HHS Public Access

Author manuscript

IEEE Trans Cybern. Author manuscript; available in PMC 2016 May 27.

Published in final edited form as:

IEEE Trans Cybern. 2014 October ; 44(10): 1858–1870. doi:10.1109/TCYB.2014.2298235.

Sparse Extreme Learning Machine for Classification

Zuo Bai,
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798

Guang-Bin Huang,
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798

Danwei Wang,
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798

Han Wang, and
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798

M. Brandon Westover
Massachusetts General Hospital and Harvard Medical School, Boston, MA 02114 USA

Zuo Bai: zbai1@e.ntu.edu.sg; Guang-Bin Huang: egbhuang@ntu.edu.sg; Danwei Wang: edwwang@ntu.edu.sg; Han Wang: hw@ntu.edu.sg; M. Brandon Westover: mwestover@mgh.harvard.edu

Abstract

Extreme learning machine (ELM) was initially proposed for single-hidden-layer feedforward neural networks (SLFNs). In the hidden layer (feature mapping), nodes are randomly generated independently of training data. Furthermore, a unified ELM was proposed, providing a single framework to simplify and unify different learning methods, such as SLFNs, least square support vector machines, proximal support vector machines, and so on. However, the solution of unified ELM is dense, and thus, usually plenty of storage space and testing time are required for large-scale applications. In this paper, a sparse ELM is proposed as an alternative solution for classification, reducing storage space and testing time. In addition, unified ELM obtains the solution by matrix inversion, whose computational complexity is between quadratic and cubic with respect to the training size. It still requires plenty of training time for large-scale problems, even though it is much faster than many other traditional methods. In this paper, an efficient training algorithm is specifically developed for sparse ELM. The quadratic programming problem involved in sparse ELM is divided into a series of smallest possible sub-problems, each of which are solved analytically. Compared with SVM, sparse ELM obtains better generalization performance with much faster training speed. Compared with unified ELM, sparse ELM achieves similar generalization performance for binary classification applications, and when dealing with

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

Authors' photographs and biographies not available at the time of publication.

large-scale binary classification problems, sparse ELM realizes even faster training speed than unified ELM.

Index Terms

Classification; extreme learning machine (ELM); quadratic programming (QP); sequential minimal optimization (SMO); sparse ELM; support vector machine (SVM); unified ELM

I. Introduction

Extreme learning machine (ELM) was initially proposed for single-hidden-layer feedforward neural networks (SLFNs) [1]–[3]. Extensions have been made to generalized SLFNs, which may not be neuron alike, including SVM, polynomial networks, and traditional SLFNs [4]–[11]. For the initial ELM implementation

$$f(\mathbf{x})=\mathbf{h}(\mathbf{x})\boldsymbol{\beta} \quad (1)$$

where $\mathbf{x} \in \mathbf{R}^d$, $\mathbf{h}(\mathbf{x}) \in \mathbf{R}^{1\times L}$, $\boldsymbol{\beta} \in \mathbf{R}^L$. $f(\mathbf{x})$ is the output; $\mathbf{x}$ is the input; $\mathbf{h}(\mathbf{x})$ is the hidden layer; and $\boldsymbol{\beta}$ is the weight vector between the hidden nodes and output node. Hidden nodes are randomly generated and $\boldsymbol{\beta}$ is analytically calculated, trying to reach the smallest training error and the smallest norm of output weights. It has been shown that ELM can handle regression and classification problems efficiently.

Support vector machine (SVM) and its variants, such as least square SVM (LS-SVM), proximal SVM (PSVM), have been widely used for classification in the past two decades due to their good generalization capability [12]–[14]. In SVM, input data are first transformed into a higher dimensional space through a feature mapping ($\varphi : \mathbf{x} \rightarrow \varphi(\mathbf{x})$). Optimization method is used to find the optimal separating hyperplane. From the perspective of network architecture, SVM is a specific type of SLFN referred to as support vector network in [12]. The hidden nodes are $K(x, \mathbf{x}_s)$, and the output weight is $[a_1 t_1, \cdots, a_{N_s} t_{N_s}]^T$. $\mathbf{x}_s$ is the s-th support vector and a_s, t_s are respectively Lagrange variable and class label of x_s.

In the conventional SVM [12], primal problem is constructed with inequality constraints, leading to a quadratic programming (QP) problem. The computation is extremely intensive, especially for large-scale problems. Thus, variants such as LS-SVM [14] and PSVM [13] have been suggested. In LS-SVM and PSVM, equality constraints are utilized. The solution is generated by matrix inversion, reducing computational complexity significantly. However, the sparsity of the network is lost, resulting in a more complicated network with more storage space and longer testing time.

Many works have been done since the initial ELM [3]. In [15], a unified ELM was proposed, in which both kernels and random hidden nodes can work for the feature mapping. It provides a unified framework to simplify and unify different learning methods, including LS-SVM, PSVM, feedforward neural networks, and so on. However, the sparsity is lost as equality constraints like LS-SVM/PSVM are used.

In [16], an optimization method based ELM was proposed for classification. And as inequality constraints are adopted, a sparse network is constructed. However, [16] only discusses random hidden nodes as the feature mapping although kernels can be used as well.

In this paper, a comprehensive sparse ELM is proposed, in which both kernels and random hidden nodes work. Furthermore, it is shown that sparse ELM also unifies different learning theories of classification, including SLFNs, SVM and RBF networks. Compared with unified ELM, a more compact network is provided by the proposed sparse ELM, which reduces storage space and testing time.

Furthermore, a specific training algorithm is developed in this paper. Comparing to SVM, sparse ELM does not have the constraint

$$\sum_{i=1}^{N}\alpha_i t_i=0$$

in the dual problem. Thus, sparse ELM searches for the optimal solution in a wider range than SVM does. Better generalization performance is expected.

Inspired by sequential minimal optimization (SMO), which is one of the easiest implementations of SVM [17], the large QP problem of sparse ELM is also divided into a series of smallest possible sub-QP problems. In SMO, each subproblem includes two Lagrange variables (α_i's), because the sum constraint

$$\sum_{i=1}^{N}\alpha_i t_i=0$$

should be satisfied all the times. However, in sparse ELM, each sub-problem only needs to update one α_i as the sum constraint has vanished. Sparse ELM is based on iterative computation, while unified ELM is based on matrix inversion. Thus, when dealing with large problems, the training speed of sparse ELM could be faster than that of unified ELM. Consequently, sparse ELM is promising for growing-scale problems due to its faster training and testing speed, and less storage space.

The paper is organized as follows. In Section II, we give a brief introduction to SVM and its variants. Inequality and equality constraints would respectively lead to sparse and dense networks. In Section III, former works about ELM are briefed, including initial ELM with random hidden nodes and unified ELM. In Section IV, a sparse ELM for classification is proposed and proved to unify several classification methods. In Section V, the training algorithm for sparse ELM is presented, including optimality conditions, termination, convergence analysis and etc. In Section VI, the performance comparison between sparse ELM, unified ELM and SVM is conducted over some benchmark data sets.

II. Briefs of SVM and variants

The conventional SVM was proposed by Cortes and Vapnik for classification [12], and it was considered as a specific type of SLFNs. Some variants have been suggested for fast implementation, regression or multiclass classification [13], [14], [18]–[21]. There are two main stages in SVM and its variants.

A. Feature mapping

Given a training data set $(\mathbf{x}_i, t_i)$, $i = 1, \cdots, N$, $\mathbf{x}_i \in \mathbf{R}^d$ and $t_i \in \{-1, 1\}$. Normally, it is non-separable in the input space. Thus, a nonlinear feature mapping is needed

$$\phi: x_i \rightarrow \phi(x_i). \quad (2)$$

B. Optimization

Optimization method is used to find the optimal hyperplane, which maximizes the separating margin and minimizes the training errors at the same time. Inequality and equality constraints could be used.

1) Inequality Constraints—In conventional SVM, inequality constraints are used to construct the primal problem

$$\begin{aligned} &\text{Minimize: } L_p = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ &\text{Subject to: } t_i(\mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x}_i)+b) \geq 1-\xi_i \quad \xi_i \geq 0, \quad i=1,\cdots,N \end{aligned} \quad (3)$$

where C is a user-specified parameter that controls the trade-off between maximal separating margin and minimal training errors.

According to KKT theorem [22], the primal problem could be solved through its dual form

$$\begin{aligned} &\text{Minimize: } L_d = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j t_i t_j K(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i \\ &\text{Subject to: } \sum_{i=1}^{N}\alpha_i t_i = 0 \quad 0 \leq \alpha_i \leq C, \quad i=1,\ldots,N \end{aligned} \quad (4)$$

where kernel $K(u, v) = \varphi(u) \cdot \varphi(v)$ is often used since dealing with φ explicitly is sometimes quite difficult. Kernel K should satisfy Mercer's conditions [12].

2) Equality Constraints—In LS-SVM and PSVM, equality constraints are used. After optimization, the dual problem is a set of linear equations. Solution is obtained by matrix inversion. The only difference between LS-SVM and PSVM is about how to use bias b. As will be elaborated later that bias b is discarded in sparse ELM, we only need to review either of them. At here, we take PSVM as an example.

The primal problem is

$$\begin{aligned}&\text{Minimize: } L_p=\frac{1}{2}(\|w\|^2+b^2)+C\frac{1}{2}\sum_{i=1}^{N}\xi_i^2\\&\text{Subject to: } t_i(w\cdot\phi(x_i)+b)=1-\xi_i,\quad i=1,\cdots,N.\end{aligned} \tag{5}$$

The final solution is

$$\left(\frac{I}{C}+G+TT^T\right)\alpha=1 \tag{6}$$

in which, $G_{i,j} = t_i t_j K(x_i, x_j)$.

For both cases, the decision function is

$$f(\mathbf{x})=\text{sign}\left(\sum_{s=1}^{N_s}\alpha_s t_s K(\mathbf{x},\mathbf{x}_s)+b\right) \tag{7}$$

where $\mathbf{x}_s$ is support vector (SV) and N_s is the number of SVs.

For conventional SVM which has inequality constraints, many Lagrange variables (α_i's) are zero. Thus, a sparse network is provided. However, for LS-SVM/PSVM, almost all Lagrange variables are non-zero. Thus, the network is dense, requiring more storage space and testing time.

III. Introduction of ELM

ELM was first proposed by Huang *et al.* [1], [3] for SLFNs, and then extended to generalized SLFNs [4]–[7]. Its universal approximation ability has been proved in [2]. In [15], a unified ELM was proposed, providing a single framework for different networks and different applications.

