduced-sized QPPs, TSVM could learn faster than the standard SVM. The variants of TSVM [15–17] have been widely studied and exploited in the classification fields.

In this paper, motivated by TSVM, we propose a novel twin extreme learning machine (TELM) algorithm, which aims to learn two nonparallel separating hyperplanes in the ELM feature space for data classification. For each hyperplane, TELM minimizes its distance to one of the two classes and requires it to be far away from the other class. In order to alleviate over-fitting problems, TELM allows an acceptable training error by minimizing a regularization term jointly. Specifically, TELM tries to minimize both the training error and the sum of squares of the distance from one hyperplane to one of the two classes. Thus, TELM simultaneously trains two ELMs based on the optimization method, and has inherited the merits of ELM and TSVM.

It is worth noting that TELM is different from the methods proposed by Ning et al. [18], which aim to construct a prediction interval. In [18], two twin ELM models: regularized asymmetric least squares ELM (RALS-ELM) and asymmetric Bayesian ELM (AB-ELM) are proposed. In RALS-ELM, the asymmetric least squared error loss function with different weights is used, and AB-ELM exploits asymmetric Gaussian distribution as the likelihood function of the model output. RALS-ELM and AB-ELM could obtain two similar ELMs by setting a pair of weights. Thus, an upper-bound and a lower-bound regression curve can be deduced by RALS-ELM and AB-ELM, respectively, which leads to calculate the prediction interval. However, TELM mainly deal with the classification problem.

Similar to TSVM, the proposed TELM generates two nonparallel separating hyperplanes by solving a pair of QPPs. However, they are different in several aspects: (1) for the objective function and the corresponding constraints, the bias $b$ is not required in TELM, whereas in TSVM, the bias $b$ is an important optimization item; (2) in terms of two nonparallel separating hyperplanes in TELM, they pass through the origin, however, in TSVM they are basically not passing through the origin; (3) random feature mapping is adopted in the TELM, i.e., TELM initializes an network with random input weights and biases, and projects training sample into the random feature space; (4) in TELM, many nonlinear continuous functions can be utilized as activation functions, and thus TELM can be trained on diverse nonlinear feature mappings. We have evaluated the proposed algorithm on a large number of datasets comprehensively, and compare it with several related state-of-the-art algorithms. Experimental results manifest that TELM can outperform other algorithms in terms of the prediction accuracy and efficiency.

The rest of the paper is organized as follows: in Section 2, we give a brief review of SVM, TSVM and ELM. The proposed twin extreme learning machine algorithm is proposed in Section 3. In Section 4, the complete TELM and some remarks are described. Then, we give discussions about TELM in Section 5. Section 6 describes experimental details and results, and we conclude this paper in Section 7.

## 2. Background

In this section, we will briefly introduce SVM, TSVM and ELM.

### 2.1. Support vector machine

Suppose we have training data $\{\boldsymbol{x}_i, t_i\}_{i=1}^N$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ is the input pattern and $t_i \in \{+1, -1\}$ is the corresponding output. SVM aims to minimize the generalization error by maximizing the margin between two parallel support hyperplanes. When the input patterns are strictly linearly separable, to maximize the separating margin $2/\|\boldsymbol{w}\|$ is equivalent to:

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} \\ s.t. \quad & t_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1, \end{aligned} \tag{1}$$

where $\boldsymbol{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$.

When the input patterns are not linearly separable, we can utilize a mapping function $\boldsymbol{\psi}(\boldsymbol{x})$: $\boldsymbol{x}_i \to \boldsymbol{\psi}(\boldsymbol{x}_i)$ to project the input pattern $\boldsymbol{x}_i$ from the original input space to a SVM feature space $\Psi$. Here, the relax variable $\xi_i$ associated with the $i$th input pattern can be introduced, and we can rewritten (1) as:

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{N}\xi_i \\ s.t. \quad & t_i(\boldsymbol{w}\cdot\boldsymbol{\psi}(\boldsymbol{x}_i) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, \end{aligned} \tag{2}$$

where $C$ is a penalty coefficient. The dual function of (2) is

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} t_i t_j \alpha_i \alpha_j \boldsymbol{\psi}(\boldsymbol{x}_i)\cdot\boldsymbol{\psi}(\boldsymbol{x}_i) - \sum_{i=1}^{N}\alpha_i \\ s.t. \quad & \sum_{i=1}^{N} t_i\alpha_i = 0 \\ & 0 \leq \alpha_i \leq C, \end{aligned} \tag{3}$$

where $\alpha_i$ is the Lagrange multiplier corresponding to the input data $\boldsymbol{x}_i$. Then the decision function of SVM can be calculated as:

$$f(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{s=1}^{N_s}\alpha_s t_s \boldsymbol{\psi}(\boldsymbol{x})\cdot\boldsymbol{\psi}(\boldsymbol{x}_s) + b\right), \tag{4}$$

where $\alpha_s$ corresponds to support vectors $\boldsymbol{x}_s$, $N_s$ is the number of support vectors.

### 2.2. Twin support vector machine

Assume that the matrix $\boldsymbol{A} \in \mathbb{R}^{m_1\times d}$ and $\boldsymbol{B} \in \mathbb{R}^{m_2\times d}$ represent the input data belonging to class $+1$ and class $-1$, respectively, where $m_1$, $m_2$ are the numbers of the input data in the corresponding class.

TSVM searches for two nonparallel separating hyperplanes

$$\boldsymbol{f}_1(\boldsymbol{x}) = \boldsymbol{w}_1^T\boldsymbol{x} + b_1 = 0 \quad \text{and} \quad \boldsymbol{f}_2(\boldsymbol{x}) = \boldsymbol{w}_2^T\boldsymbol{x} + b_2 = 0, \tag{5}$$

where $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbb{R}^d$, and $b_1, b_2 \in \mathbb{R}$.

The TSVM constructs two primal problems:

$$\begin{aligned} \min_{\boldsymbol{w}_1,b_1,\xi} \quad & \frac{1}{2}\|(\boldsymbol{A}\boldsymbol{w}_1 + \boldsymbol{e}_1 b_1)\|_2^2 + c_1\boldsymbol{e}_2^T\boldsymbol{\xi} \\ s.t. \quad & -(\boldsymbol{B}\boldsymbol{w}_1 + \boldsymbol{e}_2 b_1) + \boldsymbol{\xi} \geq \boldsymbol{e}_2 \\ & \boldsymbol{\xi} \geq 0, \end{aligned} \tag{6}$$

and

$$\begin{aligned} \min_{\boldsymbol{w}_2,b_2,\eta} \quad & \frac{1}{2}\|(\boldsymbol{B}\boldsymbol{w}_2 + \boldsymbol{e}_2 b_2)\|_2^2 + c_2\boldsymbol{e}_1^T\boldsymbol{\eta} \\ s.t. \quad & (\boldsymbol{A}\boldsymbol{w}_2 + \boldsymbol{e}_1 b_2) + \boldsymbol{\eta} \geq \boldsymbol{e}_1 \\ & \boldsymbol{\eta} \geq 0, \end{aligned} \tag{7}$$

Please cite this article as: Y. Wan et al., Twin extreme learning machines for pattern classification, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.04.036

where $c_1$, $c_2 > 0$ are trade-off parameters. The elements of $\boldsymbol{e}_1 \in \mathbb{R}^{m_1}$ and $\boldsymbol{e}_2 \in \mathbb{R}^{m_2}$ are all one.

Based on the Karush–Kuhn–Tucker (KKT) theorem [19], to train the above TSVM is equivalent to solve the following two QPPs:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \boldsymbol{e}_2^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{G}(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{G}^T\boldsymbol{\alpha} \\ s.t. \quad & 0 \le \alpha_i \le c_1, \end{aligned} \tag{8}$$

and

$$\begin{aligned} \max_{\boldsymbol{\gamma}} \quad & \boldsymbol{e}_1^T\boldsymbol{\gamma} - \frac{1}{2}\boldsymbol{\gamma}^T\boldsymbol{H}(\boldsymbol{G}^T\boldsymbol{G})^{-1}\boldsymbol{H}^T\boldsymbol{\gamma} \\ s.t. \quad & 0 \le \gamma_i \le c_2, \end{aligned} \tag{9}$$

where $\boldsymbol{G} = [\boldsymbol{B}\ \boldsymbol{e}_2]$ and $\boldsymbol{H} = [\boldsymbol{A}\ \boldsymbol{e}_1]$.

The two nonparallel separating hyperplanes are obtained from the solution $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ of (8) and (9) by

$$\begin{aligned} \boldsymbol{u}_1 &= -(\boldsymbol{H}^T\boldsymbol{H} + \epsilon\boldsymbol{I})^{-1}\boldsymbol{G}^T\boldsymbol{\alpha}, \\ \boldsymbol{u}_2 &= -(\boldsymbol{G}^T\boldsymbol{G} + \epsilon\boldsymbol{I})^{-1}\boldsymbol{H}^T\boldsymbol{\gamma}, \end{aligned} \tag{10}$$

where $\boldsymbol{u}_1 = [\boldsymbol{w}_1\ b_1]$ and $\boldsymbol{u}_2 = [\boldsymbol{w}_2\ b_2]$, $\epsilon$ is a positive scalar, $\boldsymbol{I}$ is an identity matrix of $d+1$ dimensions. The nonlinear case is similar to the linear one.

