Sparse Extreme Learning Machine for Classification

Zuo Bai, Guang-Bin Huang, Danwei Wang, Han Wang and M. Brandon Westover

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, LS-SVM, PSVM and etc. 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 (QP) problem involved in sparse ELM is divided into a series of smallest possible sub-problems, each of which is 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 large-scale binary classification problems, sparse ELM realizes even faster training speed than unified ELM.

Index Terms—Extreme learning machine (ELM), sparse ELM, unified ELM, classification, quadratic programming (QP), support vector machine (SVM), sequential minimal optimization (SMO)

I. INTRODUCTION

EXTREME learning machine (ELM) was initially proposed for single-hidden-layer feedforward neural networks (SLFNs) [1]–[3]. And then 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:

$$f(x) = h(x)\beta$$ \hspace{1cm} (1)

where $x \in \mathbb{R}^d$, $h(x) \in \mathbb{R}^{1 \times L}$, $\beta \in \mathbb{R}^L$. $f(x)$ is the output; $x$ is the input; $h(x)$ is the hidden layer; and $\beta$ is the weight vector between the hidden nodes and output node. Hidden nodes are randomly generated and $\beta$ 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 $(\phi : x \rightarrow \phi(x))$. 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 $K(x, x_s)$, and the output weight is $[\alpha_1 t_1, \cdots, \alpha_N t_N]^T$. $x_s$ is the $s$-th support vector and $\alpha_s, t_s$ 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, 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 etc. However, sparsity is lost as equality constraints are used, like LS-SVM/PSVM.

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 sub-problem includes two Lagrange variables ($\alpha_i$’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 $\alpha_i$ 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 briefly introduced, 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 $(x_i, t_i), i = 1, \cdots, N, x_i \in \mathbb{R}^d$ and $t_i \in \{-1, 1\}$. Normally, it is non-separable in the input space. Thus, a nonlinear feature mapping is needed.

$$\phi : x_i \rightarrow \phi(x_i)$$

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:

$$\text{Minimize: } L_p = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi_i$$

Subject to:

$$t_i (w \cdot \phi(x_i) + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0, \quad i = 1, \cdots, N$$

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

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

$$\text{Minimize: } L_d = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j t_i t_j K(x_i, x_j) - \sum_{i=1}^{N} \alpha_i$$

Subject to:

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

$$0 \leq \alpha_i \leq C, \quad i = 1, \cdots, N$$

where kernel $K(u, v) = \phi(u) \cdot \phi(v)$ is often used since dealing with $\phi$ 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:

$$\text{Minimize: } L_p = \frac{1}{2} \|w\|^2 + b^2 + C \sum_{i=1}^{N} \xi_i^2$$

Subject to:

$$t_i (w \cdot \phi(x_i) + b) = 1 - \xi_i, \quad i = 1, \cdots, N$$

The final solution is:

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

in which, $G_{i,j} = t_i t_j K(x_i, x_j)$. For both cases, the decision function is:

$$f(x) = \text{sign} \left( \sum_{s=1}^{N_c} \alpha_s t_s K(x, x_s) + b \right)$$

where $x_s$ is support vector (SV) and $N_c$ is the number of SVs.

For conventional SVM which has inequality constraints, many Lagrange variables ($\alpha_i$’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. for SLFNs [1], [3], 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 $(x_i, t_i), i = 1, \cdots, N$. 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.

\[
H = \begin{bmatrix} h(x_1) \\ h(x_2) \\ \vdots \\ h(x_N) \end{bmatrix}, T = [t_1 \ t_2 \ \cdots \ t_N]^T
\]

\[
H\beta = T
\]

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 \( \beta \) can be obtained in different ways \([2],[3],[23]\). For example, a simple way is to obtain the following smallest norm least-squares solution \([3]\):

\[
\beta = H^+ T
\]

where \( H^+ \) is the Moore-Penrose generalized inverse of \( H \).

### B. Unified ELM

Liu et al. \([5]\) and Frenay et al. \([7]\) 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:

\[
\begin{align*}
\text{Minimize: } & \quad L_p = \frac{1}{2} \| \beta \|^2 + C \sum_{i=1}^{N} \xi_i^2 \\
\text{Subject to: } & \quad h(x_i)\beta = t_i - \xi_i, \quad i = 1, \cdots, N
\end{align*}
\]

The output function of the unified ELM is:

\[
\begin{align*}
f(x) &= h(x)\beta = h(x)H^T \left( \frac{1}{C} + HH^T \right)^{-1} T \\
&= h(x) \left( \frac{1}{C} + H^TH \right)^{-1} H^T T
\end{align*}
\]

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.

\[
\Omega_{ELM} = HH^T, \quad \Omega_{ELM}(x_i,x_j) = h(x_i)h(x_j)^T = K(x_i,x_j)
\]

where \( \Omega_{ELM} \) is called ELM kernel matrix.