A. Initial ELM with Random Hidden Nodes

In the initial ELM, hidden nodes are generated randomly and only the weight vector between hidden and output nodes needs to be calculated [3]. Much fewer parameters need to be adjusted than traditional SLFNs, and thus the training can be much faster.

Given a set of training data ($\mathbf{x}_i$, t_i), $i = 1, \cdots, N$. ELM could have single or multiple output nodes. For simplicity, we introduce the case with single output node. **H** and **T** are respectively hidden layer matrix and output matrix

$$\mathbf{H}=\begin{bmatrix}\mathbf{h}(\mathbf{x}_1)\\\mathbf{h}(\mathbf{x}_2)\\\vdots\\\mathbf{h}(\mathbf{x}_N)\end{bmatrix},\ \mathbf{T}=\begin{bmatrix}t_1 & t_2 & \cdots & t_N\end{bmatrix}^T \tag{8}$$

$$\mathbf{H}\boldsymbol{\beta}=\mathbf{T}.$$

The essence of ELM is that the hidden nodes of SLFNs can be randomly generated. They can be independent of the training data. The output weight β can be obtained in different ways [2], [3], [23]. For example, a simple way is to obtain the following smallest norm least-squares solution [3]

$$\boldsymbol{\beta}=\mathbf{H}^{\dagger}\mathbf{T} \quad (9)$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of $\mathbf{H}$.

B. Unified ELM

Liu *et al.* [5] and Frenay and Verleysen [7] suggested to replace SVM feature mapping with ELM random hidden nodes (or normalized random hidden nodes). In this way, feature mapping would be known to users and explicitly dealt with. However, except for feature mapping, all constraints, bias b, calculation of weight vector and training algorithm are the same with SVM. Thus, only comparable performance and training speed are achieved.

In [15], a unified ELM is proposed for regression and classification, combining different kinds of networks together, such as SLFNs, LS-SVM and PSVM. As proved in [2], ELM has universal approximation ability. Thus, the separating hyperplane tends to pass through the origin in the feature space and bias b can be discarded.

Similar to the initial ELM, unified ELM could have single or multiple output nodes. For the single output node case

$$\begin{aligned}&\text{Minimize: } L_p=\tfrac{1}{2}\|\boldsymbol{\beta}\|^2+C\tfrac{1}{2}\sum_{i=1}^{N}\xi_i^2\\&\text{Subject to: } \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta}=t_i-\xi_i,\quad i=1,\cdots,N.\end{aligned} \quad (10)$$

The output function of the unified ELM is

$$\begin{aligned}\mathbf{f}(\mathbf{x})&=\mathbf{h}(\mathbf{x})\boldsymbol{\beta}=\mathbf{h}(\mathbf{x})\mathbf{H}^T\left(\frac{\mathbf{I}}{C}+\mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T} \text{ or}\\&=\mathbf{h}(\mathbf{x})\left(\frac{\mathbf{I}}{C}+\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}.\end{aligned} \quad (11)$$

1) Random Hidden Nodes—The hidden nodes of SLFNs can be randomly generated, resulting in random feature mapping h(x), which is explicitly known to users.

2) Kernel—When the hidden nodes are unknown, kernels satisfying Mercer's conditions could be used

$$\boldsymbol{\Omega}_{\text{ELM}}=\mathbf{H}\mathbf{H}^T:\Omega_{\text{ELM}}(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{h}(\mathbf{x}_i)\mathbf{h}(\mathbf{x}_j)^T=K(\mathbf{x}_i,\mathbf{x}_j) \quad (12)$$

where $\boldsymbol{\Omega}_{\text{ELM}}$ is called ELM kernel matrix.

In unified ELM, most errors ξ_i's are non-zero (positive or negative) to make the equality constraint $\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} = t_i - \xi_i$ satisfied for all training data. As Lagrange variables α_i's are proportional to corresponding ξ_i's ($\alpha_i = C\xi_i$), almost all α_i's are non-zero. Therefore, unified ELM provides a dense network and requires more storage space and testing time comparing to a sparse one.

IV. Sparse ELM for Classification

In [16], an optimization method based ELM was first proposed for classification. A sparse network is obtained as inequality constraints are used. However, only the case where random hidden nodes are used as the feature mapping is studied in details.

In this section, we present a comprehensive sparse ELM, in which both kernels and random hidden nodes work. In addition, we show that sparse ELM also unifies different classification methods, including SLFNs, conventional SVM, and RBF networks.

A. Problem Formulation

1) Feature Mapping—At first, a feature mapping from input space to a higher dimensional space is needed. It could be randomly generated. When the feature mapping $\mathbf{h}(\mathbf{x})$ is not explicitly known or inconvenient to use, kernels also apply as long as satisfying Mercer's conditions.

2) Optimization—As proved in [15], given any disjoint regions in $\mathbf{R}^d$, there exists a continuous function $f(\mathbf{x})$ being able to separate them. ELM has universal approximation capability [2]. In other words, given any target function $f(\mathbf{x})$, there exist β_i's

$$\lim_{L\to+\infty}\|\sum_{i=1}^{L}\beta_i h_i(\mathbf{x})-f(\mathbf{x})\|=0. \quad (13)$$

Thus, bias b as in conventional SVM is not required. However, the number of hidden nodes L cannot be infinite in real implementation. Hence, training errors ξ_i's should be allowed.

Overfitting could be well solved by minimizing both empirical errors $\left(\sum_{i=1}^{N}\xi_i\right)$ and structural risks ($\|\boldsymbol{\beta}\|^2$) based on theories of statistical learning [24], and a great generalization performance would be presented.

Inequality constraints are used, and the primal problem is constructed as follows:

$$\begin{aligned}&\text{Minimize: } L_p=\tfrac{1}{2}\|\boldsymbol{\beta}\|^2+C\sum_{i=1}^{N}\xi_i\\&\text{Subject to: } t_i\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta}\geq 1-\xi_i \quad \xi_i\geq 0, \quad i=1,\ldots,N.\end{aligned} \quad (14)$$

The Lagrange function is

$$P(\boldsymbol{\beta},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\mu})=\frac{1}{2}\|\boldsymbol{\beta}\|^2+C\sum_{i=1}^{N}\xi_i-\sum_{i=1}^{N}\mu_i\xi_i-\sum_{i=1}^{N}\alpha_i\cdot(t_i\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta}-(1-\xi_i)). \quad (15)$$

At the optimal solution, we have

$$\begin{gathered}\frac{\partial P}{\partial\boldsymbol{\beta}}=0\Rightarrow\boldsymbol{\beta}=\sum_{i=1}^{N}\alpha_i t_i\mathbf{h}(\mathbf{x}_i)^T=\sum_{s=1}^{N_s}\alpha_s t_s\mathbf{h}(\mathbf{x}_s)^T\\ \frac{\partial P}{\partial\boldsymbol{\xi}}=0\Rightarrow C=\alpha_i+\mu_i.\end{gathered} \quad (16)$$

Substitute the results of (16) into (15), we obtain the dual form of sparse ELM

$$\begin{gathered}\text{Minimize: } L_d=\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j t_i t_j\Omega_{\text{ELM}}(\mathbf{x}_i,\mathbf{x}_j)-\sum_{i=1}^{N}\alpha_i\\ \text{Subject to: } 0\le\alpha_i\le C,\quad i=1,\ldots,N\end{gathered} \quad (17)$$

where $\boldsymbol{\Omega}_{\text{ELM}}$ is the ELM kernel matrix

$$\Omega_{\text{ELM}}(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{h}(\mathbf{x}_i)\mathbf{h}(\mathbf{x}_j)^T=K(\mathbf{x}_i,\mathbf{x}_j). \quad (18)$$

Therefore, the output of sparse ELM is

$$\begin{aligned}f(\mathbf{x})&=\mathbf{h}(\mathbf{x})\boldsymbol{\beta}=\mathbf{h}(\mathbf{x})\left(\sum_{i=1}^{N}\alpha_i t_i\mathbf{h}(\mathbf{x}_i)^T\right)\\ &=\mathbf{h}(\mathbf{x})\left(\sum_{s=1}^{N_s}\alpha_s t_s\mathbf{h}(\mathbf{x}_s)^T\right)=\sum_{s=1}^{N_s}\alpha_s t_s\Omega_{\text{ELM}}(\mathbf{x},\mathbf{x}_s)\end{aligned} \quad (19)$$

where $\mathbf{x}_s$ is support vector (SV), and N_s is the number of SVs.

B. Sparsity Analysis

KKT conditions are

$$\begin{gathered}\alpha_i\,(t_i\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta}-(1-\xi_i))=0\\ \mu_i\xi_i=0.\end{gathered} \quad (20)$$

Lagrange variables of SVs are non-zero. There exist two possibilities.

1. $0<\alpha_i<C$

$$\begin{gathered}\mu_i>0\Rightarrow\xi_i=0\\ \alpha_i>0\Rightarrow t_i\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta}-1=0.\end{gathered} \quad (21)$$

   In this case, the data is on the separating boundary.

2. $\alpha_i=C$

$$\begin{aligned} &\mu_i=0 \Rightarrow \xi_i>0 \\ &\alpha_i>0 \Rightarrow t_i\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta}-(1-\xi_i)=0 \\ &\Rightarrow t_i\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta}-1<0. \end{aligned} \quad (22)$$

In this case, the data is classified with error.

Remarks: Different from unified ELM, in which most errors ξ_i's are non-zero, in sparse ELM, errors ξ_i's are non-zero only when the inequality constraint $t_i\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta}-1>0$ are not met.

Considering the general distribution of all training data for sparse ELM, only a part of them would be on the boundary or classified with errors. Thus, only some training data are SVs.

As seen from Fig. 1, sparse ELM provides a compact dual network as non-SVs are excluded. For the primal network, it remains the same because the number of hidden nodes L is fixed once chosen. However, sparsity also simplifies the computation of $\boldsymbol{\beta}$ as fewer components exist as in (16). Hence, less computation is required in the testing phase. In addition, only SVs and corresponding Lagrange variables need to be stored in memory. Consequently, compared with unified ELM, sparse ELM needs less storage space and testing time.

C. Unified Framework for Different Learning Theories

As observed from Fig. 1, the primal network of sparse ELM shares the same structure with generalized SLFNs. And the dual network of sparse ELM is as the same as the dual of SVM (support vector network) [12]. In addition, both RBF kernels and RBF hidden nodes can be used in sparse ELM. Therefore, sparse ELM provides a unified framework for different learning theories of classification, including traditional SLFNs, conventional SVM and RBF networks.