### 2.3. Extreme learning machine

ELM is originally proposed for training SLFNs and then extended to the generalized SLFNs. The output function of ELM for the generalized SLFNs is

$$\boldsymbol{f}(\boldsymbol{x}) = \sum_{i=1}^{L} G(\boldsymbol{w}_i, b_i, \boldsymbol{x})\beta = \boldsymbol{h}(\boldsymbol{x}) \cdot \boldsymbol{\beta}, \tag{11}$$

where $\boldsymbol{\beta}$ is the output weight between the hidden layer and the output layer. $G(\boldsymbol{w}_i, b_i, \boldsymbol{x})$ is the activation function, where $\boldsymbol{w}_i$ is the input weight vector and $b_i$ is the bias of the $i$th hidden node. $\boldsymbol{h}(\boldsymbol{x}) = [G(\boldsymbol{w}_1, b_1, \boldsymbol{x}), \ldots, G(\boldsymbol{w}_L, b_L, \boldsymbol{x})]^T$ is the nonlinear random feature mapping output of the hidden layer with respect to the input pattern $\boldsymbol{x}$.

ELM not only minimizes the norm of output weights, but also requires to minimize the training error.

$$\min \quad \sum_{i=1}^{N} \|\boldsymbol{\beta} \cdot \boldsymbol{h}(\boldsymbol{x}_i) - \boldsymbol{t}_i\| \quad \text{and} \quad \min \quad \|\boldsymbol{\beta}\|. \tag{12}$$

If the training error reaches zero, i.e. $\sum_{i=1}^{N} \|\boldsymbol{\beta} \cdot \boldsymbol{h}(\boldsymbol{x}_i) - \boldsymbol{t}_i\| = 0$, the solution is computed by

$$\hat{\boldsymbol{\beta}} = \boldsymbol{H}^{\dagger}\boldsymbol{T}, \tag{13}$$

where $\boldsymbol{T}$ is the output matrix, $\boldsymbol{H}\dagger$ is the generalized inverse of output matrix $\boldsymbol{H} = [\boldsymbol{h}(\boldsymbol{x}_1) \quad \ldots \quad \boldsymbol{h}(\boldsymbol{x}_N)]^T$.

In order to eliminate possible over-fitting issues, one may wish that the training error is not equal to zero. When $\sum_{i=1}^{N} \|\boldsymbol{\beta} \cdot \boldsymbol{h}(\boldsymbol{x}_i) - \boldsymbol{t}_i\| \neq 0$, the objective function (12) of ELM can be written as

$$\begin{aligned} \min \quad & \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N}\xi_i \\ s.t. \quad & t_i\boldsymbol{\beta} \cdot \boldsymbol{h}(\boldsymbol{x}_i) \ge 1 - \xi_i,\ \xi_i \ge 0,\ i = 1, \ldots, N, \end{aligned} \tag{14}$$

which $\xi_i$ is the training error of the $i$th input pattern.

The decision function of the above ELM is

$$f(\boldsymbol{x}) = \text{sign}\left(\sum_{s=1}^{N_s} \alpha_s t_s K_{\text{ELM}}(\boldsymbol{x}, \boldsymbol{x}_s)\right), \tag{15}$$

where $K_{\text{ELM}}$ is the ELM kernel.

## 3. Proposed twin extreme learning machines

### 3.1. Linear TELM

We firstly consider the case where $G(\boldsymbol{w}_i, b_i, \boldsymbol{x})$ is a linear continuous function.

To simplify notations, matrixes $\boldsymbol{U}$ and $\boldsymbol{V}$ are used to denote the hidden layer output of the data points belonging to class $+1$ and class $-1$, respectively. The hidden layer output matrixes $\boldsymbol{U}$ and $\boldsymbol{V}$ are given by

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{m_1} \end{bmatrix} = \begin{bmatrix} h_1(\boldsymbol{x}_1) & \cdots & h_L(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ h_1(\boldsymbol{x}_{m_1}) & \cdots & h_L(\boldsymbol{x}_{m_1})) \end{bmatrix} \tag{16}$$

and

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1 \\ \vdots \\ \boldsymbol{v}_{m_2} \end{bmatrix} = \begin{bmatrix} h_1(\boldsymbol{x}_1) & \cdots & h_L(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ h_1(\boldsymbol{x}_{m_2}) & \cdots & h_L(\boldsymbol{x}_{m_2})) \end{bmatrix} \tag{17}$$

respectively, where $h_i(\boldsymbol{x}) = G(\boldsymbol{w}_i, b_i, \boldsymbol{x}) = \boldsymbol{w}_i \cdot \boldsymbol{x} + b_i, i = 1, \ldots, L$.

Similar to TSVM, TELM also seeks for two nonparallel separating hyperplanes in the TELM feature space

$$\begin{aligned} &\boldsymbol{f}_1(\boldsymbol{x}) = \boldsymbol{\beta}_1 \cdot \boldsymbol{h}(\boldsymbol{x}) = 0 \text{ and} \\ &\boldsymbol{f}_2(\boldsymbol{x}) = \boldsymbol{\beta}_2 \cdot \boldsymbol{h}(\boldsymbol{x}) = 0. \end{aligned} \tag{18}$$

For each hyperplane, TELM minimizes the distance to one class and requires the distance to the other class to be as large as possible. To achieve these goals, the hyperplane $\boldsymbol{\beta}_1$ minimizes the sum of the squares of distance to the data points $\boldsymbol{u}_i$ $(i = 1, \ldots, m_1)$ and requires the distance to the data points $\boldsymbol{v}_j$ $(j = 1, \ldots, m_2)$ to be larger than 1, while the hyperplane $\boldsymbol{\beta}_2$ minimizes the sum of the squares of distance to the data points $\boldsymbol{v}_j$ and limits the distance to the data points $\boldsymbol{u}_i$ to be larger than 1. This leads to the following pair of primal problems:

$$\begin{aligned} \min_{\boldsymbol{\beta}_1} \quad & \frac{1}{2}\sum_{i=1}^{m_1}(\boldsymbol{u}_i\boldsymbol{\beta}_1)^2 \\ s.t. \quad & -\boldsymbol{v}_j\boldsymbol{\beta}_1 \ge 1,\ j = 1, \ldots, m_2, \end{aligned} \tag{19}$$

and

$$\begin{aligned} \min_{\boldsymbol{\beta}_2} \quad & \frac{1}{2}\sum_{j=1}^{m_2}(\boldsymbol{v}_i\boldsymbol{\beta}_2)^2 \\ s.t. \quad & \boldsymbol{u}_i\boldsymbol{\beta}_2 \ge 1,\ i = 1, \ldots, m_1. \end{aligned} \tag{20}$$

In order to eliminate the possible over-fitting and generalize well on the unseen testing data, we introduce a relaxation term to (19) and (20). The above problems are modified as

$$\begin{aligned} \min_{\boldsymbol{\beta}_1, \boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{U}\boldsymbol{\beta}_1\|_2^2 + c_1\boldsymbol{e}_2^T\boldsymbol{\xi} \\ s.t. \quad & -\boldsymbol{V}\boldsymbol{\beta}_1 + \boldsymbol{\xi} \ge \boldsymbol{e}_2, \boldsymbol{\xi} \ge 0, \end{aligned} \tag{21}$$

and

$$\begin{aligned} \min_{\boldsymbol{\beta}_2, \boldsymbol{\eta}} \quad & \frac{1}{2}\|\boldsymbol{V}\boldsymbol{\beta}_2\|_2^2 + c_2\boldsymbol{e}_1^T\boldsymbol{\eta} \\ s.t. \quad & \boldsymbol{U}\boldsymbol{\beta}_2 + \boldsymbol{\eta} \ge \boldsymbol{e}_1, \boldsymbol{\eta} \ge 0, \end{aligned} \tag{22}$$

where $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ are the error vectors with respect to the training patterns belonging to class $-1$ and class $+1$, respectively. $c_1, c_2 > 0$ are trade-off parameters, and all the elements in vectors $\boldsymbol{e}_1 \in \mathbb{R}^{m_1}$ and $\boldsymbol{e}_2 \in \mathbb{R}^{m_2}$ are one.

Comparing with TSVM, there are three main differences:

1. Obviously, the objective function is different. The objective function of the TSVM need to calculate the bias b, while there are no biases in the objective function of the TELM.

Please cite this article as: Y. Wan et al., Twin extreme learning machines for pattern classification, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.04.036

2. The bias $b$ is not required in the TELM's optimization constraints so that the two nonparallel separating hyperplanes pass through the origin in the TELM feature space.
3. Different from traditional TSVM, the random feature mapping is adopted by TELM and $\boldsymbol{h}(\boldsymbol{x})$ is known to users.

Comparing with traditional ELM, there are three main differences for binary classification problem:

1. TELM aims at generating two nonparallel separating hyperplanes, however, ELM finds only one separable hyperplane.
2. The traditional ELM tends to reach zero training errors, however, in TELM training errors are generally not equal to zero, which leads to more better generation performance on testing data.
3. TELM has two objective functions and needs to solve two QPPs.

The Lagrange function of the primal TELM optimization problems (21) is given by

$$\begin{aligned} L(\boldsymbol{\beta}_1, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\lambda}) =& \frac{1}{2}(\boldsymbol{U}\boldsymbol{\beta}_1)^T(\boldsymbol{U}\boldsymbol{\beta}_1) + c_1\boldsymbol{e}_2^T\boldsymbol{\xi} \\ & -\boldsymbol{\alpha}^T(-\boldsymbol{V}\boldsymbol{\beta}_1 + \boldsymbol{\xi} - \boldsymbol{e}_2) - \boldsymbol{\lambda}^T\boldsymbol{\xi}, \end{aligned} \tag{23}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_{m_2})^T$ and $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_{m_2})^T$ are the vectors of Lagrange multipliers.

