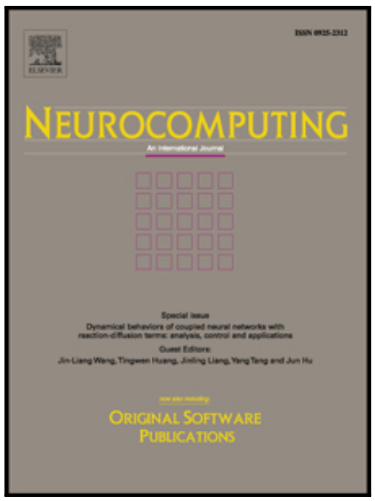


Optimization Extreme Learning Machine with ν Regularization

Ding Xiao-jian , Lan Yuan , Zhang Zhi-feng , Xu xin

PII: S0925-2312(17)30199-6
DOI: 10.1016/j.neucom.2016.05.114
Reference: NEUCOM 17998

To appear in: *Neurocomputing*

Received date: 30 September 2015
Revised date: 19 May 2016
Accepted date: 27 May 2016

Please cite this article as: Ding Xiao-jian , Lan Yuan , Zhang Zhi-feng , Xu xin , Optimization Extreme Learning Machine with ν Regularization, *Neurocomputing* (2017), doi: 10.1016/j.neucom.2016.05.114

Optimization Extreme Learning Machine with ν Regularization

Ding Xiao-jian[1], Lan Yuan[2], Zhang Zhi-feng[3], and Xu xin[1]

[1]Science and Technology on Information Systems Engineering Laboratory,
Nanjing 210007, China

[2]Key Laboratory of Ministry of Education in Advance Transducers and Intelligent Control System,
School of Mechanical Engineering, Taiyuan University of Technology, Taiyuan 030024, China

[3]Software college, Zhengzhou University of Light Industry, Zhengzhou 450002, China

wjswsl@163.com

Abstract: The problem of choosing error penalty parameter *C* for optimization extreme learning machine (OELM) is that it can take any positive value for different applications and it is therefore hard to choose correctly. In this paper, we reformulated OELM to take a new regularization parameter ν (ν-OELM) which is inspired by Schölkopf et al. The regularization in terms of ν is bounded between 0 and 1, and is easier to interpret as compared to *C*. This paper shows that: (1) ν-OELM and ν-SVM have similar dual optimization formulation, but ν-OELM has less optimization constraints due to its special capability of class separation; (2) experiment results on both artificial and real binary classification problems show that ν-OELM tends to achieve better generalization performance than ν-SVM, OELM and other popular machine learning approaches, and it is computationally efficient on high dimension data sets. Additionally, the optimal parameter ν in ν-OELM can be easily selected from few candidates.

Key words: ν-optimization extreme learning machine, classification, parameter selection

1. Introduction

Extreme learning machine (ELM) introduced by Huang and co-workers [1-5] is one of the popular tools for data classification and regression. In ELM, the input weights and the bias of the hidden nodes are randomly generated and the output weights are analytically calculated instead of iterative tuned. ELM has been proved to have good generalization performance and extremely fast learning speed. Due to its effectiveness and fast learning process, ELM has been applied in many fields [6-9]. However, it still tends to be overfitting, especially on small training instances. Thus，the stability and generalization performance of ELM could be improved.

Recently, an optimization extreme learning machine (OELM) was proposed for binary classification problems [10-14]. OELM is an optimization classifier based on conventional ELM. OELM implements the bartlett's theory [15], for which the smaller the norm of the output weights is, the better generalization performance the system tends to have. Compared to ELM, the minimization norm of output weights enables OELM to get better generalization performance. OELM solves a quadratic programming (QP) problem, which assures that a global optimal solution can be found. In OELM, to minimize the norm of the output weights is actually to find a separating hyperplane with the maximal margin between two classes of data, which is similar to the idea employed in support vector machine (SVM). Compared to SVM [16-17], OELM finds the optimal solution in the search space of $[0, C]^N$, where SVM always searches the sub-optimal solution of OELM due to its equation constraint. Empirical studies based on real benchmark problems have shown that compared with classical learning algorithms (such as SVM and ELM), OELM tends to provide better generalization ability with low

computational cost, and builds the training model without frequently tuning the parameters.

It has been shown that OELM requires ELM kernel parameter *L* and penalty parameter *C* to be tuned. Interestingly, Frénay [18] and Huang [10] found that ELM-type learning model usually maintains good generalization ability as long as the kernel parameter (the number of hidden nodes) is large enough. In fact, one can set the proper kernel parameter (e.g. 10^3) before seeing the training data. Parameter *C* determines the trade-off between the training error and generalization performance. However, *C* is a rather non-heuristic parameter that we have no a priori method to choose. Therefore time consuming methods like grid search or cross-validation are employed to select the optimal value of *C*. The same problem exists in the conventional SVM formulation as well. To solve the issue, Schölkopf et al. [19] reformulated SVM to introduce a new regularization parameter ν that is bounded between 0 and 1. This parameter is an upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors relative to the total number of training samples. It is the intuitive meaning that tuning the parameter ν is easier than parameter *C*.

To overcome the problem of the difficulty in selecting parameter *C*, we first present a new formulation with parameter ν instead of parameter *C*, and then derive the Karush-Kuhn-Tucker (KKT) optimality condition. Finally, we discuss an active set algorithm for solving the formulated problem. This paper further shows that (1) dual formulation of ν-OELM is very similar to the dual formulation of ν-SVM, except that ν-OELM has fewer optimization constraints; (2) according to the ELM theories all the training data are linearly separable by a hyperplane passing through the origin with probability one in the ELM feature space, ν-OELM tends to achieve better generalization performance than ν-SVM; (3) the optimal parameter ν of ν-OELM can be easily selected from a few candidates.

This paper is organized as follows. In Section 2, the fundamental knowledge of OELM and ν-SVM is introduced. Section 3 presents the optimization problem of ν-OELM and derives its dual problem. In section 4, we propose an active set algorithm for solving the dual problem of ν-OELM. Section 5 compares ν-OELM with other state-of-the-art classifiers for both artificial and real benchmark data sets. Section 6 concludes the paper.

2. Related works

In this section, the fundamentals of OELM and ν-SVM are reviewed.

2.1 OELM

Consider the problem of binary classification. For *N* arbitrary distinct samples $(\mathbf{x}_i, t_i)$, $i = 1,..., N$ where $\mathbf{x}_i = [x_{i1}, x_{i2},..., x_{id}] \in \mathrm{R}^d$ and $t_i \in \{-1, 1\}$, the decision function given by an ELM is

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

where β_i is the output weight from the *i*-th hidden node to the output node and $G(\mathbf{a}_i, b_i, \mathbf{x})$ is the output of the *i*-th hidden node. $h(\mathbf{x}) = [G(\mathbf{a}_1, b_1, \mathbf{x}),..., G(\mathbf{a}_L, b_L, \mathbf{x})]$ is the output vector of the hidden layer, and it can map the training data $\mathbf{x}_i$ from the input space to the *L*-dimensional ELM feature space.