D. ELM Kernel Matrix $\Omega_{\mathbf{ELM}}$

Similar to the unified ELM [15], sparse ELM can use random hidden nodes or kernels. For the sake of readability, we present them in the following.

1) Random Hidden Nodes—Ω_{ELM} is calculated from random hidden nodes directly

$$\mathbf{h}(\mathbf{x})=[G(\mathbf{a}_1,b_1,\mathbf{x}),\cdots,G(\mathbf{a}_L,b_L,\mathbf{x})] \quad (23)$$

where G is the activation function and $\mathbf{a}_i$, b_i are parameters from input to hidden layer that are randomly generated. G needs to satisfy ELM universal approximation conditions [2]

$$\Omega_{\mathrm{ELM}}=\mathbf{H}\mathbf{H}^T. \quad (24)$$

Two types of nodes could be used: additive nodes and RBF nodes. In the following, the former two are additive nodes and the latter two are RBF nodes.

a. Sigmoid function

$$G(\mathbf{a},b,\mathbf{x})=\frac{1}{1+\exp\left(-(\mathbf{a}\cdot\mathbf{x}+b)\right)}. \quad (25)$$

b. Sinusoid function

$$G(\mathbf{a},b,\mathbf{x})=\sin(\mathbf{a}\cdot\mathbf{x}+b). \quad (26)$$

c. Multiquadric function

$$G(\mathbf{a},b,\mathbf{x})=\sqrt{(\|\mathbf{x}-\mathbf{a}\|^2+b^2)}. \quad (27)$$

d. Gaussian function

$$G(\mathbf{a},b,\mathbf{x})=\exp(-\frac{\|\mathbf{x}-\mathbf{a}\|^2}{b}). \quad (28)$$

2) Kernel—Ω_{ELM} could also be evaluated by kernels as in (18). Mercer's conditions must be satisfied. The kernel K could be, but not limited to the following.

a. Gaussian kernel

$$K(\mathbf{u},\mathbf{v})=\exp(-\frac{\|\mathbf{u}-\mathbf{v}\|^2}{2\sigma^2}). \quad (29)$$

b. Laplacian kernel

$$K(\mathbf{u},\mathbf{v})=\exp(-\frac{\|\mathbf{u}-\mathbf{v}\|}{\sigma}), \quad \sigma>0. \quad (30)$$

c. Polynomial kernel

$$K(\mathbf{u},\mathbf{v})=(1+\mathbf{u}\cdot\mathbf{v})^m, \quad m\in \boldsymbol{Z}^+. \quad (31)$$

Remark: If the output function G satisfies conditions mentioned in [2], ELM has universal approximation ability. Many types of hidden nodes would work well. They can be generated randomly as in the initial ELM. They can also be generated according to some explicit or implicit relationships. When using an implicit relationship, $\mathbf{h}(\mathbf{x})$ is unknown. Kernel trick can then be adopted: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{h}(\mathbf{x}_i)\cdot\mathbf{h}(\mathbf{x}_j)$, and the kernel K needs to satisfy Mercer's conditions.

Theorem 1: Dual problem of sparse ELM (17) is convex.

Proof: First-order partial derivative is

$$\frac{\partial}{\partial \alpha_s} L_d = t_s \sum_{j=1}^{N} \alpha_j t_j \Omega_{\text{ELM}}(\mathbf{x}_s, \mathbf{x}_j) - 1. \quad (32)$$

Second-order partial derivative is

$$\frac{\partial^2}{\partial \alpha_t \partial \alpha_s} L_d = t_t t_s \Omega_{\text{ELM}}(\mathbf{x}_t, \mathbf{x}_s). \quad (33)$$

Thus, Hessian matrix $\nabla^2 L_d = \mathbf{T}^T \boldsymbol{\Omega}_{\text{ELM}} \mathbf{T}$.

1. When $\boldsymbol{\Omega}_{\text{ELM}}$ is calculated from random hidden nodes directly as in (24)

$$\nabla^2 L_d = \mathbf{T}^T \mathbf{H} \mathbf{H}^T \mathbf{T} = (\mathbf{T}^T \mathbf{H}) \mathbf{I}_{L \times L} (\mathbf{T}^T \mathbf{H})^T \quad (34)$$

   where $\nabla^2 L_d$ is positive semi-definite.

2. When $\boldsymbol{\Omega}_{\text{ELM}}$ is evaluated from kernels, Mercer's conditions ensure that $\boldsymbol{\Omega}_{\text{ELM}}$ is positive semi-definite. Therefore, $\nabla^2 L_d = \mathbf{T}^T \boldsymbol{\Omega}_{\text{ELM}} \mathbf{T} \geq 0$ is positive semi-definite.

As Hessian matrix of $L_d (\nabla^2 L_d)$ is positive semi-definite, L_d is a convex function. Therefore, dual problem of sparse ELM is convex.

V. Training Algorithm of Sparse ELM

Similar to conventional SVM, sparse ELM is essentially a QP problem. The only difference between them is that sparse ELM does not have the sum constraint

$$\sum_{i=1}^{N} \alpha_i t_i = 0.$$

Better generalization performance is expected as the optimal solution is searched within a wider range. In addition, as fewer constraint needs to be satisfied, the training would be easier as well. However, early works only discussed sparse ELM theoretically [16]. The same implementation, usually the sequential minimal optimization (SMO) as proposed in [17], is used to obtain the solution of sparse ELM. The advantage of sparse ELM is not well explored. In this section, a new training algorithm is specifically developed for sparse ELM.

At first, let us take a review at Platt's SMO algorithm. The basic idea of SMO is to break the large QP problem into a series of smallest possible sub-QP problems, and to solve one sub-problem in each iteration. Time-consuming numerical optimization is avoided because these sub-problems could be solved analytically. Since sum constraint

$$\sum_{i=1}^{N}\alpha_i t_i=0$$

always needs to be satisfied, each smallest possible subproblem includes two Lagrange variables.

In sparse ELM, only one Lagrange variable needs to be updated in each iteration, since the sum constraint has vanished. The training algorithm of sparse ELM is based on iterative computation. However, in unified ELM, matrix inversion is utilized to generate the solution and the complexity is between quadratic and cubic with respect to the training size. Thus, training speed of sparse ELM is expected to become faster than that of unified ELM when the size grows. In addition, sparse ELM achieves faster testing speed and requires less storage space for problems of all scales. Consequently, sparse ELM is quite promising for growing-scale problems, such as neuroscience, image processing, data compression, and so on.

A. Optimality Conditions

Optimality conditions are used to determine if the optimal solution has been generated or not. If they are satisfied, the optimal solution is obtained, and vice versa.

Based on KKT conditions (20), we have three cases as follows.

1. $a_i = 0$

$$\begin{array}{c}\alpha_i=0 \Rightarrow t_i\, f(\mathbf{x}_i)-1 \geq 0\\ \mu_i=C \Rightarrow \xi_i=0.\end{array} \quad (35)$$

2. $0 < a_i < C$

$$\begin{array}{c}\alpha_i>0 \Rightarrow t_i\, f(\mathbf{x}_i)-1=0\\ \mu_i>0 \Rightarrow \xi_i=0.\end{array} \quad (36)$$

3. $a_i = C$

$$\begin{array}{c}\alpha_i=C \Rightarrow t_i\, f(\mathbf{x}_i)-1 \leq 0\\ \mu_i=0 \Rightarrow \xi_i>0.\end{array} \quad (37)$$

B. Improvement Strategy

Improvement strategy decides how to decrease the objective function when optimality conditions are not fully satisfied.

Suppose a_c is chosen to be updated at the current iteration based on the selection criteria to be introduced later. Then, the first- and second-order partial derivatives of objective function L_d with respect to a_c would be

$$\frac{\partial}{\partial \alpha_c} L_d = t_c \sum_{j=1}^{N} \alpha_j t_j \Omega_{\mathrm{ELM}}(\mathbf{x}_c, \mathbf{x}_j) - 1 = t_c f(\mathbf{x}_c) - 1$$
$$\frac{\partial^2}{\partial \alpha_c^2} L_d = \Omega_{\mathrm{ELM}}(\mathbf{x}_c, \mathbf{x}_c). \quad (38)$$

According to Theorem 1, dual problem of sparse ELM is a convex quadratic one. Thus, the global minimum exists and is achieved at α_c^* [25]

$$\alpha_c^* = \alpha_c + \frac{-\frac{\partial}{\partial \alpha_c} L_d}{\frac{\partial^2}{\partial \alpha_c^2} L_d} = \alpha_c + \frac{1 - t_c f(\mathbf{x}_c)}{\Omega_{\mathrm{ELM}}(\mathbf{x}_c, \mathbf{x}_c).} \quad (39)$$

As there are bounds [0, C] for all α_i's, the constrained minimum α_c^{new} is obtained by clipping the unconstrained minimum α_c^*

$$\alpha_c^{\mathrm{new}} = \alpha_c^{*,\mathrm{clip}} = \begin{cases} 0, & \alpha_c^* < 0 \\ \alpha_c^*, & \alpha_c^* \in [0, C] \\ C, & \alpha_c^* > C \end{cases}. \quad (40)$$

C. Selection Criteria

Since only one Lagrange variable is updated in every iteration, the choice of which one is vital. It is desirable to choose the Lagrange variable, which decreases the objective function L_d the most. However, it is time consuming to calculate the exact decrease of L_d that update of each variable could result in. Instead, we suggest a method to use the step size of α_i to approximate the decrease of L_d that α_i would cause.

Definition 1—d is the update direction. d_i indicates the way in which α_i should be updated.

1. $\alpha_i = 0$

   α_i can only increase. Therefore, $d_i = 1$.

2. $0 < \alpha_i < C$

   d_i should be along the direction in which L_d decreases. Therefore,

   $d_i = -\mathrm{sign}\left(\frac{\partial}{\partial \alpha_i} L_d\right)$.

3. $\alpha_i = C$

   α_i can only decrease. Therefore, $d_i = -1$.

Definition 2—J is the selection parameter

$$J_i = \left(\frac{\partial}{\partial \alpha_i} L_d\right) d_i, \quad i = 1, 2, \ldots, N. \quad (41)$$

The Lagrange variable corresponding to the minimal selection parameter is chosen to be updated

$$c = \arg \min_{i=1,\ldots,N} J_i. \quad (42)$$

Theorem 2—The minimum value of $J_i \left(\min_{i=1,\ldots,N} J_i \right)$ is negative in the training process, which guarantees that the update of the chosen Lagrange variable a_c will definitely decrease the objective function L_d.