The Karush–Kuhn–Kucher (KKT) necessary and sufficient optimality conditions for $\boldsymbol{w}_1$, $\boldsymbol{\xi}$ and $\boldsymbol{\alpha}$, $\boldsymbol{\lambda}$ are given by

$$\nabla_{\beta_1} L = \boldsymbol{U}^T\boldsymbol{U}\boldsymbol{\beta}_1 + \boldsymbol{V}^T\boldsymbol{\alpha} = 0 \tag{24}$$

$$\nabla_{\xi} L = c_1\boldsymbol{e}_2 - \boldsymbol{\alpha} - \boldsymbol{\lambda} = 0 \tag{25}$$

$$-\boldsymbol{V}\boldsymbol{\beta}_1 + \boldsymbol{\xi} \geq \boldsymbol{e}_2, \boldsymbol{\xi} \geq 0 \tag{26}$$

$$\boldsymbol{\alpha}^T(-\boldsymbol{V}\boldsymbol{\beta}_1 + \boldsymbol{\xi} - \boldsymbol{e}_2) = 0, \boldsymbol{\lambda}^T\boldsymbol{\xi} = 0 \tag{27}$$

$$\boldsymbol{\alpha} \geq 0, \boldsymbol{\lambda} \geq 0. \tag{28}$$

since $\boldsymbol{\lambda} \geq 0$, from (25) we have

$$0 \leq \boldsymbol{\alpha} \leq c_1. \tag{29}$$

Obviously, (24) implies that

$$\boldsymbol{\beta}_1 = -(\boldsymbol{U}^T\boldsymbol{U})^{-1}\boldsymbol{V}^T\boldsymbol{\alpha}. \tag{30}$$

Although $\boldsymbol{U}^T\boldsymbol{U}$ is always positive semidefinite, it may be singular in some cases, i.e. the inverse matrix $(\boldsymbol{U}^T\boldsymbol{U})^{-1}$ is not exist. By introducing a regularization term $\epsilon\boldsymbol{I}$, we can avoid the possible ill-conditioning of $\boldsymbol{U}^T\boldsymbol{U}$. Here, $\epsilon$ is a randomly small positive scalar and $\boldsymbol{I}$ is an identity matrix of appropriate dimensions. So (30) can be modified as

$$\boldsymbol{\beta}_1 = -(\boldsymbol{U}^T\boldsymbol{U} + \epsilon\boldsymbol{I})^{-1}\boldsymbol{V}^T\boldsymbol{\alpha}. \tag{31}$$

Then substituting (31) into (40) and using (24)–(28), we obtain the dual problem of (21) as follows:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \boldsymbol{e}_2^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{V}(\boldsymbol{U}^T\boldsymbol{U} + \epsilon\boldsymbol{I})^{-1}\boldsymbol{V}^T\boldsymbol{\alpha} \\ s.t. \quad & 0 \leq \alpha_i \leq c_1, i = 1, \ldots, m_2. \end{aligned} \tag{32}$$

Similarly, we can obtain the dual of (22) as

$$\begin{aligned} \max_{\boldsymbol{\gamma}} \quad & \boldsymbol{e}_1^T\boldsymbol{\gamma} - \frac{1}{2}\boldsymbol{\gamma}^T\boldsymbol{U}(\boldsymbol{V}^T\boldsymbol{V} + \epsilon\boldsymbol{I})^{-1}\boldsymbol{U}^T\boldsymbol{\gamma} \\ s.t. \quad & 0 \leq \gamma_i \leq c_2, i = 1, \ldots, m_1, \end{aligned} \tag{33}$$

where $\boldsymbol{\gamma}$ is the vector of Lagrange multiplier. The vector $\boldsymbol{\beta}_2$ is given by

$$\boldsymbol{\beta}_2 = -(\boldsymbol{V}^T\boldsymbol{V} + \epsilon\boldsymbol{I})^{-1}\boldsymbol{U}^T\boldsymbol{\gamma}. \tag{34}$$

Once the vectors $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are computed, the separating hyperplanes

$$\boldsymbol{f}_1(\boldsymbol{x}) = \boldsymbol{\beta}_1 \cdot \boldsymbol{h}(\boldsymbol{x}) = 0 \quad \text{and} \quad \boldsymbol{f}_2(\boldsymbol{x}) = \boldsymbol{\beta}_2 \cdot \boldsymbol{h}(\boldsymbol{x}) = 0 \tag{35}$$

are obtained. A new data point $\boldsymbol{x} \in \mathbb{R}^d$ belongs to the class $+1$ or class $-1$, depending on its distance to these two hyperplanes, i.e.

$$\boldsymbol{f}(\boldsymbol{x}) = \arg\min_{r=1,2} d_r(\boldsymbol{x}) = \arg\min_{r=1,2} |\boldsymbol{\beta}_r^T h(\boldsymbol{x})|, \tag{36}$$

where $|\cdot|$ is the perpendicular distance of data point $\boldsymbol{x}$ from the hyperplane $\boldsymbol{\beta}_r$.

### 3.2. Kernel TELM

To extend our algorithm to the nonlinear case, we consider the following two kernel-generated separating surfaces:

$$\begin{aligned} & K_{\mathrm{ELM}}(\boldsymbol{x}^T, \boldsymbol{C}^T)\boldsymbol{\mu}_1 = 0, \\ & K_{\mathrm{ELM}}(\boldsymbol{x}^T, \boldsymbol{C}^T)\boldsymbol{\mu}_2 = 0, \end{aligned} \tag{37}$$

where $\boldsymbol{C}^T = [\boldsymbol{A} \quad \boldsymbol{B}]^T$ and $K_{\mathrm{ELM}}$ is an ELM kernel function. We can calculate the ELM kernel function as

$$\begin{aligned} K_{\mathrm{ELM}}(\boldsymbol{x}_j, \boldsymbol{x}_i) &= \boldsymbol{h}(\boldsymbol{x}_j) \cdot \boldsymbol{h}(\boldsymbol{x}_i) \\ &= [G(\boldsymbol{w}_1, b_1, \boldsymbol{x}_j), \ldots, G(\boldsymbol{w}_L, b_L, \boldsymbol{x}_j)]^T \\ &\cdot [G(\boldsymbol{w}_1, b_1, \boldsymbol{x}_i), \ldots, G(\boldsymbol{w}_L, b_L, \boldsymbol{x}_i)]. \end{aligned} \tag{38}$$

For the surface $K_{\mathrm{ELM}}(\boldsymbol{x}^T, \boldsymbol{C}^T)\boldsymbol{\mu}_1 = 0$, the primal problem is given by

$$\begin{aligned} \min_{\boldsymbol{\mu}_1, \boldsymbol{\xi}} \quad & \frac{1}{2}\|K_{\mathrm{ELM}}(\boldsymbol{A}, \boldsymbol{C}^T)\boldsymbol{\mu}_1\|_2^2 + c_1\boldsymbol{e}_2^T\boldsymbol{\xi} \\ s.t. \quad & -K_{\mathrm{ELM}}(\boldsymbol{B}, \boldsymbol{C}^T)\boldsymbol{\mu}_1 + \boldsymbol{\xi} \geq \boldsymbol{e}_2, \boldsymbol{\xi} \geq 0, \end{aligned} \tag{39}$$

where $c_1 > 0$ is a parameter. Its Lagrangian function is

$$\begin{aligned} L(\boldsymbol{\mu}_1, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\lambda}) =& \frac{1}{2}\|K_{\mathrm{ELM}}(\boldsymbol{A}, \boldsymbol{C}^T)\boldsymbol{\mu}_1\|_2^2 + c_1\boldsymbol{e}_2^T\boldsymbol{\xi} && (40) \\ & -\boldsymbol{\alpha}^T(-K_{\mathrm{ELM}}(\boldsymbol{B}, \boldsymbol{C}^T)\boldsymbol{\mu}_1 + \boldsymbol{\xi} - \boldsymbol{e}_2) && (40) \\ & -\boldsymbol{\lambda}^T\boldsymbol{\xi}, && (40) \end{aligned}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_{m_2})^T$ and $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_{m_2})^T$ are the vectors of Lagrange multipliers.