Based on the theory of Bartlett [15], feedforward neural network with smaller norm of weights, not only has the smaller training error, but also obtains the better generalization performance. Therefore, ELM aims to minimizing the training error with the minimized norm of the output weight

$$\begin{aligned} &\text{Minimize}: \sum_{i=1}^{N} \left\| \boldsymbol{\beta} \cdot h(\mathbf{x}_i) - t_i \right\| \\ &\text{Minimize}: \left\| \boldsymbol{\beta} \right\| \end{aligned} \tag{2}$$

Any set of training data transformed from the input space to the ELM feature space with the mapping $h(\mathbf{x})$ is linearly separable. In order to prevent the classification problem from overfitting,

variables $\xi_i, i = 1,...,N$ are introduced and one can minimize the testing error as follows

$$\begin{aligned} \boldsymbol{\beta} \cdot h(\mathbf{x}_i) &\geq 1 - \xi_i \quad \text{for } t_i = +1 \\ \boldsymbol{\beta} \cdot h(\mathbf{x}_i) &\leq -1 + \xi_i \quad \text{for } t_i = -1 \end{aligned} \tag{3}$$

Two set of inequalities in (3) can be combined into one set of inequalities

$$t_i \cdot \boldsymbol{\beta} \cdot h(\mathbf{x}_i) \geq 1 - \xi_i \quad \forall i \tag{4}$$

Thus, OELM algorithm with penalization of the training errors consists of solving the following quadratic program mathematical model

$$\begin{aligned} \text{Minimize:} \quad & \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{Subject to:} \quad & t_i \cdot \boldsymbol{\beta} \cdot h(\mathbf{x}_i) \geq 1 - \xi_i, \quad i = 1,...,N \\ & \xi_i \geq 0, \ \ i = 1,...,N \end{aligned} \tag{5}$$

2.2 ν-SVM

The ν-SVM primal formulation problem, as given in [19], is as follows:

$$\begin{aligned} \text{Minimize:} \quad & \frac{1}{2}\|\mathbf{w}\|^2 - \nu\rho + \frac{1}{N}\sum_{i=1}^{N}\xi_i \\ \text{Subject to:} \quad & t_i(\mathbf{w} \cdot \phi(\mathbf{x}_i) + b) \geq \rho - \xi_i, \quad i = 1,...,N \\ & \xi_i \geq 0, \ \ i = 1,...,N, \ \ \rho \geq 0 \end{aligned} \tag{6}$$

Here ν is the user specified parameter between 0 and 1, and training data $\mathbf{x}_i$ are mapped into a feature space by through a mapping $\phi(\mathbf{x})$. The Wolfe dual of this problem is:

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

where each Lagrange multiplier α_i corresponds to a training example $(\mathbf{x}_i, t_i)$, $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ is the kernel.

3. ν-optimization extreme learning machine (ν-OELM)

In the standard formulation of OELM, user specified parameter *C* has no direct interpretation, and it can take any positive value. It is therefore difficult to choose parameter *C* correctly and one has to resort to cross validation or direct experimentation to look for a suitable value. In this section, we will present a new formulation for OELM, in which parameter *C* is replaced by parameter ν.

3.1 optimization formulation

By introducing the new parameter ν, the hyperplane $u(\mathbf{x}) = \boldsymbol{\beta} \cdot h(\mathbf{x}_i)$ of OELM separates the data if and only if

$$t_i \cdot \boldsymbol{\beta} \cdot h(\mathbf{x}_i) \geq \rho - \xi_i \quad \forall i \tag{8}$$

This changes the width of the margin in OELM to $2\rho/\|\boldsymbol{\beta}\|$, which is to be maximized while minimizing the margin errors. The hyperplane is determined by solving the following primal problem

$$\begin{aligned} \text{Minimize:} \quad & L_P = \frac{1}{2}\|\boldsymbol{\beta}\|^2 - \nu\rho + \frac{1}{N}\sum_{i=1}^{N}\xi_i \\ \text{Subject to:} \quad & t_i \boldsymbol{\beta} \cdot h(\mathbf{x}_i) \geq \rho - \xi_i, \quad i = 1,...,N \\ & \rho \geq 0, \quad \xi_i \geq 0, \ \ i = 1,...,N \end{aligned} \tag{9}$$

where ξ_i is the slack variable of $\mathbf{x}_i$, ρ is the variable to change the margin and ν is penalty weight that is a positive constant.

For these constraints, we consider using Lagrangian method and we have

$$L_1(\boldsymbol{\beta},\boldsymbol{\xi},\rho,\boldsymbol{\alpha},\delta,\boldsymbol{\mu}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 - v\rho + \frac{1}{N}\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(\, t_i\boldsymbol{\beta}\cdot h(\mathbf{x}_i) - \rho + \xi_i) - \delta\rho - \sum_{i=1}^{N}\mu_i\xi_i \tag{10}$$

where multipliers $\alpha_i, \delta, \mu_i \geq 0$. This function has to be minimized with respect to variables $(\boldsymbol{\beta},\boldsymbol{\xi},\rho)$ of the primal problem and maximized with respect to dual variables $(\boldsymbol{\alpha},\delta,\boldsymbol{\mu})$. Setting the gradients of this Lagrangian expression with respect to $(\boldsymbol{\beta},\boldsymbol{\xi},\rho)$ equal to 0 gives the following KKT optimality conditions

$$\begin{aligned}
&\frac{\partial L_1(\boldsymbol{\beta},\boldsymbol{\xi},\rho,\boldsymbol{\alpha},\delta,\boldsymbol{\mu})}{\partial\boldsymbol{\beta}} = 0 \quad \Rightarrow\ \boldsymbol{\beta} - \sum_{i=1}^{N}\alpha_i t_i h(\mathbf{x}_i) = 0 \\
&\frac{\partial L_1(\boldsymbol{\beta},\boldsymbol{\xi},\rho,\boldsymbol{\alpha},\delta,\boldsymbol{\mu})}{\partial\boldsymbol{\xi}} = 0 \quad \Rightarrow\ \alpha_i + \mu_i - \frac{1}{N} = 0 \\
&\frac{\partial L_1(\boldsymbol{\beta},\boldsymbol{\xi},\rho,\boldsymbol{\alpha},\delta,\boldsymbol{\mu})}{\partial\rho} = 0 \quad \Rightarrow\ \sum_{i=1}^{N}\alpha_i - \delta - v = 0
\end{aligned} \tag{11}$$

Substituting three equations of (11) into L_{ELM} leaves us with the following quadratic optimization problem

$$\begin{aligned}
&\text{Maximize}: \quad L_D = -\frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j t_i t_j h(\mathbf{x}_i)\cdot h(\mathbf{x}_j) \\
&\text{Subject to}: \quad 0 \leq \alpha_i \leq \frac{1}{N}, \quad \sum_{i=1}^{N}\alpha_i \geq v, \quad i = 1,...,N
\end{aligned} \tag{12}$$

Incorporating kernels for dot products of $h(\mathbf{x}_i)$, solving (12) is equivalent to solving the following dual optimization problem

$$\begin{aligned}
&\text{Minimize}: \quad L_D = \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j t_i t_j K_{\mathrm{ELM}}(\mathbf{x}_i,\mathbf{x}_j) \\
&\text{Subject to}: \quad 0 \leq \alpha_i \leq \frac{1}{N}, \quad \sum_{i=1}^{N}\alpha_i \geq v, \quad i = 1,...,N
\end{aligned} \tag{13}$$

ELM kernel function K_{ELM} is defined as

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

where $G(\boldsymbol{a},b_1,\mathbf{x})$ is nonlinear piecewise continuous function satisfying ELM universal approximation capability theorems, and the input weights $\mathbf{a}$ and biases b are randomly generated from $(-1, 1)^N \times (0, 1)$ based on the uniform probability distribution.