In unified ELM, most errors \( \xi_i \)'s are non-zero (positive or negative) to make the equality constraint \( h(x_i)\beta = t_i - \xi_i \) satisfied for all training data. As Lagrange variables \( \alpha_i \)'s are proportional to corresponding \( \xi_i \)'s (\( \alpha_i = C\xi_i \)), almost all \( \alpha_i \)'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 \( h(x) \) 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 \( \mathbb{R}^d \), there exists a continuous function \( f(x) \) being able to separate them. ELM has universal approximation capability \([2]\). In other words, given any target function \( f(x) \), there exist \( \beta_i \)'s:

\[
\lim_{L \to +\infty} \left\| \sum_{i=1}^{L} \beta_i h_i(x) - f(x) \right\| = 0
\]

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 \( \xi_i \)'s should be allowed. Overfitting could be well solved by minimizing both empirical errors \( \sum_{i=1}^{N} \xi_i \) and structural risks \( \| \beta \|^2 \) 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:

\[
\begin{align*}
\text{Minimize: } & \quad L_p = \frac{1}{2} \| \beta \|^2 + C \sum_{i=1}^{N} \xi_i \\
\text{Subject to: } & \quad t_i h(x_i)\beta \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \cdots, N
\end{align*}
\]

The Lagrange function is:

\[
P(\beta, \xi, \alpha, \mu) = \frac{1}{2} \| \beta \|^2 + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \mu_i \xi_i \\
- \sum_{i=1}^{N} \alpha_i \left( t_i h(x_i)\beta - (1 - \xi_i) \right)
\]

where \( \alpha_i \)'s are Lagrange multipliers.
At the optimal solution, we have:

\[
\frac{\partial P}{\partial \beta} = 0 \Rightarrow \beta = \sum_{i=1}^{N} \alpha_i t_i h(\mathbf{x}_i)^T = \sum_{s=1}^{N_s} \alpha_s t_s h(\mathbf{x}_s)^T
\]

\[
\frac{\partial P}{\partial \xi_t} = 0 \Rightarrow C = \alpha_i + \mu_i
\]

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

\[
\text{Minimize: } L_d = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j t_i t_j \Omega_{\text{ELM}}(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{N} \alpha_i
\]

subject to: \( 0 \leq \alpha_i \leq C, \quad i = 1, ..., N \)

where \( \Omega_{\text{ELM}} \) is the ELM kernel matrix:

\[
\Omega_{\text{ELM}}(\mathbf{x}_i, \mathbf{x}_j) = h(\mathbf{x}_i)h(\mathbf{x}_j)^T = K(\mathbf{x}_i, \mathbf{x}_j)
\]

Therefore, the output of sparse ELM is:

\[
f(\mathbf{x}) = h(\mathbf{x})\beta = h(\mathbf{x}) \left( \sum_{i=1}^{N} \alpha_i t_i h(\mathbf{x}_i)^T \right) = h(\mathbf{x}) \left( \sum_{s=1}^{N_s} \alpha_s t_s \Omega_{\text{ELM}}(\mathbf{x}, \mathbf{x}_s) \right)
\]

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

**B. Sparsity analysis**

**KKT conditions are:**

\[
\alpha_i (t_i h(\mathbf{x}_i)\beta - (1 - \xi_i)) = 0
\]

\[
\mu_i \xi_i = 0
\]

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

1) \( 0 < \alpha_i < C \)

\[
\mu_i > 0 \Rightarrow \xi_i = 0
\]

\[
\alpha_i > 0 \Rightarrow t_i h(\mathbf{x}_i)\beta - 1 = 0
\]

2) \( \alpha_i = C \)

\[
\mu_i = 0 \Rightarrow \xi_i > 0
\]

\[
\alpha_i > 0 \Rightarrow t_i h(\mathbf{x}_i)\beta - (1 - \xi_i) = 0 \quad \Rightarrow \quad t_i h(\mathbf{x}_i)\beta - 1 < 0
\]