The KKT conditions for $\boldsymbol{w}_1$, $\boldsymbol{\xi}$ and $\boldsymbol{\alpha}$, $\boldsymbol{\lambda}$ are given by

$$\begin{aligned} \nabla_{\mu_1} L =& K_{\mathrm{ELM}}(\boldsymbol{A}, \boldsymbol{C}^T)^T K_{\mathrm{ELM}}(\boldsymbol{A}, \boldsymbol{C}^T)\boldsymbol{\mu}_1 \\ & + K_{\mathrm{ELM}}(\boldsymbol{B}, \boldsymbol{C}^T)^T\boldsymbol{\alpha} = 0 \end{aligned} \tag{41}$$

$$\nabla_{\xi} L = c_1\boldsymbol{e}_2 - \boldsymbol{\alpha} - \boldsymbol{\lambda} = 0 \tag{42}$$

$$-K_{\mathrm{ELM}}(\boldsymbol{B}, \boldsymbol{C}^T)\boldsymbol{\mu}_1 + \boldsymbol{\xi} \geq \boldsymbol{e}_2, \boldsymbol{\xi} \geq 0 \tag{43}$$

$$\begin{aligned} & \boldsymbol{\alpha}^T(-K_{\mathrm{ELM}}(\boldsymbol{B}, \boldsymbol{C}^T)\boldsymbol{\mu}_1 + \boldsymbol{\xi} - \boldsymbol{e}_2) = 0, \\ & \boldsymbol{\lambda}^T\boldsymbol{\xi} = 0 \end{aligned} \tag{44}$$

$$\boldsymbol{\alpha} \geq 0, \boldsymbol{\lambda} \geq 0. \tag{45}$$

Since $\boldsymbol{\lambda} \geq 0$, from (42) we have

$$0 \leq \boldsymbol{\alpha} \leq c_1. \tag{46}$$

Please cite this article as: Y. Wan et al., Twin extreme learning machines for pattern classification, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.04.036

Let $\boldsymbol{R} = K_{\mathrm{ELM}}(\boldsymbol{A}, \boldsymbol{C}^T)$ and $\boldsymbol{S} = K_{\mathrm{ELM}}(\boldsymbol{B}, \boldsymbol{C}^T)$, (41) can be written as

$$\boldsymbol{R}^T\boldsymbol{R}\boldsymbol{\mu}_1 + \boldsymbol{S}\boldsymbol{\alpha} = 0, \quad \text{i.e.,} \quad \boldsymbol{\mu}_1 = -(\boldsymbol{R}^T\boldsymbol{R})^{-1}\boldsymbol{S}\boldsymbol{\alpha}. \tag{47}$$

Similarly, (47) can be modified as

$$\boldsymbol{\mu}_1 = -(\boldsymbol{R}^T\boldsymbol{R} + \epsilon\boldsymbol{I})^{-1}\boldsymbol{S}\boldsymbol{\alpha}. \tag{48}$$

The dual problem of (39) is given by

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \boldsymbol{e}_2^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{S}(\boldsymbol{R}^T\boldsymbol{R} + \epsilon\boldsymbol{I})^{-1}\boldsymbol{S}^T\boldsymbol{\alpha} \\ s.t. \quad & 0 \le \alpha_i \le c_1, i = 1, \ldots, m_2. \end{aligned} \tag{49}$$

In a similar manner, for the surface $K_{\mathrm{ELM}}(\boldsymbol{x}^T, \boldsymbol{C}^T)\boldsymbol{\mu}_2 = 0$, the primal problem is given by

$$\begin{aligned} \min_{\boldsymbol{\mu}_2, \boldsymbol{\eta}} \quad & \frac{1}{2}\|K_{\mathrm{ELM}}(\boldsymbol{B}, \boldsymbol{C}^T)\boldsymbol{\mu}_2\|_2^2 + c_2\boldsymbol{e}_1^T\boldsymbol{\eta} \\ s.t. \quad & -K_{\mathrm{ELM}}(\boldsymbol{A}, \boldsymbol{C}^T)\boldsymbol{\mu}_2 + \boldsymbol{\eta} \ge \boldsymbol{e}_1, \boldsymbol{\eta} \ge 0, \end{aligned} \tag{50}$$

where $c_2 > 0$ is a parameter. The dual problem of (50) is given by

$$\begin{aligned} \max_{\boldsymbol{\gamma}} \quad & \boldsymbol{e}_1^T\boldsymbol{\gamma} - \frac{1}{2}\boldsymbol{\gamma}^T\boldsymbol{R}(\boldsymbol{S}^T\boldsymbol{S} + \epsilon\boldsymbol{I})^{-1}\boldsymbol{R}^T\boldsymbol{\gamma} \\ s.t. \quad & 0 \le \gamma_i \le c_2, i = 1, 2, \ldots, m_1 \end{aligned} \tag{51}$$

with the output weight vector

$$\boldsymbol{\mu}_2 = -(\boldsymbol{S}^T\boldsymbol{S} + \epsilon\boldsymbol{I})^{-1}\boldsymbol{R}^T\boldsymbol{\gamma}. \tag{52}$$

Once the solutions $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ of problems (39) and (50) are obtained, a new data point $\boldsymbol{x}$ is assigned to the class $c$ ($c = +1, -1$) in a manner similar to the linear case.

## 4. Complete TELM algorithm

We can summarize our TELM algorithm as follows:

*Algorithm TELM:*Given a training set $\boldsymbol{X} = \{(\boldsymbol{x}_i, t_i) | \boldsymbol{x}_i \in \mathbb{R}^d, t_i = \{+1, -1\}, i = 1, \ldots N\}$, activation function $G(\boldsymbol{x})$, and the number of hidden node number $L$.

*Step 1:* Initiate an ELM network with $L$ hidden nodes using the random input weight $\boldsymbol{w}_i$ and bias $b_i$.

*Step 2:* Construct input matrixes $\boldsymbol{A}$ and $\boldsymbol{B}$. For linear TELM, then calculate their hidden layer output matrixes $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively; for nonlinear TELM, calculate matrixes $\boldsymbol{R}$ and $\boldsymbol{S}$, respectively.

*Step 3:* For linear TELM, construct convex QPPs (32) and (33); for nonlinear TELM, construct convex QPPs (49) and (51).

*Step 4:* Obtain Lagrange multipliers $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ by solving two QPPs.

*Step 5:* For linear TELM, calculate the output weights $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ using (31) and (34); for nonlinear TELM, calculate the output weights $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ using (48) and (52);

*Step 6:* Calculate the perpendicular distance of data point $\boldsymbol{x}$ from the separating hyperplane using (36), then assign the $\boldsymbol{x}$ to class $i$ ($i = +1, -1$).

**Remark 1.** In principle, any infinitely differentiable function can used as the activation function $G(\boldsymbol{x})$ in the TELM algorithm, such as the radial basis, sigmoid, and many nonregular functions.

**Remark 2.** In our TELM, there are four strictly convex QPPs to be solved: (32), (33), (49) and (51). These problems can be rewritten in the following unified form:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{Q}\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha} \\ s.t. \quad & 0 \le \alpha_i \le c \quad i = 1, \ldots, m \end{aligned} \tag{53}$$

where $\boldsymbol{Q} \in \mathbb{R}^{m \times m}$ is positive definite. For example, for the problem (32), $\boldsymbol{Q} = \boldsymbol{V}(\boldsymbol{U}^T\boldsymbol{U} + \epsilon\boldsymbol{I})^{-1}\boldsymbol{V}^T$.

Several methods can be used to solve the convex quadratic problems. These methods may include but are not limited the Brazilai–Borwein method [20], the Nesetrov gradient method [21], and trust region method [22]. In the paper, we introduce the successive overrelaxation (SOR) technique [23,24], which can solve efficiently quadratic problem.

*SOR technique:*Choose $t \in (0, 2)$. Start with any $\boldsymbol{\alpha}^0 \in \mathbb{R}^n$. Having $\boldsymbol{\alpha}^i$, compute $\boldsymbol{\alpha}^{i+1}$ as follows:

$$\boldsymbol{\alpha}^{i+1} = (\boldsymbol{\alpha}^i - t\boldsymbol{E}^{-1}(\boldsymbol{Q}\boldsymbol{\alpha}^i - \boldsymbol{e} + \boldsymbol{L}(\boldsymbol{\alpha}^{i+1} - \boldsymbol{\alpha}^i))) \tag{54}$$

until $\|\boldsymbol{\alpha}^{i+1} - \boldsymbol{\alpha}^i\|$ is less than some prescribed tolerance, where $\boldsymbol{e}$ is vector of ones of appropriate dimensions. $\boldsymbol{L} \in \mathbb{R}^{m \times m}$ is the strictly lower triangular matrix, where $l_{ij} = q_{ij}, i > j$. $\boldsymbol{E} \in \mathbb{R}^{m \times m}$ is the diagonal matrix, where $e_{ij} = q_{ij}, i = j$.

SOR technique is an efficient and simple QPP solver. SOR does not need to store the data set in memory during the calculation, so it can effectively handle large data sets. Due to the properties of the matrix $\boldsymbol{E}$ and $\boldsymbol{L}$, the computation of the Lagrange multipliers $\boldsymbol{\alpha}$ is greatly simplified. So the speed of solving QPPs is sped up. Furthermore, SOR technique has been shown to converge linearly to a solution [23]. The experiment in Section 6 shows that SOR technique has a significant acceleration effect on TELM algorithm.

## 5. Discussions

### 5.1. Difference of separating hyperplane

For binary classification problem, we analyze the difference of separating hyperplane between SVM, TSVM, ELM and TELM algorithms.

SVM finds a separating hyperplane by maximizing the width of two parallel support hyperplanes while minimizing the training errors.

The traditional ELM finds a separating hyperplane by jointly minimizing the norm of output weights and reaching zero training error. The separating hyperplane passes through the origin in the ELM feature space.

TSVM generates two nonparallel separating hyperplanes so that each hyperplane tends to reach the smallest distance to one of the two classes and tries to be far away from the other class. The bias $b$ is required in the primal objective functions (6) and (7) of TSVM, so that the two hyperplanes in TSVM do not basically pass through the origin in the TSVM feature space.

Similar to TSVM, TELM finds two nonparallel separating hyperplanes in the TELM feature space. However, the two hyperplanes pass through the origin in the TELM feature space. The main reason is that the bias $b$ is not required in the primal objective functions (19) and (20) of TELM.

The difference and relationship among TELM, ELM, TSVM, and SVM can be summarized in Table 1.