The resulting decision function can be shown as

$$f(\mathbf{x}) = \mathrm{sign}\left(\sum_{i=1}^{N}\alpha_i t_i K_{\mathrm{ELM}}(\mathbf{x},\mathbf{x}_i)\right) \tag{14}$$

There are several differences between proposed ν-OELM and ν-SVM formulation:

(1) The mapping function used in ν-SVM is usually unknown and cannot be computed directly. Function $h(\mathbf{x}_i)$ used in OELM can be any bounded non-constant piecewise continuous activation function for additive node, which can be calculated exactly.

(2) Kernel parameters in ν-SVM need to be tuned manually, whereas all parameters of $h(\mathbf{x})$ for ν-OELM are chosen randomly.

(3) The bias b is not required in the constraints of ν-OELM's optimization problem, because the separating hyperplane of OELM tends to pass through the origin.

3.2 Karush-Kuhn-Tucker conditions of ν-OELM

From the KKT optimality condition [20], primal and dual optimal solutions satisfy the following slackness conditions.

Primal feasibility

$$t_i \boldsymbol{\beta} \cdot h(\mathbf{x}_i) - \rho + \xi_i \geq 0, \quad \forall i \tag{15}$$

$$\rho \geq 0 \tag{16}$$

$$\xi_i \geq 0 \tag{17}$$

Dual feasibility

$$\alpha_i \geq 0, \quad \forall i \tag{18}$$

$$\delta \geq 0, \quad \forall i \tag{19}$$

$$\mu_i \geq 0, \quad \forall i \tag{20}$$

Complementary slackness

$$\alpha_i (t_i \boldsymbol{\beta} \cdot h(\mathbf{x}_i) - \rho + \xi_i) = 0, \quad \forall i \tag{21}$$

$$\delta \rho = 0, \quad \forall i \tag{22}$$

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

From the second equation of (11) and (23), we haves

$$(\frac{1}{N} - \alpha_i)\xi_i = 0 \tag{24}$$

Let us define f_i as the predicted value of the data point α_i ($f_i = f(\alpha_i)$). If $\alpha_i = 0$, from (24) we have $\xi_i = 0$. According to (15) and (16), obviously we have $f_i \geq 0$. If $\alpha_i = \frac{1}{N}$, from (21) we have $f_i - \rho + \xi_i = 0$. According to the third equation of (11) and (16), we have $\rho = 0$. From (15) and (17) we have $f_i \leq 0$. If $0 < \alpha_i < \frac{1}{N}$, from (24) we have $\xi_i = 0$. According to (21), we have $f_i \geq 0$. Thus, we call $\bar{\boldsymbol{\alpha}} \in \mathbb{R}^N$ is a stationary point if there exists a vector $\bar{\boldsymbol{\alpha}}$ solves the KKT conditions

$$\begin{aligned} &f_i \geq 0 \text{ if } 0 \leq \bar{\alpha}_i < \frac{1}{N} \\ &f_i \leq 0 \text{ if } \bar{\alpha}_i = \frac{1}{N} \end{aligned} \tag{25}$$

where $\bar{\boldsymbol{\alpha}}=[\bar{\alpha}_1,...,\bar{\alpha}_N]$.

3.3 Discussions

ν-SVM and ν-OELM have similar dual optimization formulations and ν-SVM needs to satisfy one more optimization condition $\sum_{i=1}^{N} \alpha_i t_i = 0$ as compared to ν-OELM. Obviously, ν-SVM tends to find the sub-optimal solution of the original problem.

For ν-SVM, the trivial things we need to consider are the choice of kernel and the parameters for the chosen kernel. Generally speaking, one has to do many simulations to find the best combination of kernel parameter and penalty parameter. On the contrary, kernel parameter tuning is not necessary in ν-OELM and the parameter can be set before seeing the training samples.

As Schölkopf et al. already pointed out, parameter ν is an upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors relative to the whole training samples. In other words, if we set the value of ν to be 0.1, at most 10% of training samples could be misclassified and at least 10% of training samples could be support vectors. Moreover, when ν is 0.9, the margin is very large and most of the data points will fall in the margin.

4. ν-OELM algorithm

Based on existing active set method for OELM [12], in this section we introduce an active set

method to solve the quadratic programming optimization problem of ν-OELM. In the method, a subset of the variables is fixed at their bounds and the objective function is minimized with respect to the remaining variables. After a number of iterations, the correct active set is identified and the objective function converges to a stationary point.

For the sake of convenience, (13) can be simply written as

$$\begin{aligned} &\text{Minimize:} \quad \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}} K\boldsymbol{\alpha} \\ &\text{Subject to:} \quad \sum\boldsymbol{\alpha} \geq \nu,\ 0 \leq \alpha_i \leq \frac{1}{N},\ i = 1,...,N \end{aligned} \tag{26}$$

To solve the problem (26), the inequality $\sum_{i=1}^{N}\alpha_i \geq \nu$ can be reformulated by an equality $\sum_{i=1}^{N}\alpha_i = \nu$ [19,21]. Thus, the quadratic problem (26) is replaced by

$$\begin{aligned} &\text{Minimize:} \quad \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}} K\boldsymbol{\alpha} \\ &\text{Subject to:} \quad \sum\boldsymbol{\alpha} = \nu,\ 0 \leq \alpha_i \leq \frac{1}{N},\ i = 1,...,N \end{aligned} \tag{27}$$

We begin with some notations. For a point $\boldsymbol{\alpha}$ in the feasible region, we define $L = \{i \mid \alpha_i = 0\}$ and $U = \left\{i \mid \alpha_i = \frac{1}{N}\right\}$ the indices set of the active constraints, and the set $\{L \cup U\}$ of bound variables to be the active set at the current point. We also define $S = \left\{i \mid 0 < \alpha_i < \frac{1}{N}\right\}$ the free set, and the set S of variables are corresponding to free variables.

The vector of $\boldsymbol{\alpha}$ variables whose indices belong to the set L will be denoted by $\boldsymbol{\alpha}_L$, and other $\boldsymbol{\alpha}$ variables will be denoted by $\boldsymbol{\alpha}_U$ and $\boldsymbol{\alpha}_S$ respectively. Corresponding to the choice of indices set L, U and S, we partition and rearrange the matrix K as follows

$$K = \begin{bmatrix} K_{LL} & K_{LS} & K_{LU} \\ K_{SL} & K_{SS} & K_{SU} \\ K_{UL} & K_{US} & K_{UU} \end{bmatrix}$$

Thus, the objective function value of (27) is equal to $\frac{1}{2}\boldsymbol{\alpha}_S^T K_{SS}\boldsymbol{\alpha}_S + \boldsymbol{\alpha}_U^T K_{SU}\boldsymbol{\alpha}_S + \frac{1}{2}\boldsymbol{\alpha}_U^T K_{UU}\boldsymbol{\alpha}_U$. At each iteration, $\boldsymbol{\alpha}_U$ is fixed and problem (27) has the following form:

$$\begin{aligned} \underset{\boldsymbol{\alpha}_S}{\text{Minimize}} \quad & \frac{1}{2}\boldsymbol{\alpha}_S^T \boldsymbol{K}_{SS}\boldsymbol{\alpha}_S + \boldsymbol{\alpha}_U^T \boldsymbol{K}_{SU}\boldsymbol{\alpha}_S \\ \text{Subject to} \quad & \sum\boldsymbol{\alpha}_S = \nu - \sum\boldsymbol{\alpha}_U, \quad 0 \leq \alpha_i \leq \frac{1}{N}, \quad i \in S \end{aligned} \tag{28}$$