Proof: In the training process, at least one data violates the optimality conditions. Otherwise, the optimal solution has been generated and the training algorithm would be terminated. Assuming the data corresponding to a_v, violates the optimality conditions given before. Three possible cases are given below.

1. $a_V = 0$

$$\begin{aligned} &\Rightarrow \frac{\partial}{\partial \alpha_v} L_d = t_v f(\mathbf{x}_v) - 1 < 0 \\ &J_v = \left(\frac{\partial}{\partial \alpha_v} L_d \right) \cdot 1 < 0. \end{aligned} \quad (43)$$

2. $0 < a_V < C$

$$\begin{aligned} &\Rightarrow \frac{\partial}{\partial \alpha_v} L_d = t_v f(\mathbf{x}_v) - 1 \neq 0 \\ &J_v = \frac{\partial}{\partial \alpha_v} L_d \cdot \left(-\text{sign} \left(\frac{\partial}{\partial \alpha_v} L_d \right) \right) < 0. \end{aligned} \quad (44)$$

3. $a_V = C$

$$\begin{aligned} &\Rightarrow \frac{\partial}{\partial \alpha_v} L_d = t_v f(\mathbf{x}_v) - 1 > 0 \\ &J_v = \left(\frac{\partial}{\partial \alpha_v} L_d \right) \cdot (-1) < 0. \end{aligned} \quad (45)$$

Therefore, $\min_{i=1,\ldots,N} J_i$ is always negative in the training process and L_d will decrease after every iteration.

D. Termination

The algorithm is based on iterative computation. Thus, the KKT conditions cannot be satisfied exactly. In fact, KKT conditions only need to be satisfied within a tolerance ε. According to [17], $\varepsilon = 10^{-3}$ could ensure great accuracy.

When $\min_{i=1,\ldots,N} (J_i) > -\varepsilon$, KKT conditions are fulfilled within a tolerance ε, and the training algorithm is terminated.

E. Convergence Analysis

Theorem 3—Training algorithm proposed in this paper will converge to the global optimal solution in a finite number of iterations.

Proof: As proved in Theorem 1, dual problem of sparse ELM is a convex QP problem. At every iteration, the chosen Lagrange variable α_c violated optimality conditions before the update, and as proved in Theorem 2, the update of α_c will make the objective function L_d monotonically decrease definitely.

In addition, Lagrange variables are all bounded within $[0, C]^N$. According to Osuna's theorem [26], the algorithm is convergent to the global optimal solution in a finite number of iterations.

F. Training Algorithm

The training algorithm is summarized in Algorithm 1. In the table, **g** denotes the gradient of L_d, where $g_i = \frac{\partial}{\partial \alpha_i} L_d$, **d** is update direction and **J** is selection parameter. For **G**, $G_{i,j} = t_i t_j \Omega_{\text{ELM}}(\mathbf{x}_i, \mathbf{x}_j)$.

Algorithm 1

Sparse ELM for classification

Problem formulation: Given a set of training data $\{\mathbf{x}_i, t_i | \mathbf{x}_i \in \mathbf{R}^d, t_i \in \{1, -1\}, i = 1, \ldots, N\}$, obtain the QP problem with an appropriate ELM kernel matrix $\mathbf{\Omega}_{\text{ELM}}$ and parameter C as in (17).

1: **Initialization:**

$\boldsymbol{\alpha} = \mathbf{0}$, $\mathbf{g} = \mathbf{G}\boldsymbol{\alpha} - \mathbf{1}$, $\mathbf{J} = \mathbf{g}$, $\mathbf{d} = \mathbf{1}$, $\boldsymbol{\alpha}, \mathbf{g}, \mathbf{J}, \mathbf{d} \in \mathbf{R}^N$.

2: **While** $\min_{i=1,\ldots,N} J_i < -\varepsilon$:

1 Update **J**, $J_i = g_i d_i$.

2 Obtain the minimum of J_i, $c = \arg \min_{i=1,\ldots,N} J_i$. And update the corresponding Lagrange variable α_c.

3 Update **g**, **d**.

Endwhile

VI. Performance Evaluation

In this section, the performance of sparse ELM is evaluated and compared with SVM and unified ELM on some benchmark data sets. All the datasets except for COD RNA are evaluated with MATLAB R2010b running in an Intel i5-2400 3.10 GHz CPU with 8.00 GB RAM. COD RNA dataset which needs more memory, marked with * in tables, is conducted in VIZ server with IBM system x3550 M3, dual quad-core Intel Xeon E5620 2.40 GHz CPU with 24.00 GB RAM. SVM and Kernel Methods MATLAB Toolbox [27] is used to implement SVM algorithms.

Sparse ELM and training algorithm are originally developed for binary classification. For multiclass problems, one-against-one (OAO) method is adopted to combine several binary

sparse ELM together. In order to conduct a fair comparison, multiclass classification of SVM also utilizes OAO method.

A. Data Sets Description

A wide types of data sets are used in experiments in order to obtain a thorough evaluation on the performance of sparse ELM. Binary and multiclass data sets are both included, which are of high or low dimensions, large or small sizes. These data sets are taken from UCI Repository, LIBSVM portal and etc [28]–[32]. Details are summarized in Tables I and II.

20 trials are conducted for each data set. In each trial, random permutation is performed within training data set and testing data set separately. Preprocessing is carried out for training data, making all attributes linearly scaled into [−1, 1]. And attributes of testing data would be scaled accordingly based on factors used in the scaling of training data. For binary classification, the label is either 1 or −1. For multiclass classification, the label is $1, 2, \cdots, N$, where N is the number of classes.

Data sets of colon cancer and Leukemia are originally taken from UCI repository. There are too many features. And they are not well selected. In order to obtain a better generalization performance for all these methods, feature selection is performed to these two data sets using the method proposed in [33]. 60 genes are selected from 2000 and 7129 ones respectively.

B. Influence of the Number of Hidden Nodes L

As stated before, L cannot be infinite in real implementation. Thus, training errors exist. It is expected that when L increases, training errors decrease. In addition, since over-fitting has been well solved, testing errors are also expected to decrease with the increase of L.

As shown in Figs. 2 and 3, training and testing accuracies get better when L increases in all values of C. And after L gets big enough, training and testing performance remain almost fixed. More results are plotted in Figs. 4 and 5 with $C = 1$ for simplicity. The relationship is consistent with our expectation.

In order to reduce human involvement, five-fold cross validation method is used to find a single L, which is large enough for all problems. In this paper, binary and multiclass problems are treated separately, and for all the cases reported, $L = 200$ for binary ones and $L = 1000$ for multiclass ones present great validation accuracy.

C. Parameter Specifications

Gaussian kernel $K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{\|\mathbf{u}-\mathbf{v}\|^2}{2\sigma^2}\right)$ and polynomial kernel $K(\mathbf{u}, \mathbf{v}) = (\mathbf{u}\cdot\mathbf{v}+1)^m$ are used. Figs. 6 and 7 respectively show the generalization performance of SVM and sparse ELM with Gaussian kernel. The plot for unified ELM is similar. Parameter combination of cost parameter C and kernel parameter s or m should be chosen *a priori*. Five-fold cross validation method is utilized for training data and the best parameter combination is thus chosen. Parameters of C and s are both tried with 14 different values, i.e., 0.01, 0.1, 0.2, 0.5,

1, 2, 5, 10, 20, 50, 100, 200, 500, and 1000, and m is tried with five values, i.e., 1, 2, 3, 4, and 5.

For sparse ELM and unified ELM with random hidden nodes, L is set to 200 for binary problems and 1000 for multiclass ones. Parameter C is tried with 14 values, i.e., 0.01, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 200, 500, and 1000.

As shown in Table III, the best parameters of C and σ or m are specified for each problem.

D. Performance Comparison

Best parameters of C and σ or m chosen by cross validation method are used for training and testing. The results include average training and testing accuracy, standard deviation of training and testing accuracy, and training and testing time. For each problem, the best testing accuracy and shortest training time are highlighted.

1) Binary Problems—Compared with SVM, as observed from Tables IV and V, sparse ELM of kernel form (Gaussian and polynomial) achieves better generalization performance for most data sets. And sparse ELM of random hidden nodes (Tables VI and VII) obtains comparable generalization performance with SVM, some cases better and other cases worse. As for the training speed, sparse ELM is much faster, up to 500 times when the data set is large, for both kernel and random hidden nodes form. Comparable testing speed is achieved since both SVM and sparse ELM provide compact networks.

Comparing to unified ELM (Tables IV–VII), similar generalization performance is achieved. When the data set is very small, training speed of sparse ELM is slower. However, this is not important since training speed is not the major concern when facing small problems. When the data set grows, training of sparse ELM becomes much faster than unified ELM, up to five times. In addition, sparse ELM requires less testing time for almost all data sets except very few cases, Colon (Gene Sel) and Leukemia (Gene Sel) with sigmoid hidden nodes. In these two cases, the size of training data is very small. Thus, even though sparse ELM provides a more compact network, the computations needed in the testing phase is only reduced slightly. And some unexpected perturbations in the computing might be account for the result.

2) Multiclass Problems—Compared with SVM, sparse ELM of kernel form (Gaussian and polynomial) obtains better generalization performance for most data sets. However, sparse ELM of random hidden nodes cannot achieve as great performance as SVM. The reason is that when dealing with multiclass problems, OAO method is utilized to combine several binary sparse ELM together. Due to the randomness in nature, the deviation of each binary sparse ELM of random hidden nodes is higher than that of corresponding binary SVM. And after combining these binary classifiers together, the effects of relatively high deviation are magnified, causing the decline of performance. In the aspect of training speed, sparse ELM is much faster than SVM, in both kernel and random hidden nodes form.

Compared with unified ELM (Tables IV–VII), similar generalization performance is achieved. Unified ELM solves multiclass problems directly, while sparse ELM needs to

combine several binary sparse ELM together by OAO method. Thus, it makes sparse ELM less advantageous than unified ELM in multiclass applications. For most data sets, unified ELM achieves faster training and testing speed. In addition, the deviation of training and testing accuracy of sparse ELM is much higher than that of unified ELM. Therefore, for multiclass problems, sparse ELM is sub-optimal to unified ELM.

3) Number of Support Vectors and Storage Space—Unified ELM deals with multiclass problems directly while both SVM and the sparse ELM adopt OAO method to combine several binary classifiers together. Thus, the number of total vectors are different. Therefore, when dealing with multiclass problems, the number of SVs of unified ELM is not compared with that of sparse ELM and SVM.

Observing from Table VIII, unified ELM provides a dense network as all vectors are SVs. The sparsity of SVM and the proposed sparse ELM may vary in different cases. However, generally speaking, the proposed ELM is sparse and provides a more compact network than unified ELM in all cases. And the storage space is proportional to the number of SVs. Therefore, less storage space is required by sparse ELM.