### 5.2. Computational complexity

For SVM, the computational time is mainly spent on obtaining the Lagrange multipliers $\boldsymbol{\alpha}$ by solving QPP (3). To obtain $\boldsymbol{\beta}$, the computational time of ELM is mainly spent on calculating the generalized inverse. In practice, we can calculate $\boldsymbol{\beta}$ depends on the magnitude of the number of hidden nodes and the number of training samples:

$$\boldsymbol{\beta} = \begin{cases} (\frac{\boldsymbol{I}}{C} + \boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{T} & \text{if } N < L \\ \boldsymbol{H}^T(\frac{\boldsymbol{I}}{C} + \boldsymbol{H}\boldsymbol{H}^T)^{-1}\boldsymbol{T} & \text{if } N \geqslant L. \end{cases} \tag{55}$$

Using (55) appropriately can reduce the computational cost dramatically.

For TSVM and TELM, the main computational time is spend on solving the quadratic programming problems. However, there are

**Table 1**
Feature comparisons among TELM, ELM, TSVM, and SVM.

| | TELM | ELM | TSVM | SVM |
|---|---|---|---|---|
| Feature mapping | Random feature mapping or ELM kernels can be used | Random feature mapping or ELM kernels can be used | Using SVM kernels | Using SVM kernels |
| Separating hyperplane | Two nonparallel separating hyperplanes which pass through the origin | A separating hyperplane | Two nonparallel separating hyperplanes | A separating hyperplane |
| Support vectors based | No | No | No | Yes (support vectors $\boldsymbol{x}_i$ corresponding to $\alpha_i = 0$) |
| Multi classes | $k$ or $k(k-1)/2$ binary TELM for $k$ classes | Single ELM for $k$ classes | $k$ or $k(k-1)/2$ binary TSVM for $k$ classes | $k$ or $k(k-1)/2$ binary SVM for $k$ classes |
| Optimization task | Two smaller sized QPPs | Moore–Penrose generalized inverse | Two smaller sized QPPs | One large QPP |
| Output function | $\boldsymbol{f}(\boldsymbol{x}) = \text{argmin}_{r=1,2}\lvert\boldsymbol{\beta}_r^T\boldsymbol{h}(\boldsymbol{x})\rvert$ | $\boldsymbol{f}(\boldsymbol{x}) = h(\boldsymbol{x})(\frac{\boldsymbol{I}}{C} + \boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{T}$ or $\boldsymbol{f}(\boldsymbol{x}) = h(\boldsymbol{x})\boldsymbol{H}^T(\frac{\boldsymbol{I}}{C} + \boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{T}$ | $\boldsymbol{f}(\boldsymbol{x}) = \text{argmin}_{r=1,2}\lvert\boldsymbol{w}_r^T\boldsymbol{x} + b_r\rvert$ | $\boldsymbol{f}(\boldsymbol{x}) = \text{sign}(\sum_{i=1}^{N}\alpha_i t_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b)$ |

**Table 2**
Specification of binary classification problems.

| Datasets | # Train | # Test | # Features | Random perm |
|---|---|---|---|---|
| Australian | 552 | 138 | 15 | Yes |
| Liver | 276 | 69 | 7 | Yes |
| Colon | 50 | 12 | 2000 | Yes |
| Heart | 216 | 54 | 14 | Yes |
| Ionosphere | 281 | 70 | 34 | Yes |
| Leukemia | 38 | 34 | 7129 | No |
| Monk1 | 124 | 432 | 7 | No |
| Monk2 | 169 | 432 | 7 | No |
| Mushroom | 1500 | 6624 | 22 | Yes |
| Pima | 614 | 154 | 9 | Yes |
| Sonar | 166 | 42 | 61 | Yes |
| Adult | 4781 | 27,780 | 123 | No |

**Table 3**
Specification of multi-class classification problems.

| Datasets | # Train | # Test | # Features | # Classes | Random perm |
|---|---|---|---|---|---|
| Wine | 142 | 36 | 13 | 3 | Yes |
| Vowel | 528 | 462 | 10 | 11 | No |
| Segment | 1848 | 462 | 19 | 7 | Yes |
| Satimage | 4435 | 2000 | 36 | 6 | No |
| Vehicle | 677 | 169 | 18 | 4 | Yes |
| DNA | 2000 | 1186 | 180 | 3 | No |
| Pendigit | 7494 | 3498 | 16 | 10 | No |
| USPS | 7291 | 2007 | 256 | 10 | No |
| MNISTA | 1600 | 400 | 256 | 10 | Yes |

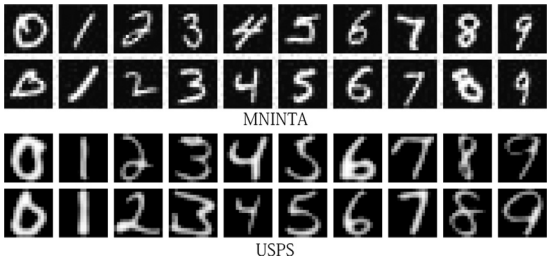



**Fig. 1.** Example images extracted from the USPS and MNISTA data sets.

some differences: (1) TELM does not compute the bias $b$; (2) more importantly, their matrixes are different in the process of solving QPPs. Linear TSVM obtains Lagrange multipliers $\boldsymbol{\alpha}$ by solving the QPPs (8) and (9), where $\boldsymbol{G}^T\boldsymbol{G}$ (size: $(d+1)\times(d+1)$, where $d$ is the number of sample feature) and $\boldsymbol{H}^T\boldsymbol{H}$ (size: $(d+1)\times(d+1)$) is used. TELM obtains Lagrange multipliers $\boldsymbol{\alpha}$ by solving the QPPs (32) and (33), where $\boldsymbol{U}^T\boldsymbol{U}$ (size: $L\times L$) and $\boldsymbol{V}^T\boldsymbol{V}$ (size: $L\times L$) is used. When the number of sample feature is relatively large, i.e. $L \ll d$, the computational time can be reduced dramatically. In addition, the computational cost of generating two separating hyperplanes is negligible because the inverse, such as $(\boldsymbol{U}^T\boldsymbol{U} + \epsilon\boldsymbol{I})^{-1}$, is calculated in the process of solving the QPPs.

## 6. Experiment and results

In this section, we demonstrate the effectiveness of the proposed method on real-world binary and multi-class classification data sets. To verify the capabilities of our algorithm, comparisons are made with different algorithms such as ELM, SVM and TSVM.

### 6.1. Data sets description

In this subsection, we will introduce the data sets we utilized in our experiments in detail, which includes 10 binary classification cases and 9 multi-class classification cases. Most of the data sets are from UCI machine Learning Repository [25].

#### 6.1.1. Binary class datasets

The binary class data sets contain seven UCI data sets [25], two genes data sets (Leukemia data set [26] and Colon Microarray data set [27]), and one bio-medical data set (Heart data set [28]). We have summarized the characteristics of these binary data sets in Table 2.

Column "random perm" in Table 2 indicates whether the input data are reshuffled during each experiment. If the input data remain fixed in each trial, it is marked as "No". Otherwise, it is marked as "Yes". These reshuffled data sets are randomly permuted and taken without replacement, where eighty percent of the samples are used as training data and the rest are used as testing data. The same definition and strategy are also applied for Table 3.

#### 6.1.2. Multi-class data sets

We also test TELM algorithm on nine multi-class data sets that have been widely used for evaluating classification algorithms. Table 3 summarizes the characteristics of these multi-class data sets.

USPS and MNIST [26] data sets are well-known handwritten digit data sets with samples from '0' to '9', which are shown in Fig. 1. MNIST data set has 60,000 training samples and 10,000 test data. The MNISTA data set is a subset of MNIST by randomly selecting 2000 images. All the images in USPS and MNISTA are rescaled to size $16 \times 16$.

### 6.2. Experimental setting

We implement TELM and all the baselines by using matlab R2013a and run all the algorithms on the same x 86-64 windows

Please cite this article as: Y. Wan et al., Twin extreme learning machines for pattern classification, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.04.036

**Table 4**
Performance comparison of TELM, ELM, TSVM and SVM with linear kernel.

| Data sets | TELM | | ELM | | TSVM | | SVM | |
|---|---|---|---|---|---|---|---|---|
| | Accuracy | Times (s) | Accuracy | Times (s) | Accuracy | Times (s) | Accuracy | Times (s) |
| Australian | **87.32**± 2.70 | 0.1425 | 87.17 ± 2.94 | 0.0056 | 86.88± 3.18 | 0.1189 | 86.16± 3.06 | 30.4092 |
| Liver | **72.64**± 5.52 | 0.0309 | 71.45± 3.42 | 0.0011 | 67.97± 4.35 | 0.0120 | 69.42± 3.44 | 0.2307 |
| Colon | 62.67± 7.08 | 0.0005 | 54.17± 9.82 | 0.0005 | 52.08± 7.20 | 0.3179 | **85.83**± 6.86 | 0.0078 |
| Heart | **87.96**± 4.21 | 0.0528 | 87.59± 4.79 | 0.0007 | 85.28± 2.74 | 0.0103 | 84.69± 3.85 | 0.8659 |
| Ionosphere | **91.57**± 3.72 | 0.0331 | 87.29± 4.28 | 0.0010 | 90.07± 1.76 | 0.0159 | 89.57± 2.13 | 0.0956 |
| Leukemia | 58.82 | 0.0009 | 64.71 | 0.0019 | 52.35 | 1.5244 | **82.35** | 0.0209 |
| Mushroom | **94.66**± 0.49 | 124.9509 | 94.21± 0.58 | 2.8816 | 94.57± 0.19 | 85.4989 | 94.53± 0.88 | 157.8353 |
| Pima | **79.22**± 1.06 | 0.1024 | 71.90± 2.87 | 0.0063 | 77.20± 1.96 | 0.0349 | 78.49± 3.03 | 0.0506 |
| Sonar | 79.92± 6.11 | 0.0147 | **81.22**± 5.52 | 0.0005 | 77.59± 2.29 | 0.0074 | 79.95± 6.51 | 0.0023 |
| Adult | **84.50** | 15.9780 | 84.34 | 1.3248 | 84.39 | 12.6939 | 84.47 | 58.7185 |

machine with 8 GB of memory and Core 2 Quad, 2.50-GHz CPU. The codes used for SVM, TSVM, and ELM are downloaded from [29,30] and [31], respectively. From the code of TSVM algorithm, we find that the SOR technique is also used to solve the quadratic problems in the TSVM algorithm.