Under the definition of $\boldsymbol{\alpha}_S$, all variables are satisfied for $(0, \frac{1}{N})$, so problem (28) can be transformed to the equality constrained optimization problem

$$\begin{aligned} \underset{\boldsymbol{\delta}}{\text{Minimize}} \quad & \frac{1}{2}\boldsymbol{\eta}^T \boldsymbol{H}\boldsymbol{\eta} + \boldsymbol{c}^T\boldsymbol{\eta} \\ \text{Subject to} \quad & \boldsymbol{A}\boldsymbol{\eta} = t \end{aligned} \tag{29}$$

where $\boldsymbol{\eta} = \boldsymbol{\alpha}_S$, $\boldsymbol{H} = \boldsymbol{K}_{SS}$, $\boldsymbol{c} = \boldsymbol{K}_{SU}\boldsymbol{\alpha}_U$, $\boldsymbol{A}$ is the vector of all ones, and $t = \nu - \sum\boldsymbol{\alpha}_U$.

The Lagrangian expression for this problem is

$$L_2(\boldsymbol{\eta},\lambda)=\frac{1}{2}\boldsymbol{\eta}^T\boldsymbol{H}\boldsymbol{\eta}+\boldsymbol{c}^T\boldsymbol{\eta}-\lambda(\boldsymbol{A}\boldsymbol{\eta}-t) \tag{30}$$

The partial derivatives of the Lagrangian expression are set to zero, which leads to the following simple linear system

$$\begin{bmatrix}\boldsymbol{H} & \boldsymbol{A}^T\\ \boldsymbol{A} & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{\eta}\\ \lambda\end{bmatrix}=\begin{bmatrix}-\boldsymbol{c}\\ t\end{bmatrix} \tag{31}$$

Then, the optimal solution η with the corresponding λ can be solved by (31).

Indeed, we consider choosing a two-loop approach to minimize the objective function of (27). The outer loop iterates over all instances violating the KKT conditions (25), and the iteration starts from the instances that are not on bound. The outer loop keeps alternating between passes over entire training samples and passes over the non-bound instances. If the optimality conditions are satisfied over all instances, the algorithm stops with the solution; otherwise an inner loop begins.

The inner loop focuses on minimize the sub-problem (28). As (28) is the convex quadratic problem, it has a global minimum. The strict decrease of the objective function holds and the theoretical convergence proof was given in [12].

5. Experiments and results

In our experiments, we have evaluated ν-OELM on a number of standard classification data sets, both artificial and real. The artificial data set includes a classical data set with two gaussianly distributed classes with similar variances but different means. The real world data sets include 11 benchmark problems commonly utilized in previous work. In order to evaluate the effectiveness of the proposed ν-OELM algorithm, we compare our approach not only with the recently proposed OELM algorithm, but also with two other well-accepted learning algorithms: SVM and ν-SVM. All the simulations are running in MATLAB R2008a (Windows version) environments with Intel 3.0 GHZ and 2G RAM. For the implementation of SVM and ν-SVM, we use SVM and Kernel Methods MATLAB Toolbox, which is publicly available from [22]. MATLAB codes of OELM can be downloaded from ELM host site: http://www.ntu.edu.sg/home/egbhuang/.

The popular Gaussian kernel function $K(\mathrm{x}_i,\mathrm{x}_j)=\exp(-\gamma\left\|\mathrm{x}_i-\mathrm{x}_j\right\|^2)$ is used in both SVM and ν-SVM. To train OELM, Sigmoid type of ELM kernel is used: $K_{\mathrm{ELM}}(\mathbf{x},\mathbf{x}_s)=[G(\mathbf{a}_1,b_1,\mathbf{x}),...,G(\mathbf{a}_L,b_L,\mathbf{x})]^{\mathrm{T}}$ $\cdot[G(\mathbf{a}_1,b_1,\mathbf{x}_s),...,G(\mathbf{a}_L,b_L,\mathbf{x}_s)]^{\mathrm{T}}$, where $G(\mathbf{a},b,\mathbf{x})=1/(1+\exp(-(\mathbf{a}\cdot\mathbf{x}+b)))$. In addition, for the Sigmoid active function of ELM kernel, the input weights and biases are randomly generated from $(-1, 1)^N\times(0, 1)$ based on the uniform probability distribution.

5.1 Selection of parameters

In order to achieve good generalization performance, we use grid search to determine the kernel parameter γ and penalty parameter C for SVM, and both γ and ν for ν-SVM. Similar to Ghanty et al. [23], penalty parameter C and kernel parameter γ of SVM are tuned on a grid of $\{0.001,0.01,0.05,0.1,0.2,0.5,1,2,5,10, 20, 50,100,1000,10000\}\times\{0.001,0.01,0.1,0.2,0.4,0.8,1, 2, 5, 10, 20, 50, 100, 1000, 10000\}$. When ν-SVM is used, parameters ν and γ are tuned on a grid of $\{0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1\}\times\{0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10, 20, 50, 100, 1000,10000\}$. According to the suggestion of [10-13, 18], OELM tends to achieve better generalization performance when kernel parameter L is large enough. Thus, parameters C and L of OELM are tuned on a grid of $\{0.001,0.01,0.05,0.1,0.2,0.5,1,2,5,10, 20, 50,100,1000,10000\}\times\{1000, 1200, 1500, 1800, 2000, 2500, 3000, 3500, 4000, 5000\}$. Similarly, parameters v and L of ν-OELM are tuned on a grid of $\{0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1\}\times\{1000, 1200, 1500, 1800, 2000, 2500, 3000, 3500, 4000, 5000\}$.

5.2 Artificial data set

To illustrate graphically the effectiveness of ν-OELM classifier, we test its abilities on an artificial data set. We create artificial data set of Gaussian distribution by using data Generator in SVM and Kernel Methods in MATLAB Toolbox. The artificial data set consists of 300 training points and 900 testing points. In artificial data set, the two classes of data points are generated from the mixtures of two Gaussian distributions with overlapping samples, as shown in Figure 1.

(a) (b)

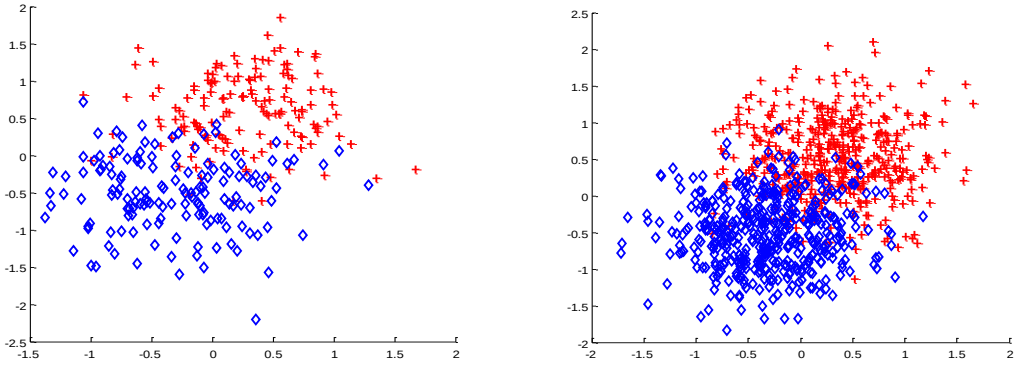


Figure 1 Artificial data set with (a) training points and (b) testing points. The 'plus' and 'diamond' signs indicate the two classes.