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

**Remarks:** Different from unified ELM, in which most errors \( \xi_i \)'s are non-zero, in sparse ELM, errors \( \xi_i \)'s are non-zero only when the inequality constraint \( t_i h(\mathbf{x}_i)\beta - 1 > 0 \) 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 \( \beta \) 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 \( \Omega_{\text{ELM}} \)**

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: $\Omega_{\text{ELM}}$ is calculated from random hidden nodes directly.

$$h(x) = [G(a_1, b_1, x), \ldots, G(a_L, b_L, x)]$$ (23)

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

$$\Omega_{\text{ELM}} = HH^T$$ (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.

i) Sigmoid function

$$G(a, b, x) = \frac{1}{1 + \exp(-(a \cdot x + b))}$$ (25)

ii) Sinusoid function

$$G(a, b, x) = \sin(a \cdot x + b)$$ (26)

iii) Multiquadric function

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

iv) Gaussian function

$$G(a, b, x) = \exp\left(-\frac{\|x - a\|^2}{b}\right)$$ (28)

2) Kernel: $\Omega_{\text{ELM}}$ could also be evaluated by kernels as in (24). Mercer’s conditions must be satisfied. The kernel $K$ is calculated from random hidden nodes directly.

Second-order partial derivative is:

$$\frac{\partial^2}{\partial \alpha_s \partial \alpha_s} L_d = t_s \sum_{j=1}^{N} \alpha_j t_j \Omega_{\text{ELM}}(x_s, x_j) - 1$$ (32)

Thus, Hessian matrix $\nabla^2 L_d = T^T \Omega_{\text{ELM}} T$.

1) When $\Omega_{\text{ELM}}$ is calculated from random hidden nodes directly as in (24),

$$\nabla^2 L_d = T^T HH^T T = (T^T H) I_{L \times L} (T^T H)^T$$ (34)

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

2) When $\Omega_{\text{ELM}}$ is evaluated from kernels, Mercer’s conditions ensure that $\Omega_{\text{ELM}}$ is positive semi-definite. Therefore, $\nabla^2 L_d = T^T \Omega_{\text{ELM}} T \succeq 0$ is positive semi-definite.

Because 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 sub-problem 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 etc.

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. \( \alpha_i = 0 \)
   \( \alpha_i = 0 \Rightarrow t_i f(x_i) - 1 \geq 0 \)
   \( \mu_i = C \Rightarrow \xi_i = 0 \)  
   \( \frac{\partial}{\partial \alpha_i} L_d = t_c \sum_{j=1}^{N} \alpha_j t_j \Omega_{ELM}(x_c, x_j) - 1 = t_c f(x_c) - 1 \)  
   \( \frac{\partial^2}{\partial \alpha_i^2} L_d = \Omega_{ELM}(x_c, x_c) \)  
   \( J_v = \left( \frac{\partial}{\partial \alpha_v} \right) L_d \cdot 1 < 0 \)  

2. \( 0 < \alpha_i < C \)
   \( \alpha_i > 0 \Rightarrow t_i f(x_i) - 1 = 0 \)
   \( \mu_i > 0 \Rightarrow \xi_i = 0 \)  
   \( \alpha_i = C \Rightarrow t_i f(x_i) - 1 \leq 0 \)
   \( \mu_i = 0 \Rightarrow \xi_i > 0 \)  

3. \( \alpha_i = C \)
   \( \alpha_i = C \Rightarrow t_i f(x_i) - 1 \leq 0 \)
   \( \mu_i = 0 \Rightarrow \xi_i > 0 \)  

\( \alpha_i \) can only decrease. Therefore, \( d_i = -1 \).

**Definition 5.2:** \( J \) is the selection parameter:

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

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

\[ c = \arg \min_{i=1,...,N} J_i \] (42)

**Theorem 5.1:** The minimum value of \( J_i \) is negative in the training process, which guarantees that the update of the chosen Lagrange variable \( \alpha_c \) 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 \( \alpha_c \) violates optimality conditions given before. Three possible cases are:

1. \( \alpha_v = 0 \)
   \( \Rightarrow \frac{\partial}{\partial \alpha_v} L_d = t_v f(x_v) - 1 < 0 \)
   \( J_v = \left( \frac{\partial}{\partial \alpha_v} L_d \right) \cdot 1 < 0 \)  