In order to test the performance of TELM, we test the TELM algorithm with different kernel, such as linear kernel, Gaussian kernel and Sigmoid kernel. Here, we give the range of values for different parameters in the experiment. For TELM and TSVM, we treat parameter $C = 0.5$ as a fixed value for all the data sets. However, for SVM parameter $C$ can take 10 different values, which are $\{2^{-18}, 2^{-14}, \ldots, 2^{14}, 2^{18}\}$. For TELM and ELM, we need to adjust the number of hidden layer nodes, i.e. parameter $L$. The different values of parameter $L$ are {50, 100, 200, 500, 1000}. When the Gaussian function $L(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\kappa\|\boldsymbol{x} - \boldsymbol{y}\|^2)$ is used as the kernel function, parameter $\kappa$ need to be reasonably chosen. The value of $\kappa$ is the same as parameter $C$.

To test the sensitivity of the algorithm to the parameters, we have conduct extensively parameter sensitivity analysis by two experiments. To be specific, the first experiment is to analyze the sensitivity of TELM and SVM with Gaussian kernel to parameters $C$ and $\kappa$. Another experiment is to analyze the sensitivity of the TELM with the Sigmoid kernel to the parameters $C$ and $L$. In order to make the two experiments more reasonable, the range of parameters is larger than the above. The parameter $C$ and parameter $\kappa$ can take 37 different values, which are $\{2^{-18}, 2^{-17}, \ldots, 2^{17}, 2^{18}\}$. Parameter $L$ can take 19 different values, which are $\{100, 200, \ldots, 1800, 1900\}$.

By using one-against-all (OAA) or one-against-one (OAO) methods, TELM can also be applied to multi-class classification problem. In the experiment, OAO method has been used.

For each data set, the experiments are repeated ten times with different parameter values, and the average results are reported. The experimental results include prediction accuracy, corresponding standard deviation and training time.

### 6.3. Experimental results

In this subsection, we will present the results of TELM and other baselines in terms of prediction accuracy and computational efficiency.

#### 6.3.1. Results of binary data sets

We first experiment on the binary data sets with linear and Gaussian kernel functions. The experiment results are shown in Tables 4 and 5, in which the best results are marked in bold.

From Tables 4 and 5, we can see that TELM algorithm achieves much better performance on most tasks than ELM, TSVM, and SVM algorithms. When using linear kernel function, TELM is better than ELM in eight data sets and TSVM is better than SVM on five data sets and when the kernel function is Gaussian, TELM is better than ELM in seven tasks, and TSVM is better than SVM in six tasks. Comparing Tables 4 and 5, TELM with Gaussian kernel achieves better performance on seven data sets than TELM with linear kernel.

In order to evaluate the computational efficiency of the proposed algorithm, the training time of TELM, ELM, TSVM and SVM are also listed in Tables 4 and 5. It is clear that the ELM is the most efficient algorithm, and the training time of TELM, TSVM and SVM algorithms are of the same order of magnitude. This phenomenon is consistent with theoretical analysis. TELM, TSVM and SVM need to solve QPPs, but ELM only need to solve a least square problem.

However, it is noteworthy that on those data sets with relatively large number of sample features, such as Leukemia and Colon Microarray, TELM is several times faster than the other three algorithms. This result is consistent with the analysis given in Section 5. TELM shows a more obvious advantage when the number of sample features of the data set are relatively large or quite large.

#### 6.3.2. Results of multi-class data sets

We use the algorithms with Gaussian kernel to do experiments on multi-class data sets, and the results are presented in Table 6. From Table 6, it is obvious that TELM outperforms ELM, TSVM, and SVM consistently on eight out of nine data sets, which demonstrates that the TELM also has powerful classification ability for multi-class classification problem. In terms of classification accuracy, TELM is better than ELM on 8 data sets, and TSVM is better than SVM on seven data sets.

The training time are also listed in Table 6. ELM algorithm is the fastest on most data sets (7 out of 9), while TSVM is the most time-consuming algorithm. TELM algorithm gets the fastest speed for one data set. Though TELM is not as fast as ELM, its training is still quite efficient.

### 6.4. Experiment analysis

In this subsection, we first analyze the experimental results. Then, we will do several analytical experiments to further verify the effectiveness of TELM.

#### 6.4.1. Results analysis

From the experiment results comparison, we have the following three observations:

(1) The classifier with two nonparallel separating hyperplanes achieves better performance than classifier with only one separating hyperplane on most data sets. This demonstrates that the classifier with two nonparallel separating hyperplanes can predict more correctly in most classification cases.

Please cite this article as: Y. Wan et al., Twin extreme learning machines for pattern classification, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.04.036

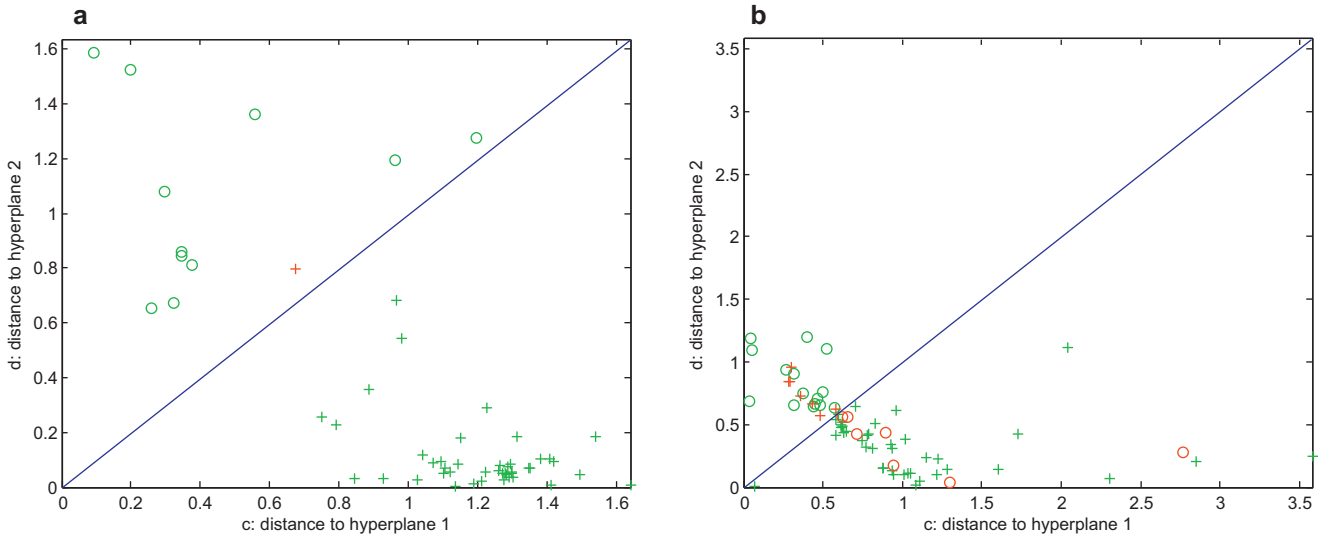



**Fig. 2.** Two-dimensional projection for (a) Ionosphere and (b) Bupa Liver Disorders. If the value of $c_i$ is smaller than $d_i$, the data point $\boldsymbol{x}_i$ is assigned to class $+1$, and vice versa. If the data point belongs to class $+1$, it is drawn as "o". If the data point belongs to class $-1$, it is drawn as "+", and all the misclassified points are marked in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