We first show the generalization performance comparisons of 4 classifiers over ten trails. In each trail, 300 training points and 900 testing points are randomly generated by data Generator. The parameter combination with the best fitness value is selected as the parameters to build the model. Two measures (testing accuracy and training time) are adopted to describe the performance of 4 classifiers. From Figure 2(a), we can see that ν-OELM tends to have the best testing accuracy over the most trails. Figure 2(b) compares the training time of SVM, ν-SVM, OELM and ν-OELM, the results demonstrate that ν-OELM is slightly slower than SVM and OELM. However, the training time in this experiment is the time for training classifier model, which does not include the time for parameter tuning. The total training time should be the sum of the time for searching the 'best' values among all parameter combinations and the time for training classifier using the 'best' parameter combination chosen. Huang [10] showed that OELM prediction is not sensitive to the selection of the tuning parameter L. In this experiment, we both set kernel parameter L of OELM and ν-OELM to be 2000. To see how the training time is affected by parameter tuning effort, we conduct the following experiment with total training time is involved. Table 1 shows the comparison results for the total training time of 4 classifiers. Obviously, ν-OELM achieves the shortest total training time among these classifiers. For SVM and ν-SVM, 99% of total training time is used to search the 'best' parameter combination.

(a) (b)

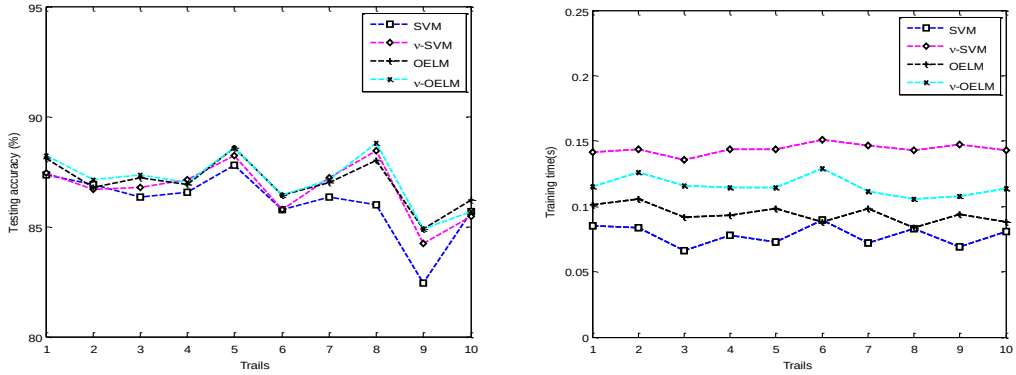


Figure 2 Comparison results for 4 classifiers via 10 different trails on artificial data set. (a) Testing accuracy comparison, and (b) training time comparison.

Table 1 Total training time comparison

Data set	Total training time (s)			
	SVM	ν-SVM	OELM	ν-OELM
Artificial data set	107.9	170.2	1.437	**1.136**

In the second experiment, we focus on the performance comparison of the different kernels used in ν-OELM. Four commonly used kernel functions of ELM kernel $K_{\mathrm{ELM}}(\mathbf{x},\mathbf{x}_s)=[G(\mathbf{a}_1,b_1,\mathbf{x}),...,G(\mathbf{a}_L,b_L,\mathbf{x})]^{\mathrm{T}}\cdot[G(\mathbf{a}_1,b_1,\mathbf{x}_s),...,G(\mathbf{a}_L,b_L,\mathbf{x}_s)]^{\mathrm{T}}$ in literatures are adopted in this experiment:

- Sigmoid function:

$G(\mathbf{a},b,\mathbf{x})=1/(1+\exp(-(\mathbf{a}\cdot\mathbf{x}+b)))$

- Sin function:

$G(\mathbf{a},b,\mathbf{x})=\sin(\mathbf{a}\cdot\mathbf{x}+b)$

- Hard-limit function:

$G(\mathbf{a},b,\mathbf{x})=\mathrm{hardlimit}(\mathbf{a}\cdot\mathbf{x}+b)$

- Exponential function:

$G(\mathbf{a},b,\mathbf{x})=\exp(-(\mathbf{a}\cdot\mathbf{x}+b))$

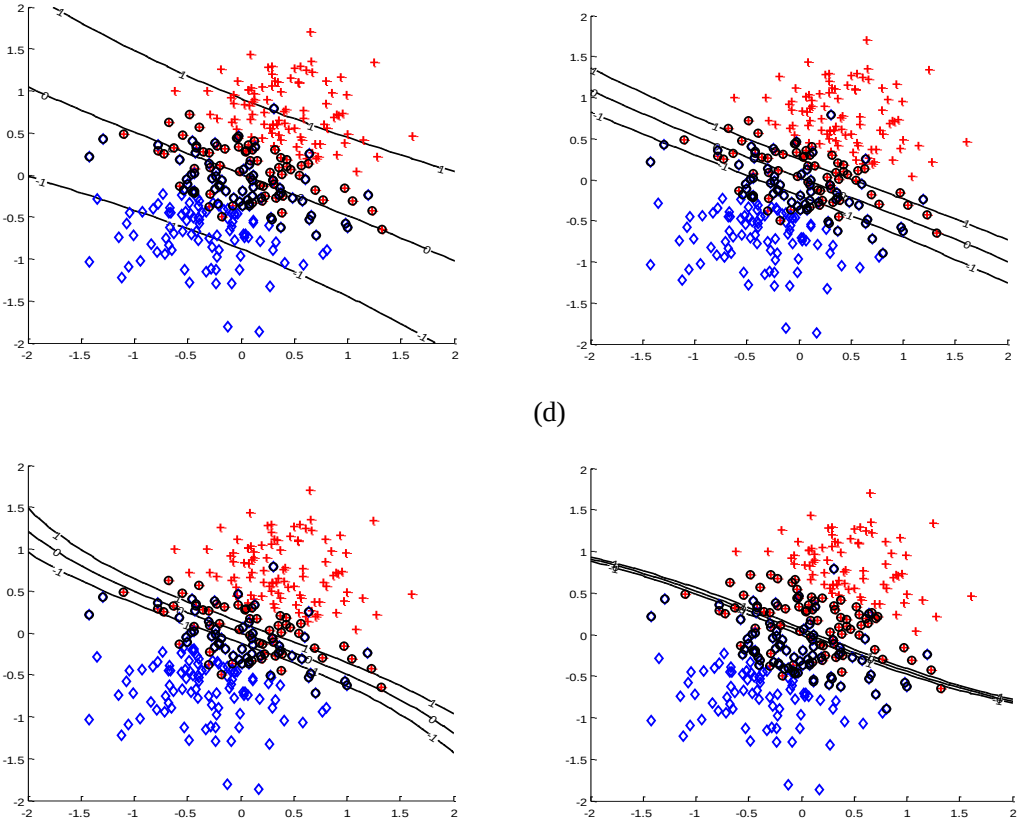
All the vectors $\boldsymbol{\alpha}$ and variables b in these kernel functions are randomly generated from (-1, 1)^N×(0, 1) based on the uniform probability distribution. The accuracy of classification performance on the artificial data set for the above ELM kernels is shown in Figure 3. The Sin and Exponential kernel functions present very similar performance, and their results are more accurate than the result using Hard-limit kernel function. The result using sigmoid kernel function is the most accurate in this experiment.