2. \( 0 < \alpha_v < C \)
   \( \Rightarrow \frac{\partial}{\partial \alpha_v} L_d = t_v f(x_v) - 1 \neq 0 \)
   \( J_v = \left( \frac{\partial}{\partial \alpha_v} L_d \right) \cdot (-\text{sign} \left( \frac{\partial}{\partial \alpha_v} L_d \right)) < 0 \)  

3. \( \alpha_v = C \)
   \( \Rightarrow \frac{\partial}{\partial \alpha_v} L_d = t_v f(x_v) - 1 > 0 \)
   \( J_v = \left( \frac{\partial}{\partial \alpha_v} L_d \right) \cdot (1 - 1) < 0 \)  

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

\[ \text{D. Termination} \]

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 \( \varepsilon \). According to [17], \( \varepsilon = 10^{-3} \) could ensure great accuracy.

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

\[ \text{E. Convergence analysis} \]

**Theorem 5.2:** Training algorithm proposed in this paper will converge to the global optimal solution in a finite number of iterations.

**Proof:** As proved in Theorem 4.1, dual problem of sparse ELM is a convex QP problem. At every iteration, the chosen Lagrange variable \( \alpha_v \) violated optimality conditions before
the update. And as proved in Theorem 5.1, the update of $\alpha_c$ 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 $G$, $G_{i,j} = t_i t_j \Omega_{ELM}(x_i, x_j)$.

Algorithm 1 Sparse ELM for classification

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

1. Initialization:

\[ \alpha = 0, \ g = G\alpha - 1, \ J = g, \ d = 1, \ \alpha, \ g, \ J, \ d \in R^N. \]

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

1. Update $J$, $J_i = g d_i$.
2. Obtain the minimum of $J_i$, $c = \arg \min_{i=1,\ldots,N} J_i$. And update the corresponding Lagrange variable $\alpha_c$.
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, ..., 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 overfitting has been well solved, testing errors are also expected to decrease with the increase of $L$.

As shown in Fig. 2 and Fig. 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 Fig. 4 and Fig. 5 with $C = 1$ for simplicity. The relationship is consistent with our expectation.

In order to reduce human involvement, 5-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.

![Fig. 2. Training accuracy of Sparse ELM with Sinusoid nodes (Ionosphere)](image)

![Fig. 3. Testing accuracy of Sparse ELM with Sinusoid nodes (Ionosphere)](image)

![Fig. 4. Performance of Sparse ELM with Sinusoid nodes for Australian & Diabetes (C=1)](image)

![Fig. 5. Performance of Sparse ELM with Sinusoid nodes for Iris & Segment (C=1)](image)

![Fig. 6. SVM (Gaussian kernel) for Ionosphere data set](image)
C. Parameter specifications

Gaussian kernel $K(u, v) = \exp\left(-\frac{\|u-v\|^2}{2\sigma^2}\right)$ and polynomial kernel $K(u, v) = (u \cdot v + 1)^m$ are used. Fig. 6 and Fig. 7 respectively shows 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 $\sigma$ or $m$ should be chosen a priori. 5-fold cross validation method is utilized for training data and the best parameter combination is thus chosen. Parameters of $C$ and $\sigma$ are both tried with 14 different values: [0.01, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000]. And $m$ is tried with 5 values: [1, 2, 3, 4, 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: [0.01, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000]. And $m$ is specified for each problem.

D. Performance comparison

Best parameters of $C$ and $\sigma$ or $m$ chosen by cross validation method are used for training and testing. The results include average training & testing accuracy, standard deviation of training & testing accuracy, and training & 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 5 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. Because of 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. CONCLUSIONS

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: i) it achieves better generalization performance and faster training speed than SVM; ii) it requires less testing time and storage space than unified ELM. Furthermore, for large-scale binary problems, it has even faster training speed than unified ELM which has already outperformed many other methods.