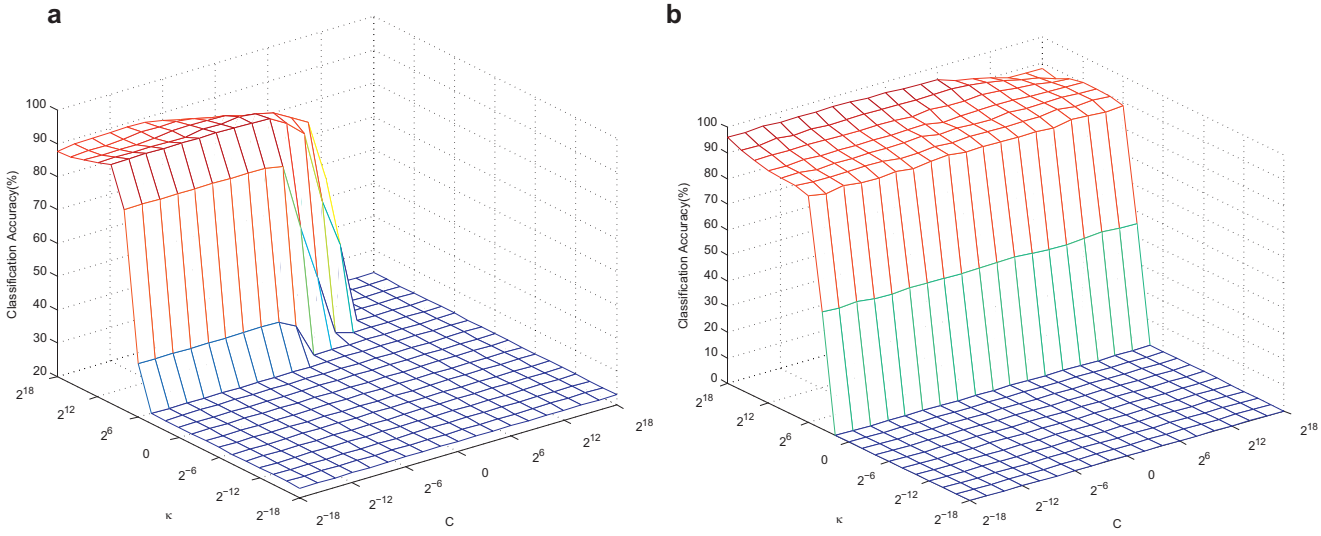



**Fig. 3.** The sensitivity analysis of (a) SVM and (b) TELM with Gaussian kernel to parameters $C$ and $\kappa$ on Satimage data set.

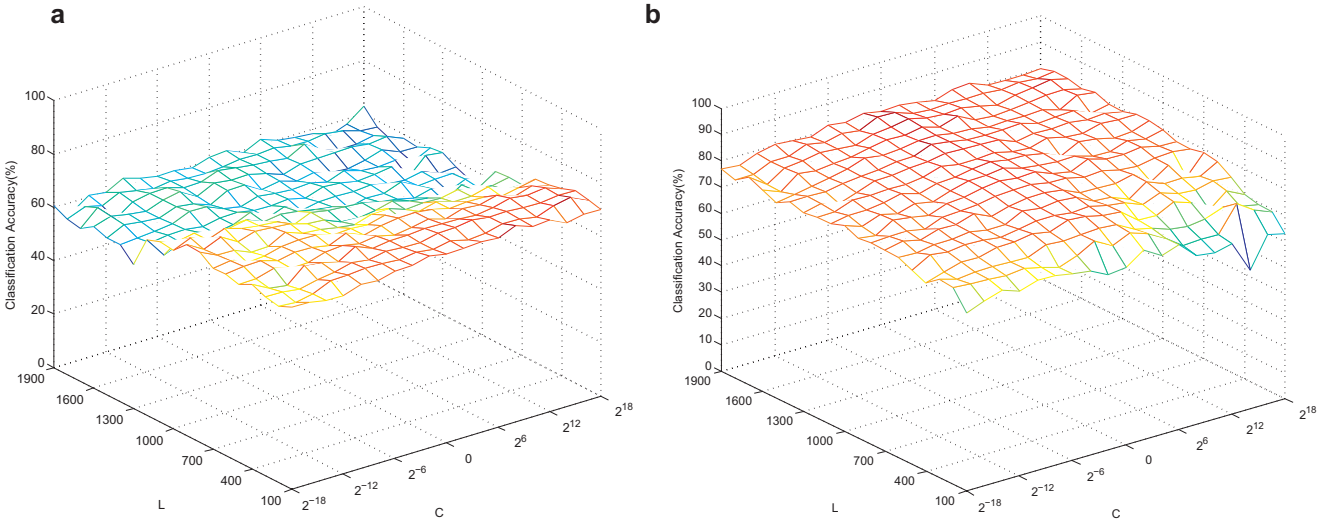



**Fig. 4.** The sensitivity analysis of TELM with Sigmoid kernel to parameters $C$ and $L$ on: (a) Australian and (b) Satimage data sets.

Please cite this article as: Y. Wan et al., Twin extreme learning machines for pattern classification, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.04.036

**Table 5**
Performance comparison of TELM, ELM, TSVM and SVM with Gaussian kernel.

| Data sets | TELM | | ELM | | TSVM | | SVM | |
|---|---|---|---|---|---|---|---|---|
| | Accuracy | Times (s) | Accuracy | Times (s) | Accuracy | Times (s) | Accuracy | Times (s) |
| Australian | **75.14**± 2.96 | 0.0723 | 71.45± 2.90 | 0.0118 | 74.43± 5.36 | 0.0759 | 73.91± 5.33 | 0.1378 |
| Liver | **76.96**± 5.61 | 0.0154 | 73.48± 5.38 | 0.0115 | 75.51± 4.66 | 0.0154 | 72.17± 6.14 | 0.0031 |
| Colon | 82.92± 5.36 | 0.0004 | **89.17**± 7.91 | 0.0010 | 88.75± 7.36 | 0.0025 | 86.67± 8.96 | 0.0009 |
| Heart | **83.15**± 4.15 | 0.0075 | 71.67± 6.71 | 0.0010 | 76.76± 3.77 | 0.0093 | 80.19± 3.71 | 0.0117 |
| Ionosphere | **95.71**± 2.36 | 0.0091 | 88.86± 2.68 | 0.0016 | 93.71± 1.68 | 0.0125 | 94.43± 2.47 | 0.0035 |
| Leukemia | **94.12** | 0.0006 | 76.47 | 0.0084 | 85.29 | 0.0068 | 70.59 | 0.0291 |
| Mushroom | **99.81**± 0.10 | 40.5867 | 87.04± 0.73 | 3.6437 | 96.67± 0.24 | 45.7031 | 98.87± 1.13 | 2.1407 |
| Pima | 77.71 ± 1.00 | 0.0484 | **78.76**± 2.03 | 0.0142 | 77.56± 2.44 | 0.0542 | 76.73± 4.25 | 0.0842 |
| Sonar | **91.49**± 2.50 | 0.0041 | 75.85± 7.40 | 0.0008 | 81.69± 3.45 | 0.0068 | 88.54± 4.31 | 0.0039 |
| Adult | 84.66 | 15.1027 | **84.79** | 1.7757 | 84.71 | 26.1425 | 84.56 | 2.5570 |

**Table 6**
Performance comparison of TELM, ELM, TSVM and SVMwith Gaussian kernel.

| Data sets | TELM | | ELM | | TSVM | | SVM | |
|---|---|---|---|---|---|---|---|---|
| | Accuracy | Times (s) | Accuracy | Times (s) | Accuracy | Times (s) | Accuracy | Times (s) |
| Wine | 100.00 ± 0 | 0.0036 | 98.74± 0.79 | 0.0135 | 100.00± 0 | 0.0058 | 98.57± 1.51 | 0.0008 |
| Vowel | **100.00** | 0.0535 | 44.59 | 0.0146 | 99.35 | 0.6810 | 70.35 | 0.0115 |
| Segment | **99.98**± 0.11 | 1.8255 | 88.93± 0.38 | 0.1953 | 99.87± 0.12 | 0.6781 | 96.93± 0.78 | 0.0344 |
| Satimage | **95.85** | 10.4335 | 90.85 | 0.4212 | 95.60 | 13.4074 | 91.95 | 0.8533 |
| Vehicle | **85.86**± 3.39 | 0.1581 | 84.32± 1.85 | 0.0392 | 81.78± 1.74 | 0.1646 | 83.85± 1.12 | 0.1198 |
| DNA | **99.58** | 2.3511 | 95.19 | 0.2923 | 98.65 | 3.8963 | 95.53 | 0.4441 |
| MNISTA | **99.88**± 0.11 | 0.4015 | 94.87± 0.20 | 0.1606 | 99.10± 0.47 | 0.8318 | 92.95± 1.47 | 0.3319 |
| USPS | **99.85** | 24.0354 | 94.67 | 0.5148 | 99.50 | 35.1993 | 95.72 | 3.4399 |
| Pendigit | 97.74 | 23.9337 | 98.14 | 5.5348 | 88.59 | 32.4331 | **98.31** | 1.2159 |

(2) The results of algorithm with a nonlinear kernel are superior to the algorithm with a linear kernel. This is because the samples can be mapped into a high-dimensional space, which can be separated more easily.
(3) TELM performs better than TSVM on most tasks, which verifies the effectiveness and the potential of TELM.

From the training time comparison of different algorithms, it is interesting to see that SVM is slightly faster than TELM and TSVM on many data sets. However, in theory, solving two small QPPs should be faster than solving a large QPP, so TELM and TSVM should be faster than SVM, which is consistent to the results in [14]. We attribute this phenomenon to two reasons. The first reason is that the methods of solving QPP can be different, and the second reason is that we use the SVM code in the famous libsvm library[1] whose basic algorithm is written in C language.

*6.4.2. Two-dimensional projection*

To clearly understand TELM, we can visualize our classification results by projecting a given data to a two-dimensional space according to the distance between the data and the two learned nonparallel separating hyperplanes.

We take the distance from the data point to the two hyperplane as the coordinates. TELM with Sigmoid kernel is tested on Ionosphere and Bupa Liver Disorders data sets. Fig. 2 shows the scatter plot of the test data set. Here, we define $c_i$ and $d_i$ are the distance from the data point $\boldsymbol{x}_i$ to the two hyperplanes respectively, and each point in the graph can be drawn according to its coordinate $(c_i, d_i)$. We can easily predict the class of each point by comparing $c_i$ and $d_i$, thus a diagonal line is also drawn in Fig. 2. The misclassified data are marked in red.