In the third experiment, we present the classification margin variation for different values of ν used in ν-OELM. The role of ν is the tradeoff between a large classification margin and classifier error. A high error penalty will force the training of ν-OELM model to avoid classification errors. The influence of ν is studied in Figure 4. When ν is 0.1, the margin is large and the number of support vectors is a small percentage of the number of training points. When ν is 1.0, the margin is small and most of the training points fall out of the margin. When ν is larger than 0.6, the training model is going toward overfitting. That is, by decreasing the ν, training error is decreasing but testing error is

increasing.

Note that in OELM a large C will result in a large search space for the quadratic programming optimizer, which generally increases the cost of the quadratic programming search. However, this phenomenon would not appear in ν-OELM, because the search space of parameter ν is small.

(a) (b)



(c) (d)

Figure 3 Comparison results of the Sigmoid, Sin, Hard-limit and Exponential kernel function on the artificial data set. The middle lines of the above graphs represent the classification hyperplane. Those black marked points are support vectors. (a) Sigmoid kernel function-81 classification errors, (b) Sin kernel function-82 classification errors, (c) Hard-limit kernel function-86 classification errors, (d) Exponential kernel function-82 classification errors.

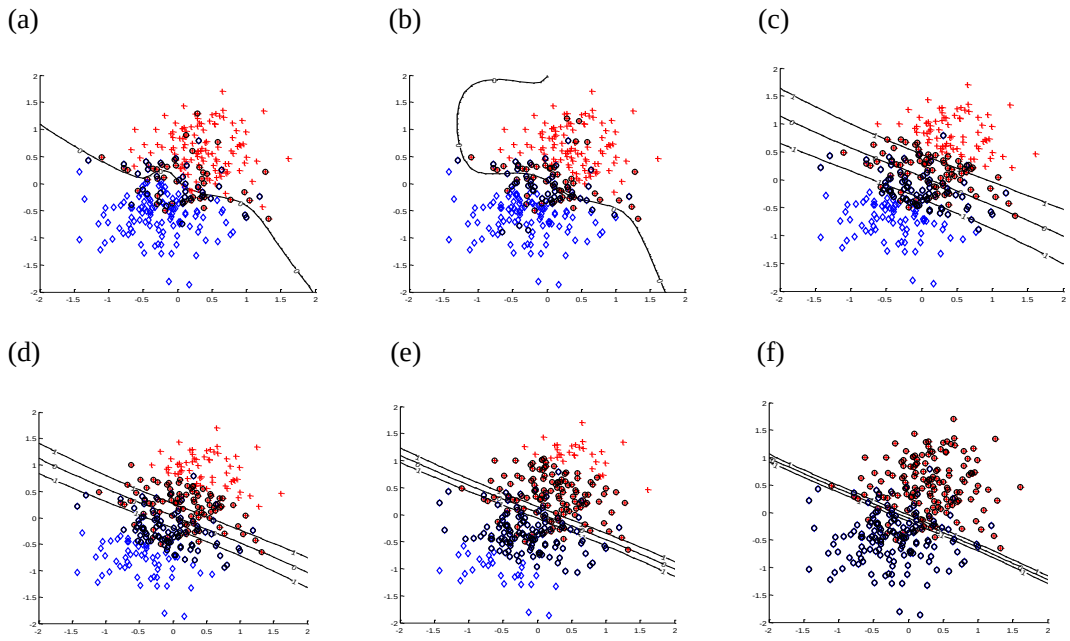


Figure 4 Comparison results of different ν of OELM on the artificial data set (a) ν=0.1 and testing accuracy is 83.67%, (b) ν=0.3 and testing accuracy is 87.67%, (c) ν=0.5 and testing accuracy is 88.11%, (d) ν=0.6 and testing accuracy is 88.11%, (e) ν=0.8 and testing accuracy is 88.00%, (f) ν=1 and testing accuracy is 87.44%.

5.3 Real world data set

In this section, the proposed method is evaluated on the 11 real world benchmark problems, as listed in Table 2. We show that the performance of ν-OELM can be comparable to other state-of-the-art learning methods, namely SVM, ν-SVM, and OELM. In our experiments, for each evaluated classifier, we use one training/testing partition to do parameter selection. The selected parameter combination is then used for other partitions. In our binary classification applications, the inputs (attributes) are normalized into the range [0, 1] while the outputs (targets) are normalized into the range [-1, 1].

Table 2 Specification of benchmark data sets

Data sets	Sources	# Training Samples	# Testing Samples	# attributes
Heart	UCI [24]	70	200	13
Pwlinear	UCI [24]	100	100	10
Sonar	UCI [24]	100	158	60
Liver-disorders	UCI [24]	200	145	6
Ionosphere	UCI [24]	100	251	34
Breast-cancer	UCI [24]	300	383	10
Australian	UCI [24]	300	390	14
Pimadata	UCI [24]	400	368	8
Monk's Problem 1	UCI [24]	124	432	6
Monk's Problem 2	UCI [24]	169	432	6
A1a	JP98a [25]	1605	30956	123

The generalization performance of four algorithms (SVM, ν-SVM, OELM and ν-OELM) on the 'Monk's problem 1' data set for different parameter combination discussed in section 5.1 is presented in Figure 5. It is easy to see that in Figure 5 the best generalization performance of SVM and ν-SVM depends heavily on the combinations of $(C,\gamma)/(\nu,\gamma)$. The best generalization performance is usually achieved in a narrow range of such combinations. In contrast, generalization performance of OELM and ν-OELM is less sensitive to the variation of parameter L. Moreover, when parameter ν is fixed, ν-OELM seems more stable than OELM. Such insensitivity of ν-OELM on parameter L makes the implementation of ν-OELM worthy.