REFERENCES


### Table IV

<table>
<thead>
<tr>
<th>Data sets</th>
<th>SVM (Gaussian Kernel)</th>
<th>Unified ELM (Gaussian Kernel)</th>
<th>Sparse ELM (Gaussian Kernel)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training Accuracy</td>
<td>Testing Accuracy</td>
<td>Training Time (s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Australian</td>
<td>90.96 ± 0.01</td>
<td>83.09 ± 0.01</td>
<td>0.2411 ± 0.0003</td>
</tr>
<tr>
<td>Breast Cancer</td>
<td>92.35 ± 0.01</td>
<td>97.36 ± 0.01</td>
<td>0.0550 ± 0.0010</td>
</tr>
<tr>
<td>Diabetes</td>
<td>98.65 ± 0.01</td>
<td>73.96 ± 0.01</td>
<td>0.1319 ± 0.0026</td>
</tr>
<tr>
<td>Heart</td>
<td>92.59 ± 0.01</td>
<td>82.96 ± 0.01</td>
<td>0.0385 ± 0.0007</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>93.37 ± 0.01</td>
<td>93.71 ± 0.01</td>
<td>0.0067 ± 0.0000</td>
</tr>
<tr>
<td>Mushroom</td>
<td>100 ± 0.00</td>
<td>100 ± 0.00</td>
<td>41.4878 ± 0.3148</td>
</tr>
<tr>
<td>SVMguide1</td>
<td>97.09 ± 0.00</td>
<td>96.30 ± 0.00</td>
<td>3.1869 ± 0.1115</td>
</tr>
<tr>
<td>Magic</td>
<td>89.42 ± 0.00</td>
<td>93.71 ± 0.00</td>
<td>1117.7311 ± 2.4356</td>
</tr>
<tr>
<td>* COD RNA</td>
<td>95.31 ± 0.00</td>
<td>95.25 ± 0.00</td>
<td>18588.0660 ± 11.8995</td>
</tr>
<tr>
<td>Colon Cancer</td>
<td>100 ± 0.00</td>
<td>79.37 ± 0.00</td>
<td>0.0894 ± 0.0038</td>
</tr>
<tr>
<td>Colon (Gene Sel)</td>
<td>100 ± 0.00</td>
<td>93.55 ± 0.00</td>
<td>0.0128 ± 0.0004</td>
</tr>
<tr>
<td>Leukemia</td>
<td>100 ± 0.00</td>
<td>82.35 ± 0.00</td>
<td>0.4327 ± 0.0039</td>
</tr>
<tr>
<td>Leukemia (Gene Sel)</td>
<td>100 ± 0.00</td>
<td>100 ± 0.00</td>
<td>0.0114 ± 0.0012</td>
</tr>
<tr>
<td>Spambase</td>
<td>96.61 ± 0.00</td>
<td>92.83 ± 0.00</td>
<td>0.7904 ± 0.0176</td>
</tr>
<tr>
<td>Adult</td>
<td>90.47 ± 0.00</td>
<td>84.33 ± 0.00</td>
<td>172.3218 ± 7.7212</td>
</tr>
<tr>
<td>Multiclass Classification</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iris</td>
<td>100 ± 0.00</td>
<td>93.33 ± 0.00</td>
<td>0.0253 ± 0.0009</td>
</tr>
<tr>
<td>Wine</td>
<td>100 ± 0.00</td>
<td>97.75 ± 0.00</td>
<td>0.0304 ± 0.0011</td>
</tr>
<tr>
<td>Vowel</td>
<td>99.81 ± 0.00</td>
<td>62.53 ± 0.00</td>
<td>0.6316 ± 0.0010</td>
</tr>
<tr>
<td>Segment</td>
<td>100 ± 0.00</td>
<td>91.41 ± 0.00</td>
<td>5.1300 ± 0.0036</td>
</tr>
<tr>
<td>Salimgage</td>
<td>100 ± 0.00</td>
<td>90.33 ± 0.00</td>
<td>11.3949 ± 0.0461</td>
</tr>
<tr>
<td>DNA</td>
<td>100 ± 0.00</td>
<td>94.10 ± 0.00</td>
<td>2.0669 ± 0.0106</td>
</tr>
<tr>
<td>SVMguide2</td>
<td>100 ± 0.00</td>
<td>86.41 ± 0.00</td>
<td>0.1832 ± 0.0031</td>
</tr>
<tr>
<td>USPS</td>
<td>99.88 ± 0.00</td>
<td>95.07 ± 0.00</td>
<td>20.0226 ± 1.9588</td>
</tr>
<tr>