VII. Conclusion

ELM was initially proposed for SLFNs. Both regression and classification problems can be dealt with efficiently. The unified ELM simplify and unify different learning methods and different networks, including SLFNs, LS-SVM, and PSVM. However, both the initial ELM and unified ELM do not have sparsity, and require much storage space and testing time. In this paper, a sparse ELM is proposed for classification, reducing storage space and testing time significantly. Furthermore, the sparse ELM is also proved to unify several classification methods, including SLFNs, conventional SVM and RBF networks. Both kernels and random hidden nodes can be used in sparse ELM.

In addition, a fast iterative training algorithm is specifically developed for sparse ELM. In general, for binary classification, sparse ELM is advantageous over SVM and unified ELM: 1) it achieves better generalization performance and faster training speed than SVM and 2) it requires less testing time and storage space than unified ELM. Furthermore, for large-scale binary problems, it has even a faster training speed than unified ELM that has already outperformed many other methods.

References

1. Huang G-B, Zhu Q-Y, Siew C-K. Extreme learning machine: A new learning scheme of feedforward neural networks. Proc IJCNN. 2004; 2:985–990.
2. Huang GB, Chen L, Siew CK. Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw. Jul; 2006 17(4):879–892. [PubMed: 16856652]
3. Huang G-B, Zhu Q-Y, Siew C-K. Extreme learning machine: Theory and applications. Neurocomputing. Dec.2006 70:489–501.
4. Huang G-B, Chen L. Convex incremental extreme learning machine. Neurocomputing. Oct.2007 70:3056–3062.

5. Liu, Q.; He, Q.; Shi, Z. Advances in Knowledge Discovery and Data Mining (LNCS). Berlin, Germany: Springer; 2008. Extreme support vector machine classifier; p. 222-233.
6. Huang G-B, Chen L. Enhanced random search based incremental extreme learning machine. Neurocomputing. Oct.2008 71:3460–3468.
7. Frénay B, Verleysen M. Using SVMs with randomised feature spaces: An extreme learning approach. Proc 18th ESANN. 2010:315–320.
8. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagation errors. Nature. Oct.1986 323:533–536.
9. Huang GB, Zhu QY, Mao KZ, Siew CK, Saratchandran P, Sundararajan N. Can threshold networks be trained directly? IEEE Trans Circuits Syst II. Mar; 2006 53(3):187–191.
10. Li M-B, Huang G-B, Saratchandran P, Sundararajan N. Fully complex extreme learning machine. Neurocomputing. Oct.2005 68:306–314.
11. Liang NY, Huang GB, Saratchandran P, Sundararajan N. A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans Neural Netw. Nov; 2006 17(6):1411–1423. [PubMed: 17131657]
12. Cortes C, Vapnik V. Support vector networks. Mach Learn. 1995; 20(3):273–297.
13. Fung G, Mangasarian OL. Proximal support vector machine classifiers. Proc Int Conf Knowl Discover Data Min. 2001:77–86.
14. Suykens JAK, Vandewalle J. Least squares support vector machine classifiers. Neural Process Lett. 1999; 9(3):293–300.
15. Huang GB, Zhou H, Ding X, Zhang R. Extreme learning machine for regression and multiclass classification. IEEE Trans Syst, Man, Cybern B, Cybern. Apr; 2012 42(2):513–529. [PubMed: 21984515]
16. Huang G-B, Ding X, Zhou H. Optimization method based extreme learning machine for classification. Neurocomputing. Dec.2010 74:155–163.
17. Platt, J. Microsoft Research. Redmond, WA, USA: 1998. Sequential minimal optimization: A fast algorithm for training support vector machines. Tech. Rep. MSR-TR-98-14
18. Suykens J, Vandewalle J. Multiclass least squares support vector machines. Proc IJCNN. 2:900–903.
19. Tang Y, Zhang HH. Multiclass proximal support vector machines. J Comput Graph Statist. 2006; 15(2):339–355.
20. Drucker, H.; Burges, CJ.; Kaufman, L.; Smola, A.; Vapnik, V. Support vector regression machines. In: Mozer, M.; Jordan, J.; Petscbe, T., editors. Neural Information Processing Systems. Vol. 9. Cambridge, MA, USA: MIT Press; 1997. p. 155-161.
21. Smola A, Schölkopf B. A tutorial on support vector regression. Statist Comput. 2004; 14(3):199–222.
22. Fletcher, R. Practical Methods of Optimization: Volume 2 Constrained Optimization. New York, NY, USA: Wiley; 1981.
23. Widrow B, Greenblatt A, Kim Y, Park D. The no-prop algorithm: A new learning algorithm for multilayer neural networks. J Comput Graph Statist. 2013; 37:182–188.
24. Vapnik, V. The Nature of Statistical Learning Theory. Berlin, Germany: Springer; 1999.
25. Nocedal, J.; Wright, SJ. Numerical Optimization. Berlin, Germany: Springer; 1999.
26. Osuna E, Freund R, Girosi F. An improved training algorithm for support vector machines. Proc IEEE Workshop Neural Netw Signal Process. 1997:276–285.
27. Canu, S.; Grandvalet, Y.; Guigue, V.; Rakotomamonjy, A. Perception Systèmes et Information. INSA de Rouen; Rouen, France: 2005. SVM and kernel methods MATLAB toolbox.
28. Frank, A.; Asuncion, A. UCI Machine Learning Repository. 2010. [Online]. Available: archive.ics.uci.edu/ml/
29. Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D, et al. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc Nat Acad Sci. 1999; 96(12):6745–6750. [PubMed: 10359783]

30. Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, et al. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. Science. Oct.1999 286:531–537. [PubMed: 10521349]

31. Hsu, CW.; Chang, CC.; Lin, CJ. A Practical Guide to Support Vector Classification. 2003. [Online]. Available: http://www.csie.ntu.edu.tw/cjlin/papers.html

32. Uzilov A, Keegan J, Mathews D. Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change. BMC Bioinf. 2006; 7(1):173.

33. Peng H, Long F, Ding C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans Pattern Anal Mach Intell. Aug; 2005 27(8):1226–1238. [PubMed: 16119262]

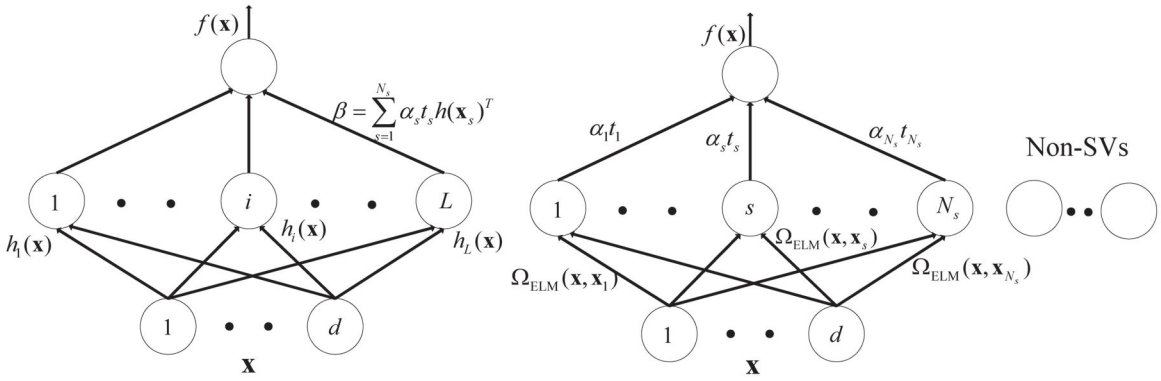

Fig. 1.
Primal and dual networks of sparse ELM.

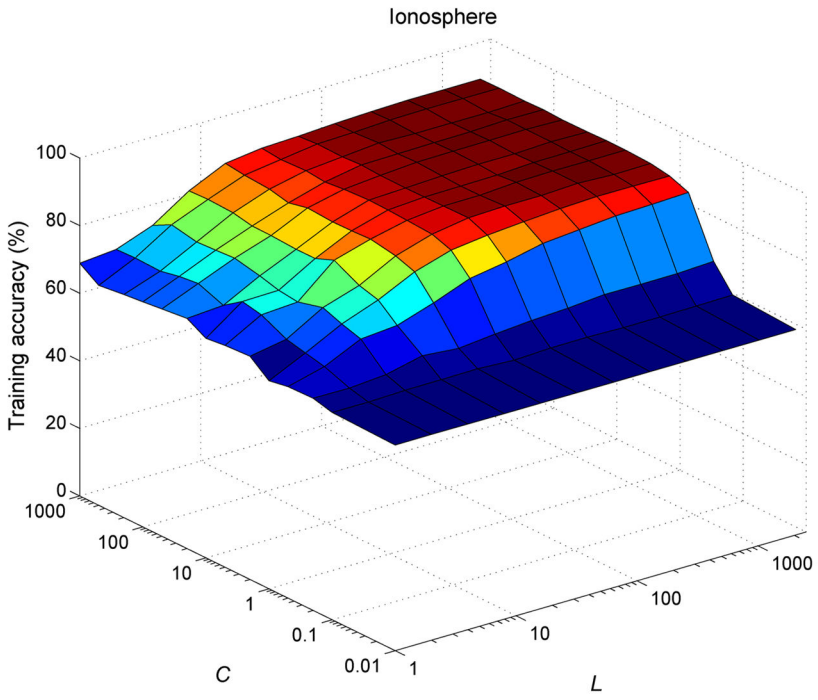

Fig. 2.
Training accuracy of sparse ELM with sinusoid nodes (ionosphere).