Based on the scatter plot of the projection points, we can easily evaluate the performance of a classifier. If the projected data can be clearly aggregated into two groups, and the distance is large, indicating that the classifier on the data set can achieve very good performance. From Fig. 2, we can observe that the points in different classes in Fig. 2(a) cluster more better than those in Fig. 2(b). This also shows that the two classes in Fig. 2(a) can be separated more easily than the data in Fig. 2(b), which results to a better classification accuracy. This is consistent with our testing results.

*6.4.3. Parameter sensitivity*

In order to achieve a better prediction accuracy, it is crucial to appropriately choose parameters $C$ and $\kappa$ for SVM and TELM with Gaussian kernel. We have conducted empirical parameter sensitivity analysis to show that TELM can achieve optimal results under a wide range of parameter values.

For each data set, $C$ and $\kappa$ can take 37 different values, resulting in 1369 combinations of $(C, \kappa)$. Fig. 3 shows the sensitivity of the TELM and SVM algorithms to parameters $C$ and $\kappa$. As can be seen from the Fig. 3, SVM and TELM can achieve better prediction accuracy when $\kappa$ is large and $C$ takes a relative small value. It is clear to see that SVM is sensitive to both $\kappa$ and $C$, but TELM is not sensitive to parameter $C$. Thus, in practice, we can choose $C$ first and then select a reasonable $\kappa$ for TELM.

TELMs with Sigmoid kernel also need to select parameter $C$ and parameter $L$. Fig. 4 shows the sensitivity of TELM to parameter $C$ and parameter $L$. From the Fig. 4, we can see that TELM is not very sensitive to the combination of parameters $(C, L)$ on many data sets. (Due to the space limitation, we only plot two cases as examples.) It is worth noting that the increasing of $L$ will consume more time, but the prediction accuracy stays stable. Thus we recommend the readers to select a relative small $L$ for TELM.

From the above analysis, we can see that TELM is not very sensitive to the parameters on many tasks. It can also be explained why we fixed the value of parameter $C$ in the previous experiments.

## 7. Conclusion

In the paper, we propose a new approach for data classification problem, termed as TELM, which extends ELM to two

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvm/ .

nonparallel separating hyperplanes classifier. TELM first uses the random feature mapping mechanism to construct the network, and then learns two nonparallel separating hyperplanes in the random feature space. For each hyperplane, TELM jointly minimizes its distance to one class and requires it to be far away from the other class. TELM incorporates the idea of TSVM into the basic framework of ELM, thus TELM could inherit the advantages of the both algorithms.

As demonstrated by the experimental results, TELM achieves similar or better performance on the binary classification problem compared to other famous classification algorithms such as ELM, SVM and TSVM, and performs much better on multi-class classification problem. When the number of sample features is large, TELM has much faster learning speed than TSVM. Moreover, it is worth noting that TELM has less sensitive to the user-specified parameters.

The experimental results demonstrate that TELM will greatly expand the application of ELM in pattern classification and provide a paradigm of new depth insight into ELM.

## Acknowledgment

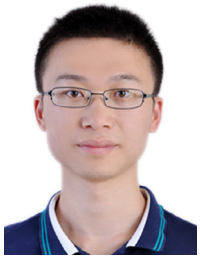
This research is supported by the National Natural Science Foundation of China under Grant nos. 41427806 and 61273233, and the National Key R&D Program under Grant no. 2016YFB1200203.

## References

[1] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536.
[2] M.T. Hagan, M.B. Menhaj, Training feedforward networks with the marquardt algorithm, IEEE Trans. Neural Netw. 5 (6) (1994) 989–993.
[3] B.M. Wilamowski, N.J. Cotton, O. Kaynak, G. Dundar, Computing gradient vector and jacobian matrix in arbitrarily connected neural networks, IEEE Trans. Ind. Electron. 55 (10) (2008) 3784–3790.
[4] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[5] C.J. Burges, A tutorial on support vector machines for pattern recognition, Data Min. Knowl. Discov. 2 (2) (1998) 121–167.
[6] J.A. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (3) (1999) 293–300.
[7] Y.-J. Lee, O.L. Mangasarian, RSVM: reduced support vector machines, in: Proceedings of the SIAM International Conference on Data Mining, 2001.
[8] G.M. Fung, O.L. Mangasarian, Multicategory proximal support vector machine classifiers, Mach. Learn. 59 (1–2) (2005) 77–97.
[9] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of 2004 IEEE International Joint Conference on Neural Networks, 2004, vol. 2, IEEE, 2004, pp. 985–990.
[10] G. Huang, S. Song, J.N. Gupta, C. Wu, Semi-supervised and unsupervised extreme learning machines, IEEE Trans. Cybern. 44 (12) (2014) 2405–2417.
[11] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1) (2006) 489–501.
[12] G. Huang, G.-B. Huang, S. Song, K. You, Trends in extreme learning machines: a review, Neural Netw. 61 (2015) 32–48.
[13] G.-B. Huang, X. Ding, H. Zhou, Optimization method based extreme learning machine for classification, Neurocomputing 74 (1) (2010) 155–163.
[14] R. Khemchandani, S. Chandra, et al., Twin support vector machines for pattern classification, IEEE Trans. Pattern Anal. Mach. Intell. 29 (5) (2007) 905–910.
[15] Y.-H. Shao, C.-H. Zhang, X.-B. Wang, N.-Y. Deng, Improvements on twin support vector machines, IEEE Trans. Neural Netw. 22 (6) (2011) 962–968.
[16] Z. Qi, Y. Tian, Y. Shi, Laplacian twin support vector machine for semi-supervised classification, Neural Netw. 35 (2012) 46–53.
[17] M.A. Kumar, M. Gopal, Least squares twin support vector machines for pattern classification, Expert Syst. Appl. 36 (4) (2009) 7535–7543.
[18] K. Ning, M. Liu, M. Dong, C. Wu, Z. Wu, Two efficient twin elm methods with prediction interval, Neural Netw. Learn. Syst. 26 (9) (2015) 2058–2071.
[19] D.P. Bertsekas, Nonlinear Programming, Athena Scientific, 1999.
[20] J. Barzilai, J.M. Borwein, Two-point step size gradient methods, IMA J. Numer. Anal. 8 (1) (1988) 141–148.
[21] Y. Nesterov, A method of solving a convex programming problem with convergence rate o (1/k2), Sov. Math. Dokl. vol. 27 (1983) 372–376.
[22] M. Celis, J. Dennis, R. Tapia, A trust region strategy for nonlinear equality constrained optimization, Numer. Optim. 1984 (1985) 71–82.
[23] Z.-Q. Luo, P. Tseng, Error bounds and convergence analysis of feasible descent methods: a general approach, Ann. Oper. Res. 46 (1) (1993) 157–178.
[24] O.L. Mangasarian, D.R. Musicant, Successive overrelaxation for support vector machines, IEEE Trans. Neural Netw. 10 (5) (1999) 1032–1037.
[25] C. Blake, C.J. Merz, UCI Repository of Machine Learning Databases, [online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.
[26] J. Hull, A database for handwritten text recognition research, IEEE Trans. Pattern Anal. Mach. Intell. 16 (1994) 550–554.
[27] J. Li, H. Liu, Kent Ridge Bio-medical Data Set Repository, Institute for Infocomm Research, 2002.
[28] C.-C. Chang, C.-J. Lin, LIBSVM : a library for support vector machines, [online]. Available: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/, 2011.
[29] S. Canu, Y. Grandvalet, V. Guigue, SVM and kernel methods matlab toolbox, [online]. Available: http://asi.insarouen.fr/enseignants/~arakotom/toolbox/index.html, 2005.
[30] Y.-H. Shao, C.-H. Zhang, X.-B. Wang, N.-Y. Deng, Improvements on twin support vector machines, [online]. Available: http://www.optimal-group.org/Resource/TWSVM.html, 2011.
[31] G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, [online]. Available: http://www.ntu.edu.sg/home/egbhuang/elm_random_hidden_nodes.html, 2012.

**Yihe Wan** received the B.S. degree from the Military Equipment at Army Officer Academy of PLA in 2003. From 2012 he started his Ph.D. at Institute of System Integration, Department of Automation, Tsinghua University, China. His current research interests include machine learning, pattern recognition and system modeling.

**Shiji Song** received the Ph.D. degree from the Department of Mathematics at Harbin Institute of Technology in 1996. He is a Professor in the Department of Automation, Tsinghua University. His research interests include system modeling, control and optimization, computational intelligence and pattern recognition.



**Gao Huang** received the B.S. degree from the School of Automation Science and Electrical Engineering at Beihang University, in 2009, and the Ph.D. degree from the Department of Automation at Tsinghua University in 2015. He was a Visiting Research Scholar with the Department of Computer Science and Engineering, Washington University in St. Louis, in 2013. He is currently a joint Post-Doctoral Researcher at the Department of Automation at Tsinghua University and the Department of Computer Science at Cornell University. His current research interests include machine learning and statistical pattern recognition.

**Shuang Li** received his B.S. degrees in Department of Automation, Northeastern University in 2012. From 2012 he started his Ph.D. at Institute of System Integration, Department of Automation, Tsinghua University, China. He was a Visiting Research Scholar at the Department of Computer Science, Cornell University from November 2015 to June 2016. His main research interests include machine learning and pattern recognition, especially in transfer learning and domain adaptation.