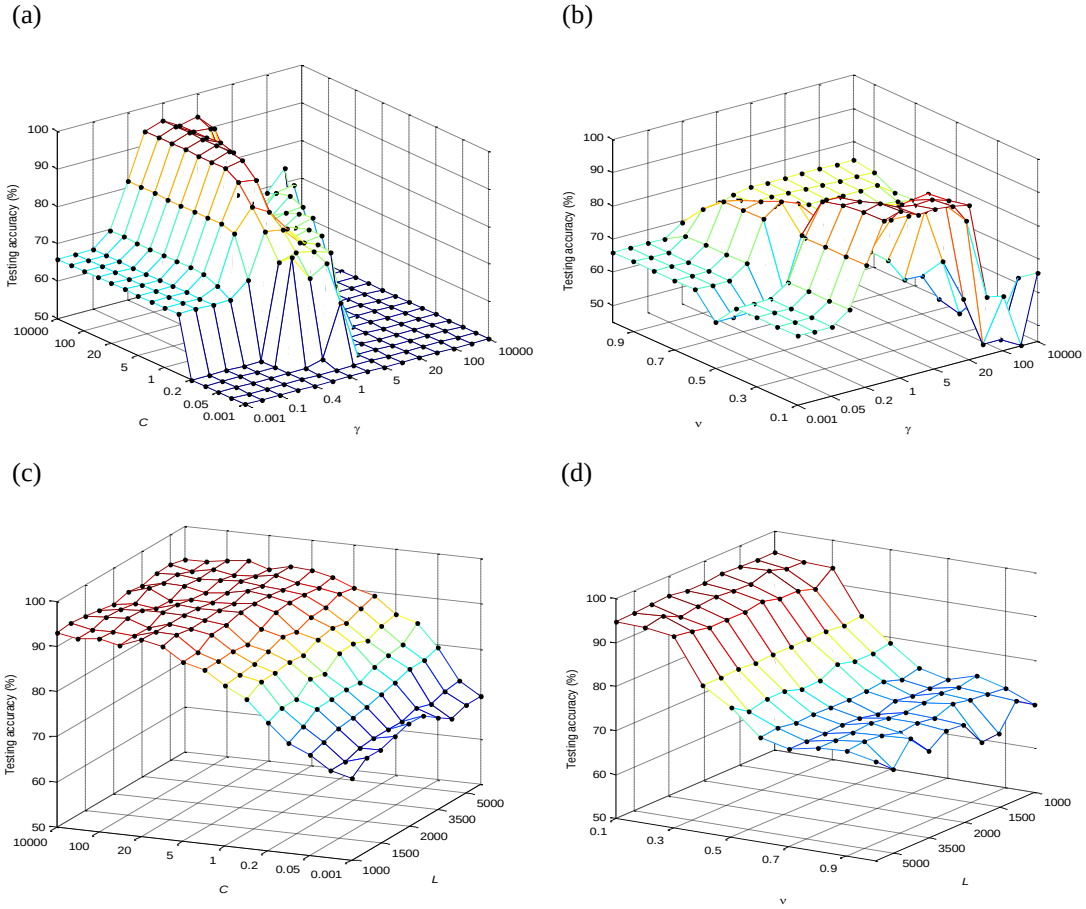


Figure 5 Generalization performance of four classifiers on Monk's problem 1 data set for different combinations of parameters. (a) SVM, (b) ν-SVM, (c) OELM, (d) ν-OELM.

In the second experiment, we perform statistical tests [26] to compare ν-OELM with that of three standard algorithms. For 11 benchmark data sets where random trial is performed, we apply ten random trials paired t-test with 9 degrees of freedom and 95% significance level. Ten random trials have been conducted for each problem with training and testing samples randomly generated for each trial. Experimental results include the testing accuracy statistics, the training time (TT), selection of parameter combinations, and the number of resulted support vectors (SVs). The selected combinations of parameters are listed in Table 3.

Table 4 reports the testing accuracy statistics using the combinations shown in Table 3. The bold font figures denotes the best performance across the algorithms compared. Based on the paired t-test, ν-OELM performs better than other standard algorithms for 7 data sets. The results demonstrate that

ν-OELM achieves better performance when compared with OELM for 8 data sets. ν-SVM performs good performance on 1 data set, while SVM shows its advantage also on 1 data set.

Table 5 lists the average training time and the number of resulted support vectors for the benchmark data sets. It can be observed that the training time of SVM is less than other classifiers over all classification problems, while it does not include the time for searching the best parameter combinations. For support vectors, the number of SVs in ν-OELM is smaller for some problems, but larger for some others. The situation highly depends on the chosen parameters.

Table 3 Parameters setting conducted by four classifiers on the benchmark data sets

Data set	SVM	ν-SVM	OELM	ν-OELM
	(C,γ)	(ν,γ)	C	ν
Heart	$(10^4, 5)$	(0.5, 2)	50	0.5
Pwlinear	$(10^4, 10^3)$	(0.6, 10)	10^{-3}	0.3
Sonar	(20, 1)	(0.3, 0.8)	10^4	0.5
Liver-disorders	(10, 2)	(0.7, 0.8)	10	0.6
Ionosphere	(5, 5)	(0.2, 50)	0.1	0.7
Breast-cancer	(5, 50)	(0.6, 10)	10^{-3}	0.2
Australian	(2, 2)	(1, 0.8)	0.1	0.7
Pimadata	$(10^3, 50)$	(0.6, 10)	10^{-2}	0.6
Monk's Problem 1	(10, 1)	(0.1, 1)	10^4	0.1
Monk's Problem 2	$(10^4, 5)$	(0.2, 5)	10^2	0.1
A1a	(2, 5)	(0.3, 2)	10^{-2}	0.4

Table 4 Performance comparison of ν-OELM with three standard algorithms

Data set	SVM (%)	ν-SVM (%)	OELM (%)	ν-OELM (%)
Heart	75.90±1.31	74.35±1.09	75.75±1.15	**76.00±0.95**
Pwlinear	83.20±2.04	81.90±2.21	84.40±1.79	**86.10±1.35**
Sonar	84.35±2.45	**84.63±1.70**	79.08±2.02	78.80±2.44
Liver-disorders	70.55±1.54	69.93±0.88	**72.55±1.57**	72.41±1.03
Ionosphere	**91.99±0.93**	91.06±0.85	89.33±1.06	90.13±0.93
Breast-cancer	93.63±0.80	94.10±0.86	96.42±0.50	**96.58±0.55**
Australian	68.15±0.78	68.15±0.80	68.02±0.83	**68.20±0.86**
Pimadata	76.85±1.04	76.74±1.47	77.17±1.18	**77.23±1.06**
Monk's Problem 1	91.69±1.67	91.69±1.67	94.37±1.05	**94.65±1.16**
Monk's Problem 2	78.17±2.02	81.23±1.98	85.04±1.56	**85.07±1.50**
A1a	83.49±0.32	81.23±1.98	**83.67±0.26**	83.46±0.34

Table 5 The training time and number of the support vectors conducted by four classifiers on the benchmark data sets

Data sets	SVM		ν-SVM		OELM		ν-OELM	
	TT (s)	SVs	TT (s)	SVs	TT (s)	SVs	TT (s)	SVs
Heart	0.0248	40	0.0335	38	0.0267	13	0.0313	43
Pwlinear	0.0268	53	0.0343	60	0.0315	10	0.0328	39
Sonar	0.0273	69	0.0463	76	0.0252	42	0.0367	43
Liver-disorders	0.0787	163	0.1865	154	0.0593	135	0.1197	129
Ionosphere	0.0295	55	0.1479	45	0.0293	28	0.0353	68
Breast-cancer	0.0860	194	0.0956	177	0.0868	94	0.0689	66
Australian	0.1421	94	0.1504	102	0.1948	202	0.1492	213
Pimadata	0.1665	208	0.2927	238	0.3657	213	0.1902	243
Monk's Problem 1	0.0311	68	0.0457	70	0.0548	39	0.0420	45
Monk's Problem 2	0.0705	79	0.0791	66	0.0792	75	0.0812	76
A1a	4.2829	663	13.876	812	5.2387	670	5.0108	656

5.4 DNA microarray data sets

In this experiment, we test the performance of ν-OELM with other three classifiers on two DNA microarray data sets. The leukemia data set [27] consists of 72 tissue samples, each with 7129 features (gene expression measurements). The samples include 47 ALL (acute lymphoblastic leukemia) and 25 AML (acute myeloid leukemia). The original data set have been separated into a training set of 38 samples and a testing set of 34 samples. The Colon cancer data set [28] contains 62 tissue samples, each with 2000 features. The tissue samples include 22 normal and 40 colon cancer cases. The training time (TT) and testing accuracy (TR) are reported in Table 6. From Table 6, ν-OELM generally achieves the best performance. Moreover, the computational cost of ν-OELM is significantly smaller than other standard algorithms for two DNA microarray data sets.