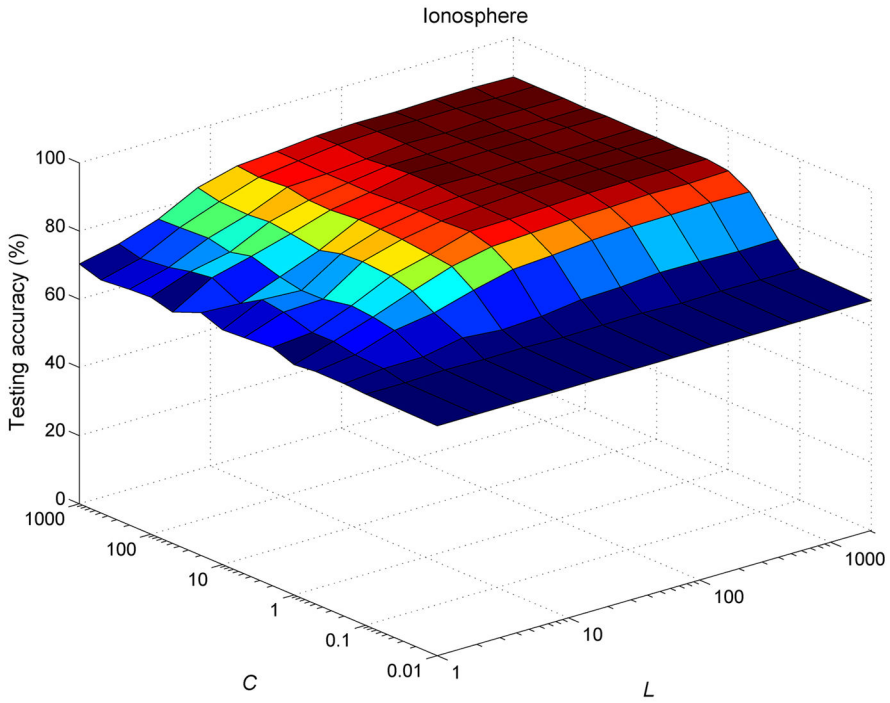

Fig. 3.
Testing accuracy of sparse ELM with sinusoid nodes (ionosphere).

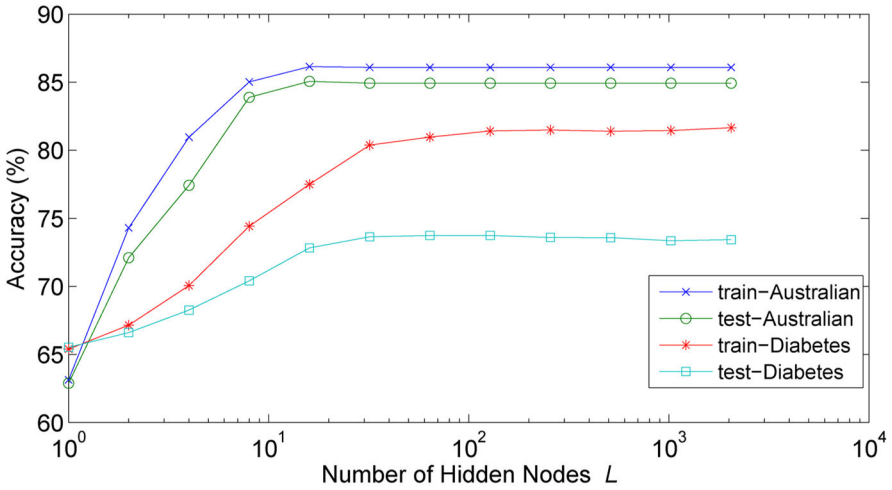

Fig. 4.
Performance of sparse ELM with sinusoid nodes for Australian and Diabetes (C=1).

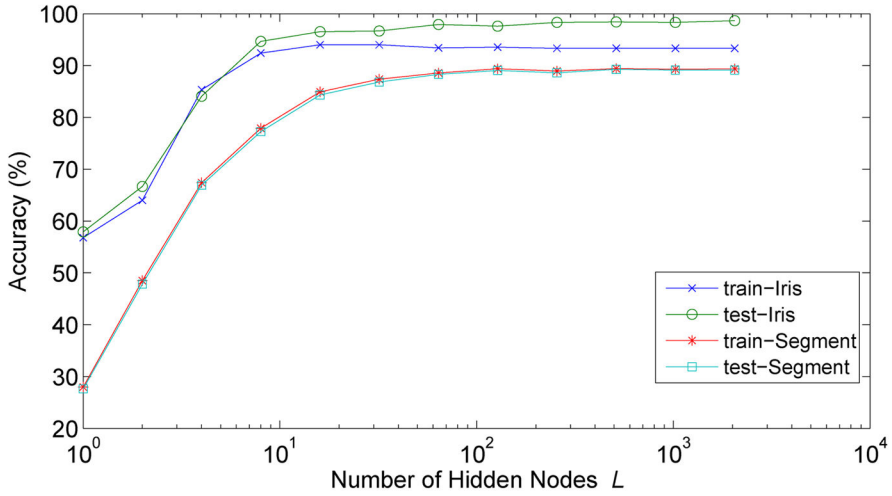

Fig. 5.
Performance of sparse ELM with sinusoid nodes for Iris and Segment (C=1).

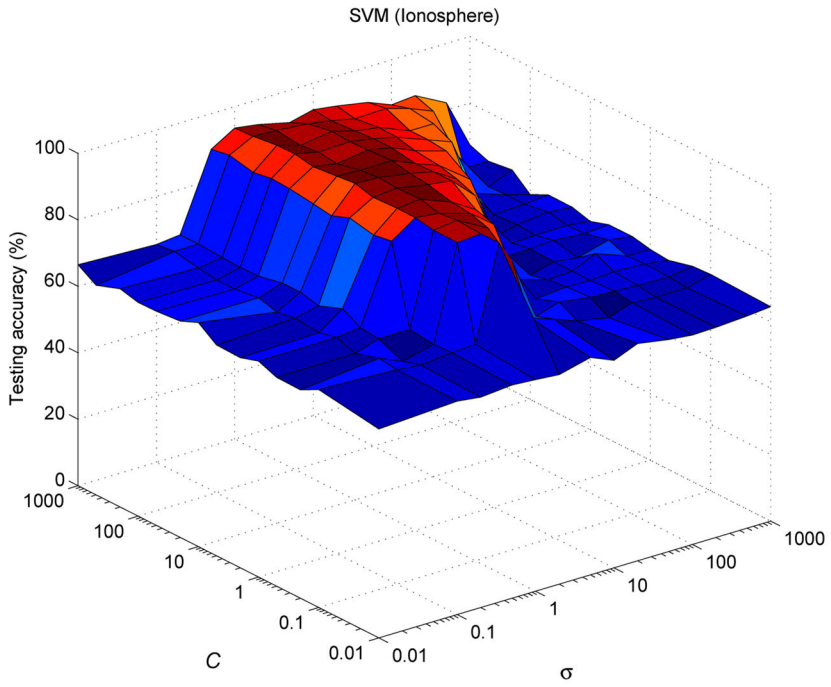

Fig. 6.
SVM (Gaussian kernel) for ionosphere data set.

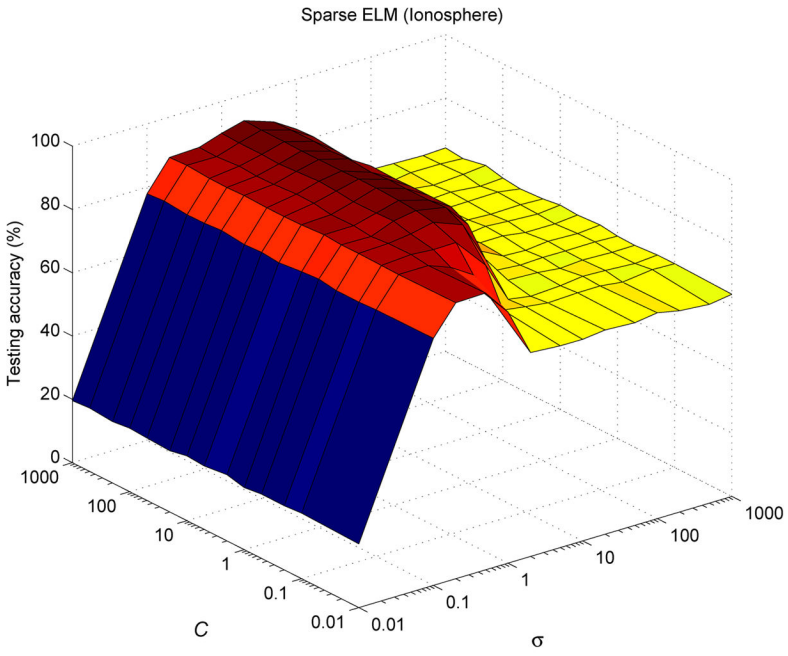

Fig. 7.
Sparse ELM (Gaussian kernel) for ionosphere data set.

TABLE I

Data Sets of Binary Classification

Class	Datasets	# Train	# Test	Features
Low Dims Small Size	Australian	345	345	14
	Breast Cancer	342	341	10
	Diabetes	384	384	8
	Heart	135	135	13
	Ionosphere	176	175	34
Low Dims Large Size	Mushroom	4062	4062	22
	SVMguide1	3089	4000	4
	Magic	9510	9510	11
	* COD RNA	29768	29767	8
High Dims Small Size	Colon Cancer	31	31	2000
	Colon (Gene Sel)	31	31	60
	Leukemia	38	34	7129
	Leukemia (Gene Sel)	38	34	60
High Dims Large Size	Spambase	2301	2300	57
	Adult	6414	26147	123

TABLE II

Data Sets of Multiclass Classification

Datasets	# Train	# Test	Features	Classes
Iris	75	75	4	3
Wine	89	89	13	3
Vowel	528	462	10	11
Segment	1155	1155	19	7
Satimage	4435	2000	36	6
DNA	2000	1186	180	3
SVMguide2	196	195	20	3
USPS	7291	2007	256	10

TABLE III

Parameter Specifications

Data sets	SVM				Unified ELM						Sparse ELM					
	Gaussian Kernel		Polynomial Kernel		Gaussian Kernel		Polynomial Kernel		Sigmoid Nodes	Sinusoid Nodes	Gaussian Kernel		Polynomial Kernel		Sigmoid Nodes	Sinusoid Nodes
	C	σ	C	m	C	σ	C	m	C	C	C	σ	C	m	C	C
Binary Classification																
Australian	1	20	0.1	2	5	2	1	2	10	0.2	200	2	1	3	20	50
Breast Cancer	2	1	1	3	2	1	2	2	100	200	200	1	1	3	100	5
Diabetes	10	5	0.1	2	10	5	1	2	5	200	0.2	0.5	0.2	3	1000	1000
Heart	1	2	10	1	20	10	5	1	2	100	5	5	0.5	1	500	50
Ionosphere	1	2	2	1	1	2	0.01	2	10	1000	1	2	0.01	2	200	5
Mushroom	1	1	1	3	1	1	1	2	20	20	1	1	1	4	5	2
SVMguide1	1	0.5	2	2	50	0.5	0.1	5	100	200	20	0.2	1	5	20	5
Magic	2	1	1	3	200	1	1	4	5	50	50	0.5	1	4	5	10
* COD RNA	2	1	2	3	5	1	1	3	5	5	1	0.5	2	3	50	20
Colon Cancer	1	1	1	1	5	50	0.01	1	5	10	1	20	1	1	10	100
Colon (Gene Sel)	2	2	0.2	1	1	0.1	1	4	5	5	5	2	1	4	0.2	100
Leukemia	50	500	1	1	500	1000	1	1	500	50	1	20	1	1	20	10
Leukemia (Gene Sel)	2	20	1	1	1	1	1	5	2	2	1	1	1	3	10	2
Spambase	5	0.5	1	3	10	1	0.01	3	100	1000	2	0.5	5	5	20	1
Adult	2	2	0.2	2	5	10	1	2	2	5	2	5	1	4	0.2	2
Multiclass Classification																
Iris	10	1	1	3	500	2	10	3	1000	500	1	0.5	2	2	1000	1000
Wine	5	1	1	3	1	2	0.5	1	2	1	5	0.5	10	2	1000	5
Vowel	10	1	10	3	20	0.5	20	4	1000	1000	2	0.2	10	4	200	100
Segment	1000	0.2	1	4	1	0.1	0.1	5	1000	1000	1	0.1	10	4	1000	50
Satimage	500	1	2	3	1	0.2	0.1	4	500	1000	1	0.2	1	3	1000	1000
DNA	500	20	1	3	1	1	1	3	2	200	1	20	10	3	10	10
SVMguide2	5	0.5	1	3	1	0.2	0.01	3	20	1000	50	0.2	1	1	1000	5