<td>Performance of sparse ELM, unified ELM and SVM with Gaussian Kernel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data sets</td>
<td>Training Accuracy</td>
<td>Testing Accuracy</td>
<td>Time (s)</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------</td>
<td>------------------</td>
<td>----------</td>
</tr>
<tr>
<td><strong>Binary Classification</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Australian</td>
<td>89.00±0.02</td>
<td>84.62±0.03</td>
<td>0.0100</td>
</tr>
<tr>
<td>Breast Cancer</td>
<td>97.52±0.01</td>
<td>97.89±0.02</td>
<td>0.0103</td>
</tr>
<tr>
<td>Diabetes</td>
<td>82.86±0.00</td>
<td>74.32±0.00</td>
<td>0.0018</td>
</tr>
<tr>
<td>Heart</td>
<td>85.70±0.01</td>
<td>83.63±0.01</td>
<td>0.0039</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>94.46±0.01</td>
<td>90.63±0.01</td>
<td>0.0047</td>
</tr>
<tr>
<td>Mushroom</td>
<td>99.91±0.01</td>
<td>99.84±0.01</td>
<td>2.0256</td>
</tr>
<tr>
<td>SVMguide1</td>
<td>94.37±0.01</td>
<td>93.35±0.01</td>
<td>0.9222</td>
</tr>
<tr>
<td>Magic</td>
<td>82.89±0.02</td>
<td>82.84±0.02</td>
<td>11.0930</td>
</tr>
<tr>
<td>+ COD RNA</td>
<td>94.65±0.01</td>
<td>94.63±0.01</td>
<td>205.9888</td>
</tr>
<tr>
<td>Colon Cancer</td>
<td>100±0</td>
<td>83.39±0.06</td>
<td>0.0085</td>
</tr>
<tr>
<td>Colon (Gene Sel)</td>
<td>100±0</td>
<td>93.08±0.02</td>
<td>0.0024</td>
</tr>
<tr>
<td>Leukemia</td>
<td>100±0</td>
<td>76.91±0.05</td>
<td>0.0379</td>
</tr>
<tr>
<td>Leukemia (Gene Sel)</td>
<td>100±0</td>
<td>98.82±0.01</td>
<td>0.0125</td>
</tr>
<tr>
<td>Spambase</td>
<td>91.29±0.00</td>
<td>91.18±0.00</td>
<td>0.5284</td>
</tr>
<tr>
<td>Adult</td>
<td>84.46±0.00</td>
<td>84.29±0.00</td>
<td>7.8926</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Training Accuracy</th>
<th>Testing Accuracy</th>
<th>Time (s)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Multiclass Classification</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iris</td>
<td>98.67±0.00</td>
<td>97.20±0.00</td>
<td>0.0045</td>
<td>0.0046</td>
</tr>
<tr>
<td>Wine</td>
<td>100±0</td>
<td>99.16±0.01</td>
<td>0.0061</td>
<td>0.0061</td>
</tr>
<tr>
<td>Vowel</td>
<td>94.65±0.08</td>
<td>57.55±0.07</td>
<td>0.0405</td>
<td>0.0414</td>
</tr>
<tr>
<td>Segment</td>
<td>97.71±0.00</td>
<td>95.88±0.00</td>
<td>0.1809</td>
<td>0.1505</td>
</tr>
<tr>
<td>Satimage</td>
<td>92.88±0.00</td>
<td>89.89±0.00</td>
<td>3.8516</td>
<td>0.7572</td>
</tr>
<tr>
<td>DNA</td>
<td>98.06±0.00</td>
<td>93.68±0.01</td>
<td>0.3864</td>
<td>0.2724</td>
</tr>
<tr>
<td>SVMguide2</td>
<td>92.59±0.01</td>
<td>54.74±0.15</td>
<td>0.0129</td>
<td>0.0148</td>
</tr>
<tr>
<td>USPS</td>
<td>99.09±0.00</td>
<td>93.31±0.00</td>
<td>11.1521</td>
<td>1.2797</td>
</tr>
</tbody>
</table>