Table 6 Comparison between SVM, ν-SVM, OELM and ν-OELM on gene classification problems

Data sets	SVM		ν-SVM		OELM		ν-OELM	
	TT(s)	TR(%)	TT(s)	TR(%)	TT(s)	TR(%)	TT(s)	TR(%)
Leukemia	1.5853	82.35	1.5732	82.35	0.1493	82.35	**0.0219**	82.35
Colon	0.1057	81.25	0.2013	78.13	0.0254	84.38	**0.0153**	84.38

6. Conclusions

In this paper, we have extended the traditional OELM classifier to a new optimization problem regularized by parameter ν. The role of ν can be easily interpreted by the optimization problem. A small ν leads to large emphasis on making small number of outliers. Generally speaking, OELM and ν-OELM are equivalent regarding their classification power. The performance of ν-OELM is evaluated using both artificial and real data sets, and the experimental results demonstrate that the performance of ν-OELM is comparable to other learning classifiers like SVM, ν-SVM and OELM. Moreover, it is

shown that the training time of ν-OELM is much shorter than that by using other classifiers on high dimension classification applications.

Both OELM and ν-OELM has only one tuning parameter, which is selected by minimizing validation error. However, parameter tuning of ν-OELM is easier than OELM, because the best generalization performance can be achieved by selecting parameter from only few candidates. Thus, compared to classical algorithms like SVM or OELM, users can use ν-OELM easily and effectively by avoiding time-consuming parameter tuning.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. 61402426), the National Natural Science Foundation of China (Grant No. 51405327), and the Natural Science Foundation for Young Scientists of Shanxi Province, China (Grant No. 2014021024-1).

References

[1] Huang G-B, Zhu Q-Y, Siew C-K. Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of international joint conference on neural networks (IJCNN2004), vol. 2, (Budapest, Hungary); 2004. p. 985–990,25–29 July.

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

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

[4] G.-B. Huang, Q. -Y, Zhu, K. -Z. Mao, C.-K. Siew, P. Saratchandran, and N. Sundararajan. Can Threshold Networks be Trained Directly? IEEE Transactions on Circuits and Systems-II: Express Briefs, 2006, 53(3): 187-191.

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

[6] A. Basu, H. M. Zhou, M. H Lim and G.-B. Huang. Silicon spiking neurons for hardware implementation of extreme learning machines. Neurocomputing, 2013, 102: 125-134.

[7] Cambria, E., Gastaldo, P., Bisio, F., and Zunino, R. An ELM-based model for affective analogical reasoning. Neurocomputing, 2015, 149: 443-455.

[8] W.-W, Zong, and G.-B. Huang. Face recognition based on extreme learning machine, Neurocomputing, 2011, 74: 2451-2551.

[9] Alexandros I, Anastastios T, and Ioannis P. On the kernel Extreme Learning Machine classifier. Pattern Recognition Letters, 2015, 54:11-17.

[10] Huang G-B, Ding X, Zhou H. Optimization method based extreme learning machine for classification. Neurocomputing, 2010, 74:155-163.

[11] Huang G-B, Zhou H, Ding X, and Zhang R. Extreme Learning Machine for Regression and Multiclass Classification. IEEE Transaction on System, Man, and Cybernetics-Part B: Cybernetics, 2012, 42(2): 513-529.

[12] Ding X, and Chang B. Active set strategy of optimized extreme learning machine, Chinese Science Bulletin, 2014, 59(31): 4152-4160.

[13] Ding X, and Lei M. Optimization ELM Based on Rough Set for Predicting the Label of Military Simulation Data. Mathematical Problems in Engineering, 2014, 2014: 1-8.

[14] Huang G-B. An Insight into Extreme Learning Machines: Random Neurons, Random Features and Kernels. Cognitive Computation, 2014, 6(3): 376-390.

[15] P. L. Bartlett, The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network, IEEE Transactions on Information Theory, 1998, 44(2):525-536.

[16] C. Cortes, V. Vapnik. Support vector networks. Machine Learning, 1995(20):273-297.

[17] C. J. C. Burges. A tutorial on support vector machines forpattern recognition. Data Mining and Knowledge Discovery. 1998, 2(2):121-167.

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

[19] Schölkopf B., A. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. Neural Computation, 2000, 12:1207-1245.

[20] R. Fletcher, Practical Methods of Optimization: Volume2 Constrained Optimization, Wiley, New York, 1981.

[21] C. C. Chang and C. J. Lin, "Training ν-support vector classifiers: Theory and algorithms, Neural Computation, vol. 13, pp. 2119–2147, 2001.

[22] S. Canu, Y. Grandvalet, V. Guigue, A. Rakotomamonjy, SVM and kernel methods matlab toolbox, http://asi.insa-rouen.fr/enseignants/arakotom/toolbox/index.html, Perception System set Information, INSA de Rouen, Rouen, France,2005.

[23] P. Ghanty, S. Paul, N. R. Pal, NEUROSVM: an architecture to reduce the effect of the choice of kernel on the performance of svm, Journal of Machine Learning Research, 2009:10:591-622.

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

[25] J.C. Platt, Fast training of support vector machines using sequential minimal optimization, Advances in kernel methods: support vector learning table of contents, 1999:185-208.

[26] T. G. Dietterich. Approximation statistical tests for comparing supervised classification learning algorithms. Neural Computation, 1998: 10(7):1895-1923.

[27] TR Golub, DK Slonim, P. Tamayo, et al. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring, Science, 1999, 286(5439):531.

[28] U. Alon, N. Barkai, D. A. Notterman, et al., Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays, Cell Biology, 1999, 96(12): 6745-6750.

Xiao-jian Ding received his Ph.D degree in the department of Computer Science and Technology in Xi'an Jiaotong University, China, in 2011. After that, he joined Science and Technology on Information Systems Engineering Laboratory, Nanjing, China. Currently, he is a senior engineer of the the same Laboratory. His research interests include machine learning and pattern recognition.

Yuan Lan received the B.Eng. Degree in electrical and electronic engineering from Nanyang Technological University, Singapore, in 2006, and the Ph.D. degree in the same university in 2011. From 2010 to 2011, she was a researcher in electrical and electronic engineering of Nanyang Technological University, Singapore. She joined the Qiito Pte Ltd as a web scientist from Jan 2012 to Dec 2012 in Singapore. Since June 2013, she has worked in Key Laboratory of Ministry of Education in Advance Transducers and Intelligent Control System, and in School of Mechanical Engineering, Taiyuan University of Technology, Taiyuan, China. Now she works as an Associate Professor, and her research interests are on mechanical fault diagnosis, signal processing, machine learning and data mining.

Zhi-feng Zhan received his M.S. degree in Computer Application Technology from Xi'an University of Technology in 2006. He is currently an associate professor at Software Engineering College, Zheng Zhou University of Light Industry. His current research interests include image processing and pattern recognition.

Xin Xu received her Ph.D degree in School of Computing from National University of Singapore in 2006. She is currently an Senior Research Engineer in Science and Technology on Information System Engineering Laboratory. Her research interests are in the area of data mining and data fusion.