Data sets	SVM				Unified ELM					Sparse ELM					
	Gaussian Kernel		Polynomial Kernel		Gaussian Kernel		Polynomial Kernel		Sigmoid Nodes	Sinusoid Nodes	Gaussian Kernel		Polynomial Kernel		Sigmoid Nodes	Sinusoid Nodes
	C	σ	C	m	C	σ	C	m	C	C	C	σ	C	m	C	C
USPS	10	10	1	4	1	1	0.01	3	1000	1000	1	1	1	4	1000	1000

TABLE IV

Performance of Sparse ELM, Unified ELM, and SVM with Gaussian Kernel

Data sets	SVM (Gaussian Kernel)				Unified ELM (Gaussian Kernel)				Sparse ELM (Gaussian Kernel)			
	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)
Binary Classification												
Australian	90.96±0	83.09±0	0.2411	0.0039	93.62±0	84.35±0	0.0098	0.0043	90.61±0.40	**84.62±0.60**	**0.0088**	0.0012
Breast Cancer	98.25±0	97.36±0	0.0550	0.0010	99.12±0	**98.24±0**	0.0092	0.0039	99.05±0.22	98.21±0.13	**0.0075**	0.0009
Diabetes	78.65±0	73.96±0	0.1319	0.0026	83.33±0	74.48±0	**0.0119**	0.0062	84.92±0.18	**74.67±0.16**	0.0164	0.0030
Heart	92.59±0	82.96±0	0.0385	0.0007	84.44±0	**84.44±0**	**0.0033**	0.0011	85.48±0.49	**84.44±0.74**	0.0042	0.0006
Ionosphere	93.75±0	**93.71±0**	0.0667	0.0009	96.02±0	91.43±0	**0.0036**	0.0015	95.45±0	90.31±0.38	0.0063	0.0007
Mushroom	100±0	100±0	41.4878	0.3148	100±0	100±0	2.3835	0.6463	100±0	100±0	**0.8188**	0.0584
SVMguide1	97.09±0	96.90±0	5.1869	0.1154	97.38±0	96.85±0	1.1208	0.4803	97.42±0.07	**97.00±0.06**	**0.4809**	0.0703
Magic	84.29±0	85.73±0	311.7731	2.2336	88.46±0	**86.88±0**	24.0994	4.5080	87.47±0.05	86.20±0.07	**5.1139**	1.4432
* COD RNA	95.31±0	**95.25±0**	3858.0860	11.8995	95.33±0	95.22±0	354.8308	51.7553	94.26±0.10	94.44±0.00	**62.7069**	19.0089
Colon Cancer	100±0	70.97±0	0.0494	0.0382	96.77±0	87.10±0	0.0412	0.0395	95.16±1.61	**90.16±2.16**	**0.0357**	0.0316
Colon (Gene Sel)	100±0	**93.55±0**	0.0128	0.0004	100±0	90.32±0	0.0027	0.0008	100±0	**93.55±0**	**0.0020**	0.0006
Leukemia	100±0	**82.35±0**	0.4327	0.4389	100±0	**82.35±0**	0.4134	0.3949	100±0	79.41±0	**0.4093**	0.3856
Leukemia (Gene Sel)	100±0	100±0	0.0114	0.0013	100±0	100±0	0.0017	0.0008	100±0	100±0	**0.0016**	0.0004
Spambase	96.61±0	92.83±0	9.7045	0.1706	95.13±0	**93.70±0**	0.5707	0.2841	95.10±0.12	93.02±0.10	**0.3197**	0.0956
Adult	90.47±0	84.33±0	172.3218	7.2712	85.02±0	**84.66±0**	6.6666	9.7930	85.03±0.11	84.48±0.04	**2.5282**	3.4892
Multiclass Classification												
Iris	100±0	93.33±0	0.0253	0.0009	100±0	**97.33±0**	0.0029	0.0008	98.40±0.53	97.27±1.60	**0.0028**	0.0007
Wine	100±0	97.75±0	0.0304	0.0011	100±0	**98.89±0**	**0.0027**	0.0008	100±0	97.92±0.82	0.0060	0.0013
Vowel	99.81±0	62.55±0	0.6316	0.0310	100±0	57.79±0	**0.0230**	0.0098	100±0	**63.55±1.25**	0.1355	0.0475
Segment	100±0	91.43±0	5.1300	0.3360	100±0	**96.10±0**	**0.2311**	0.0641	100±0	95.77±0.34	0.2357	0.2303
Satimage	100±0	90.55±0	11.5949	0.4161	100±0	**90.95±0**	2.6646	0.3495	99.85±0.02	90.08±0.26	**2.4090**	1.5518
DNA	100±0	**94.10±0**	2.0669	0.1056	100±0	85.24±0	**0.4383**	0.1628	100±0	86.94±0.38	0.5307	0.3038
SVMguide2	100±0	56.41±0	0.1832	0.0031	100±0	**63.08±0**	**0.0028**	0.0022	100±0	**63.08±0**	0.0153	0.0033
USPS	99.88±0	**95.07±0**	20.0226	1.9588	99.99±0	94.97±0	**10.4227**	0.8524	99.99±0	94.82±0.08	10.4365	9.2329

TABLE V

Performance of Sparse ELM, Unified ELM, and SVM with Polynomial Kernel

Data sets	SVM (Polynomial Kernel)				Unified ELM (Polynomial Kernel)				Sparse ELM (Polynomial Kernel)			
	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)
Binary Classification												
Australian	90.72±0	**84.93±0**	0.0749	0.0003	92.46±0	**84.93±0**	**0.0056**	0.0027	90.46±0.40	84.23±0.78	0.0108	0.0022
Breast Cancer	100	95.31±0	0.0359	0.0015	98.86±0	97.65±0	**0.0050**	0.0016	99.23±0.28	**98.53±0.21**	0.0093	0.0011
Diabetes	83.07±0	74.74±0	0.1180	0.0005	83.59±0	**75.26±0**	**0.0071**	0.0073	81.95±0.73	73.89±0.43	0.0173	0.0066
Heart	87.41±0	82.22±0	0.0348	0.0003	82.96±0	83.70±0	**0.0023**	0.0008	83.85±1.04	**84.00±1.16**	0.0033	0.0003
Ionosphere	94.89±0	89.71±0	0.0410	0.0001	97.73±0	**91.43±0**	**0.0020**	0.0008	92.59±0.99	90.40±0.50	0.0044	0.0005
Mushroom	100±0	100±0	4.5436	0.0760	100±0	100±0	1.3134	0.2049	100±0	100±0	**1.0398**	0.0468
SVMguide1	96.60±0	96.25±0	4.3610	0.0175	97.02±0	**96.63±0**	1.2677	0.7589	96.44±0.09	96.19±0.08	**0.6873**	0.1456
Magic	87.19±0	86.11±0	361.8243	2.6178	87.78±0	**86.42±0**	18.2095	5.3064	86.14±0.12	85.61±0.12	**6.4395**	2.0246
* COD RNA	95.22±0	95.00±0	4342.7460	14.4478	95.00±0	**95.01±0**	208.9760	26.0668	94.92±0.03	94.98±0.05	**36.6408**	5.0626
Colon Cancer	100±0	77.42±0	0.0121	0.0003	100±0	80.65±0	0.0039	0.0047	98.48±2.39	**89.84±2.34**	**0.0018**	0.0016
Colon (Gene Sel)	100±0	90.32±0	0.0126	0.0007	100±0	90.32±0	**0.0007**	0.0006	100±0	**93.55±0**	0.0015	0.0006
Leukemia	100±0	85.29±0	0.0121	0.0019	100±0	**88.24±0**	0.0062	0.0052	100±0	83.53±3.40	**0.0029**	0.0021
Leukemia (Gene Sel)	100±0	97.06±0	0.0088	0.0001	100±0	**100±0**	0.0029	0.0017	100±0	**100±0**	**0.0020**	0.0008
Spambase	97.83±0	91.87±0	8.0709	0.1114	94.18±0	**92.39±0**	0.5882	0.3232	88.53±0.17	88.53±0.18	**0.4283**	0.1936
Adult	90.38±0	82.15±0	244.3654	1.6786	90.04±0	82.14±0	5.4775	4.0977	89.14±0.12	**84.31±0.09**	**2.9209**	3.6742
Multiclass Classification												
Iris	100±0	96.00±0	0.0264	0.0009	100±0	97.33±0	0.0076	0.0010	99.00±0.93	**97.47±1.66**	**0.0016**	0.0003
Wine	100±0	97.75±0	0.0332	0.0013	100±0	**98.88±0**	**0.0018**	0.0009	99.44±0.75	97.46±1.89	0.0042	0.0004
Vowel	100±0	59.74±0	0.7054	0.0714	100±0	62.64±0	**0.0526**	0.0202	97.97±0.51	**64.86±3.89**	0.1561	0.1501
Segment	99.83±0	96.45±0	0.4502	0.0792	99.13±0	**96.88±0**	**0.1603**	0.0871	95.90±0.36	88.00±0.28	0.2332	0.0965
Satimage	98.35±0	89.55±0	11.2106	0.4514	95.96±0	89.05±0	4.5560	0.6345	93.96±0.14	**90.96±2.76**	**3.0376**	0.4097
DNA	100±0	94.86±0	25.6512	0.3876	100±0	94.86±0	0.5727	0.2203	99.76±0.02	**95.10±1.52**	**0.5695**	0.2369
SVMguide2	100±0	56.41±0	0.0744	0.0039	94.39±0	56.41±0	**0.0046**	0.0027	94.52±0.69	**58.85±4.00**	0.0091	0.0005
USPS	100±0	95.52±0	27.9954	2.3716	99.99±0	94.92±0	12.2367	0.9945	99.27±0.03	**96.66±5.99**	**9.9507**	1.6330