**TABLE VI**

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Training Accuracy</th>
<th>Testing Accuracy</th>
<th>Time (s)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Binary Classification</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Australian</td>
<td>86.84±0.00</td>
<td>85.54±0.00</td>
<td>0.0094</td>
<td>0.0064</td>
</tr>
<tr>
<td>Breast Cancer</td>
<td>98.03±0.00</td>
<td>98.56±0.00</td>
<td>0.0089</td>
<td>0.0057</td>
</tr>
<tr>
<td>Diabetes</td>
<td>83.48±0.00</td>
<td>74.49±0.00</td>
<td>0.0105</td>
<td>0.0068</td>
</tr>
<tr>
<td>Heart</td>
<td>88.07±0.01</td>
<td>83.15±0.01</td>
<td>0.0038</td>
<td>0.0019</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>94.91±0.01</td>
<td>88.09±0.01</td>
<td>0.0036</td>
<td>0.0025</td>
</tr>
<tr>
<td>Mushroom</td>
<td>99.92±0.00</td>
<td>99.88±0.00</td>
<td>1.9781</td>
<td>0.5237</td>
</tr>
<tr>
<td>SVMguide1</td>
<td>95.22±0.00</td>
<td>94.36±0.00</td>
<td>0.6956</td>
<td>0.2490</td>
</tr>
<tr>
<td>Magic</td>
<td>84.02±0.00</td>
<td>83.77±0.00</td>
<td>11.3833</td>
<td>1.4718</td>
</tr>
<tr>
<td>+ COD RNA</td>
<td>94.28±0.00</td>
<td>94.38±0.00</td>
<td>222.0702</td>
<td>9.3984</td>
</tr>
<tr>
<td>Colon Cancer</td>
<td>100±0</td>
<td>82.10±0.06</td>
<td>0.0084</td>
<td>0.0031</td>
</tr>
<tr>
<td>Colon (Gene Sel)</td>
<td>100±0</td>
<td>91.29±0.01</td>
<td>0.0015</td>
<td>0.0014</td>
</tr>
<tr>
<td>Leukemia</td>
<td>100±0</td>
<td>81.03±0.07</td>
<td>0.0345</td>
<td>0.0148</td>
</tr>
<tr>
<td>Leukemia (Gene Sel)</td>
<td>100±0</td>
<td>99.12±0.01</td>
<td>0.0029</td>
<td>0.0011</td>
</tr>
<tr>
<td>Spambase</td>
<td>90.32±0.00</td>
<td>90.87±0.00</td>
<td>0.5050</td>
<td>0.1201</td>
</tr>
<tr>
<td>Adult</td>
<td>84.80±0.00</td>
<td>84.55±0.00</td>
<td>4.6063</td>
<td>2.9173</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Training Accuracy</th>
<th>Testing Accuracy</th>
<th>Time (s)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Multiclass Classification</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iris</td>
<td>98.60±0.00</td>
<td>96.20±0.01</td>
<td>0.0043</td>
<td>0.0037</td>
</tr>
<tr>
<td>Wine</td>
<td>100±0</td>
<td>99.10±0.01</td>
<td>0.0056</td>
<td>0.0046</td>
</tr>
<tr>
<td>Vowel</td>
<td>97.25±0.00</td>
<td>97.17±0.01</td>
<td>0.0037</td>
<td>0.0028</td>
</tr>
<tr>
<td>Segment</td>
<td>96.29±0.00</td>
<td>95.28±0.00</td>
<td>0.1133</td>
<td>0.1423</td>
</tr>
<tr>
<td>Satimage</td>
<td>86.09±0.00</td>
<td>83.61±0.00</td>
<td>2.1227</td>
<td>0.6790</td>
</tr>
<tr>
<td>DNA</td>
<td>98.11±0.00</td>
<td>94.57±0.00</td>
<td>0.4008</td>
<td>0.2478</td>
</tr>
<tr>
<td>SVMguide2</td>
<td>95.58±0.01</td>
<td>97.57±0.00</td>
<td>0.0116</td>
<td>0.0133</td>
</tr>
<tr>
<td>USPS</td>
<td>97.97±0.00</td>
<td>93.56±0.00</td>
<td>6.7664</td>
<td>1.1829</td>
</tr>
</tbody>
</table>

**TABLE VII**
<table>
<thead>
<tr>
<th>Data sets</th>
<th># Total Vectors</th>
<th>SVM Gaussian Kernel</th>
<th>SVM Polynomial Kernel</th>
<th>ELM Gaussian Kernel</th>
<th>ELM Polynomial Kernel</th>
<th>ELM Sparse Nodes</th>
<th>Unified ELM Gaussian Kernel</th>
<th>Unified ELM Polynomial Kernel</th>
<th>Unified ELM Sparse Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Australian</td>
<td>345</td>
<td>306</td>
<td>109</td>
<td>240.35</td>
<td>142.35</td>
<td>127.7</td>
<td>119.7</td>
<td>345</td>
<td>345</td>
</tr>
<tr>
<td>Breast Cancer</td>
<td>342</td>
<td>80</td>
<td>41</td>
<td>68</td>
<td>51.3</td>
<td>53.2</td>
<td>52.25</td>
<td>342</td>
<td>342</td>
</tr>
<tr>
<td>Diabetes</td>
<td>384</td>
<td>213</td>
<td>202</td>
<td>278</td>
<td>204.1</td>
<td>219.7</td>
<td>217.55</td>
<td>384</td>
<td>384</td>
</tr>
<tr>
<td>Heart</td>
<td>135</td>
<td>72</td>
<td>46</td>
<td>79.45</td>
<td>58.75</td>
<td>68.8</td>
<td>68.05</td>
<td>135</td>
<td>135</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>176</td>
<td>93</td>
<td>48</td>
<td>98.95</td>
<td>85.05</td>
<td>94.13</td>
<td>101.45</td>
<td>176</td>
<td>176</td>
</tr>
<tr>
<td>Mushroom</td>
<td>4062</td>
<td>936</td>
<td>343</td>
<td>223.15</td>
<td>149.9</td>
<td>352.5</td>
<td>77.75</td>
<td>4062</td>
<td>4062</td>
</tr>
<tr>
<td>SVGguide1</td>
<td>3089</td>
<td>429</td>
<td>354</td>
<td>464.35</td>
<td>575.4</td>
<td>910.15</td>
<td>886.75</td>
<td>3089</td>
<td>3089</td>
</tr>
<tr>
<td>Magic</td>
<td>9510</td>
<td>3409</td>
<td>3190</td>
<td>5262.1</td>
<td>3599.75</td>
<td>4429.25</td>
<td>4428.6</td>
<td>9510</td>
<td>9510</td>
</tr>
<tr>
<td>Colon (Gene Sel)</td>
<td>31</td>
<td>31</td>
<td>24</td>
<td>29.3</td>
<td>25.2</td>
<td>27.6</td>
<td>26.85</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>Colon (Gene Sel)</td>
<td>31</td>
<td>30</td>
<td>17</td>
<td>25.85</td>
<td>24.6</td>
<td>29.33</td>
<td>19.75</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>Leukemia (Gene Sel)</td>
<td>38</td>
<td>12</td>
<td>7</td>
<td>38</td>
<td>22.45</td>
<td>12.3</td>
<td>11.1</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>Spambase</td>
<td>2301</td>
<td>772</td>
<td>392</td>
<td>810.95</td>
<td>1311.8</td>
<td>1586.2</td>
<td>1590.4</td>
<td>2301</td>
<td>2301</td>
</tr>
<tr>
<td>Adult</td>
<td>6414</td>
<td>2729</td>
<td>2261</td>
<td>2531.15</td>
<td>2450.85</td>
<td>2518.45</td>
<td>2666.1</td>
<td>6414</td>
<td>6414</td>
</tr>
<tr>
<td>Multiclass Classification</td>
<td>150</td>
<td>29</td>
<td>25</td>
<td>54.45</td>
<td>38.4</td>
<td>48.8</td>
<td>47.4</td>
<td>150</td>
<td>152</td>
</tr>
<tr>
<td>Wine</td>
<td>178</td>
<td>72</td>
<td>46</td>
<td>149105</td>
<td>419.9</td>
<td>54.53</td>
<td>52.25</td>
<td>178</td>
<td>178</td>
</tr>
<tr>
<td>Segment</td>
<td>69305</td>
<td>3010</td>
<td>328</td>
<td>6116.75</td>
<td>866.45</td>
<td>1532.5</td>
<td>133.05</td>
<td>69305</td>
<td>69305</td>
</tr>
<tr>
<td>Sattimage</td>
<td>22175</td>
<td>2376</td>
<td>1528</td>
<td>19197.1</td>
<td>207.3</td>
<td>2705</td>
<td>2690.2</td>
<td>22175</td>
<td>22175</td>
</tr>
<tr>
<td>DNA</td>
<td>400000</td>
<td>612</td>
<td>2237</td>
<td>5880.95</td>
<td>976.05</td>
<td>116.75</td>
<td>1048.85</td>
<td>400000</td>
<td>400000</td>
</tr>
<tr>
<td>SVGguide2</td>
<td>392</td>
<td>392</td>
<td>392</td>
<td>386</td>
<td>163</td>
<td>175.1</td>
<td>188.95</td>
<td>392</td>
<td>392</td>
</tr>
</tbody>
</table>

**TABLE VIII**

**NUMBER OF SUPPORT VECTORS**