TABLE VI

Performance of Sparse ELM and Unified ELM with Sigmoid Hidden Nodes

Data sets	Unified ELM (Sigmoid Hidden Nodes)				Sparse ELM (Sigmoid Hidden Nodes)			
	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)
Binary Classification								
Australian	89.00±0.02	84.62±0.03	**0.0100**	0.0067	87.10±0.94	**85.32±0.72**	0.0181	0.0023
Breast Cancer	97.42±0.01	97.89±0.02	0.0103	0.0073	98.01±0.21	**97.99±0.39**	**0.0085**	0.0029
Diabetes	82.60±0.00	**74.32±0.00**	**0.0118**	0.0076	81.41±0.93	74.11±0.59	0.0148	0.0052
Heart	85.70±0.01	83.63±0.01	**0.0039**	0.0023	85.74±0.90	**83.81±1.18**	0.0079	0.0016
Ionosphere	94.46±0.01	90.63±0.01	**0.0047**	0.0028	92.05±0.80	**90.66±1.26**	0.0073	0.0022
Mushroom	99.91±0	**99.84±0**	2.0256	0.3225	98.61±0.44	98.17±0.51	**0.5291**	0.0912
SVMguide1	94.57±0.01	**94.35±0.01**	0.9222	0.2601	94.33±0.31	94.30±0.48	**0.3313**	0.0955
Magic	82.89±0.02	**82.84±0.02**	11.0930	1.4562	81.48±0.27	81.55±0.30	**3.0090**	0.8074
* COD RNA	94.63±0	**94.63±0**	203.9884	9.4621	94.29±0.04	94.36±0.05	**32.0594**	2.2913
Colon Cancer	100±0	83.39±0.06	**0.0085**	0.0037	94.03±3.27	**89.03±4.26**	0.0097	0.0036
Colon (Gene Sel)	100±0	93.06±0.02	0.0024	0.0012	98.98±0.02	**93.55±0**	**0.0015**	0.0015
Leukemia	100±0	76.91±0.05	0.0379	0.0143	98.03±2.18	**78.09±2.37**	**0.0294**	0.0087
Leukemia (Gene Sel)	100±0	98.82±0.01	0.0026	0.0013	100±0	**99.12±1.35**	**0.0025**	0.0015
Spambase	91.29±0.00	**91.18±0.00**	0.5428	0.1239	89.03±1.58	84.78±1.44	**0.2374**	0.0921
Adult	84.46±0.00	**84.29±0**	7.8926	3.1285	83.28±0.58	83.41±0.57	**1.3478**	1.3899
Multiclass Classification								
Iris	98.67±0	97.20±0.00	**0.0045**	0.0046	97.00±1.33	**97.40±0.66**	0.0085	0.0115
Wine	100±0	**99.16±0.01**	**0.0061**	0.0061	100±0	**99.16±0.70**	0.0103	0.0142
Vowel	94.63±0.08	57.85±0.07	**0.0405**	0.0443	96.13±1.67	**59.84±2.46**	0.2709	1.3603
Segment	97.71±0.00	**95.88±0.00**	**0.1809**	0.1505	91.68±0.45	91.57±0.60	0.3961	1.5215
Satimage	92.88±0.00	**89.89±0.00**	3.8516	0.7572	87.86±0.17	85.65±0.22	**2.8562**	4.3452
DNA	98.06±0.00	**93.68±0.01**	0.5864	0.2724	94.45±1.88	88.19±2.55	**0.5002**	0.5810
SVMguide2	92.59±0.01	**54.74±0.15**	**0.0129**	0.0148	84.44±1.09	53.56±5.10	0.0233	0.0359
USPS	99.09±0	93.51±0.00	11.1521	1.2797	98.13±0.08	**93.64±0.15**	**9.0268**	15.0449

IEEE Trans Cybern. Author manuscript; available in PMC 2016 May 27.

TABLE VII

Performance of Sparse ELM and Unified ELM with Sinusoid Hidden Nodes

Data sets	Unified ELM (Sinusoid Hidden Nodes)				Sparse ELM (Sinusoid Hidden Nodes)			
	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)
Binary Classification								
Australian	86.84±0.00	**85.84±0.00**	**0.0094**	0.0064	86.42±0.62	85.30±0.62	0.0095	0.0029
Breast Cancer	98.03±0.00	**98.56±0.00**	0.0089	0.0057	98.02±0.23	97.82±0.39	**0.0077**	0.0025
Diabetes	83.48±0.00	74.49±0.00	**0.0105**	0.0068	81.72±0.57	**74.77±0.66**	0.0127	0.0038
Heart	88.07±0.01	83.15±0.01	**0.0038**	0.0019	85.22±1.11	**84.56±1.13**	0.0048	0.0010
Ionosphere	94.91±0.01	88.09±0.01	**0.0036**	0.0025	89.03±0.68	**88.63±1.02**	0.0065	0.0015
Mushroom	99.92±0	**99.88±0**	1.9781	0.3237	97.79±0.27	97.32±0.31	**0.5043**	0.0860
SVMguide1	95.22±0.00	**94.86±0.00**	0.6956	0.2490	94.50±0.17	94.61±0.24	**0.3210**	0.0867
Magic	84.02±0.00	**83.77±0.00**	11.3833	1.4718	82.20±0.13	82.82±0.13	**2.9645**	0.7939
* COD RNA	94.28±0	**94.38±0**	222.0702	9.3984	93.95±0.06	94.01±0.08	**33.0523**	2.7118
Colon Cancer	100±0	82.10±0.06	**0.0084**	0.0031	90.16±2.39	**88.06±4.22**	0.0094	0.0030
Colon (Gene Sel)	99.89±0	91.29±0.03	0.0015	0.0014	98.89±2.90	**93.71±1.24**	**0.0013**	0.0011
Leukemia	100±0	81.03±0.07	0.0345	0.0148	98.03±1.64	**81.47±4.02**	**0.0290**	0.0084
Leukemia (Gene Sel)	100±0	**99.12±0.01**	0.0029	0.0017	100±0	98.82±1.44	**0.0016**	0.0012
Spambase	90.32±0.00	**90.87±0.00**	0.5050	0.1201	87.39±3.39	85.79±2.10	**0.2281**	0.0086
Adult	84.80±0	84.55±0	4.6063	2.9173	84.71±0.16	**84.66±0.07**	**1.3163**	1.2399
Multiclass Classification								
Iris	98.60±0.00	96.20±0.01	**0.0043**	0.0037	97.27±0.29	**97.33±0.42**	0.0057	0.0082
Wine	100±0	**99.10±0.01**	**0.0056**	0.0046	100±0	**99.10±0.84**	0.0084	0.0122
Vowel	97.23±0.00	59.77±0.01	**0.0389**	0.0428	97.40±1.49	**60.74±1.98**	0.2407	1.0784
Segment	96.29±0.00	**95.28±0.00**	**0.1373**	0.1423	91.34±0.46	91.16±0.57	0.4104	1.5421
Satimage	86.09±0.00	83.61±0.00	**2.1227**	0.6790	87.84±0.14	**85.70±0.23**	2.8198	4.4148
DNA	98.11±0.00	**94.57±0.00**	**0.4008**	0.2478	96.64±0.18	94.13±0.42	0.4793	0.5367
SVMguide2	95.58±0.01	**59.77±0.07**	**0.0116**	0.0133	84.21±1.12	59.38±5.56	0.0221	0.0344
USPS	97.97±0.00	**93.56±0.00**	**6.7664**	1.1829	96.76±0.06	92.98±0.11	9.1235	14.9499

TABLE VIII

Number of Support Vectors

Data sets	# Total Vectors	SVM		Sparse ELM				Unified ELM			
		Gaussian Kernel	Polynomial Kernel	Gaussian Kernel	Polynomial Kernel	Sigmoid Nodes	Sinusoid Nodes	Gaussian Kernel	Polynomial Kernel	Sigmoid Nodes	Sinusoid Nodes
Binary Classification											
Australian	345	306	109	240.35	112.85	122.7	119.7	345	345	345	345
Breast Cancer	342	80	41	68	51.3	53.2	52.25	342	342	342	342
Diabetes	384	213	202	278	204.1	219.7	217.55	384	384	384	384
Heart	135	72	46	79.45	58.75	68.8	68.05	135	135	135	135
Ionosphere	176	93	48	98.95	85.65	94.15	101.45	176	176	176	176
Mushroom	4062	956	135	323.15	174.9	582.8	727.75	4062	4062	4062	4062
SVMguide1	3089	429	354	464.35	575.4	910.15	886.75	3089	3089	3089	3089
Magic	9510	3469	3190	3262.1	3599.75	4429.25	4428.6	9510	9510	9510	9510
* COD RNA	29767	5002	3912	11359	5972.3	7578.8	7853.6	29767	29767	29767	29767
Colon Cancer	31	31	24	29.3	25.2	27.6	26.85	31	31	31	31
Colon (Gene Sel)	31	30	17	25.85	24.6	29.35	19.55	31	31	31	31
Leukemia	38	33	32	38	32.9	27.8	27.05	38	38	38	38
Leukemia (Gene Sel)	38	12	7	38	22.45	12.3	11.1	38	38	38	38
Spambase	2301	772	392	810.95	1311.8	1586.2	1590.4	2301	2301	2301	2301
Adult	6414	2729	2261	2531.15	2450.85	2918.45	2666.1	6414	6414	6414	6414
Multiclass Classification											
Iris	150	29	23	54.45	38.4	46.8	47.4				
Wine	178	72	46	149.05	49.9	54.55	53.25				
Vowel	5280	1281	1066	4146.25	1692.3	2487.4	2483.8				
Segment	6930	5010	328	6116.75	866.45	1323.5	1370.05				
Satimage	22175	2376	1528	19197.1	2073.3	2705	2690.2				
DNA	4000	612	2237	3830.95	1967.05	1136.75	1040.85				
SVMguide2	392	386	163	392	175.1	188.95	187.5				
USPS	65619	3179	5404	58532.4	5581.3	6205.7	6573.1				