1-1-2014

A sparsity-based training algorithm for least squares SVM

Jie Yang
University of Wollongong, jiey@uow.edu.au

Jun Ma
University of Wollongong, jma@uow.edu.au

A sparsity-based training algorithm for least squares SVM

Abstract

We address the training problem of the sparse Least Squares Support Vector Machines (SVM) using compressed sensing. The proposed algorithm regards the support vectors as a dictionary and selects the important ones that minimize the residual output error iteratively. A measurement matrix is also introduced to reduce the computational cost. The main advantage is that the proposed algorithm performs model training and support vector selection simultaneously. The performance of the proposed algorithm is tested with several benchmark classification problems in terms of number of selected support vectors and size of the measurement matrix. Simulation results show that the proposed algorithm performs competitively when compared to existing methods.

Keywords

squares, training, svm, algorithm, least, sparsity

Disciplines

Engineering | Science and Technology Studies

Publication Details

Yang, J. & Ma, J. (2014). A sparsity-based training algorithm for least squares SVM. IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2014) (pp. 345-350). United States: Institute of Electrical and Electronics Engineers.

A Sparsity-Based Training Algorithm for Least Squares SVM

Jie Yang, Jun Ma
SMART Infrastructure Facility, Faculty of Engineering and Information Sciences
University of Wollongong, Northfields Avenue, Wollongong, New South Wales 2522, Australia
E-mails: {jiey, jma}@uow.edu.au

Abstract—**We address the training problem of the sparse Least Squares Support Vector Machines (SVM) using compressed sensing. The proposed algorithm regards the support vectors as a basis dictionary and selects the important ones that minimize the residual output error iteratively. A measurement matrix is also introduced to reduce the computational cost. The main advantage is that the proposed algorithm performs model training and support vector selection simultaneously. Experimentally, the proposed algorithm is tested with several benchmark classification problem. Different numbers of support vectors and sizes of the measurement matrix are taken into account to test the performance of the proposed algorithm. Simulation results show that the proposed method performs competitively when compared to existing methods.**

I. Introduction

Least Squares Support Vector Machines (LS-SVMs) are now considered as the most popular tools for regression and classification learning tasks [**? ? ? ?**]. One of the advantages of LS-SVMs over the traditional SVMs is that the sensitive loss function is replaced by a set of equality constraints; thereby, the quadratic programming problem of traditional SVMs is reduced to solving a system of linear equations. The empirical studies have shown that LS-SVMs are comparable to standard SVMs in terms of generalization performance [**? ? ?**].

The major drawback of LS-SVMs, however, is the solution sparsity, in which a great number of support vectors (SVs) are required in the model. The support vectors, used to construct the decision function, are typically a small portion of training samples. Increasing the number of SVs will influence the training accuracy, the generalization ability, and the computation cost [**? ? ?**].

In this paper, we present a sparsity-based training algorithm for the LS-SVM model using compressed sensing, termed *Sparse Least Squares Support Vector Machine* (SLS-SVM). The compressed sensing (CS) model is used to recover signals that have a sparse representation from a number of measurements lower than the number of samples required by the Shannon/Nyquist Sampling theory [**? ? ? ? ?**]. Thus, when we are aiming for a sparse LS-SVM model, the CS model can be employed. In details, the LS-SVM model is first reformulated as a sparse representation problem. The training process is then accomplished by iteratively finding important support vectors that minimize the residual error. To further reduce the computational cost, a measurement matrix is also introduced based on compressed sensing. The main advantage of the SLS-SVM is that it performs model training and SVs selection simultaneously. By this way, it does not require a full training of the LS-SVM model before finding important SVs; therefore, it reduces the computation cost compared to most sparse training methods.

The remainder of the paper is organized as follows. Section II gives a brief introduction of the typical LS-SVM training process and the compressed sensing model. Section III presents the sparsity-based training algorithm. Section IV compares the proposed algorithm with conventional training methods using four typical classification problems. Section V presents concluding remarks.

II. LS-SVMs and compressed sensing

In this section, we first briefly review the traditional training process for the LS-SVM algorithm. Then we introduce the conceptual model for compressed sensing.

A. Traditional LS-SVM

As a supervised learning approach, an LS-SVM is commonly used in classification learning task. Suppose that we have a training set consisting of N samples $\{\mathbf{x}_i, z_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the i-th input sample and $z_i \in \{1, -1\}$ is the class label. The LS-SVM is trained by solving the following problem:

$$\min \ \mathcal{J}(\mathbf{w}, b, \mathbf{e}) = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + \frac{\gamma}{2}\sum_{i=1}^{N} e_i^2,$$
$$\text{s.t.} \quad z_i = \mathbf{w}^{\mathrm{T}}\varphi(\mathbf{x}_i) + b + e_i, \quad i = 1, 2, ..., N,$$

where $\mathbf{e}$ is the error vector, γ is a regularization parameter, and $\varphi(*)$ can be any kernel function. This problem

could be solved using the Lagrange multiplier method:

$$\min \ \mathcal{L}(\mathbf{w}, b, \mathbf{e}, \boldsymbol{\alpha}) = \mathcal{J}(\mathbf{w}, b, \mathbf{e}) + \sum_{i=1}^{N} \alpha_i [z_i - \mathbf{w}^{\mathrm{T}} \varphi(\mathbf{x}_i) - b - e_i], \quad (1)$$

where α_i $(i = 1, 2, ..., N)$ are the Lagrange multipliers, which may be positive or negative due to the equality constraints. According to the Karush-Kuhn-Tucker conditions, the minimization of Eq. (1) satisfies

$$\begin{aligned} &\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \longrightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i \varphi(\mathbf{x}_i), \\ &\frac{\partial \mathcal{L}}{\partial b} = 0 \longrightarrow \sum_{i=1}^{N} \alpha_i = 0, \\ &\frac{\partial \mathcal{L}}{\partial e_i} = 0 \longrightarrow \alpha_i = \gamma e_i, \quad i = 1, 2, ...N, \\ &\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \longrightarrow \mathbf{w}^T \varphi(\mathbf{x}_i) + b = z_i - e_i, \quad i = 1, 2, ...N. \end{aligned} \quad (2)$$

Furthermore, the above conditions lead to a linear system of equations after eliminating $\mathbf{w}$ and $\mathbf{e}$:

$$\left[\begin{array}{c|c} Q + \gamma^{-1} I_N & \mathbf{1}_N \\ \hline \mathbf{1}_N^T & 0 \end{array}\right] \left[\begin{array}{c} \boldsymbol{\alpha} \\ b \end{array}\right] = \left[\begin{array}{c} \mathbf{z} \\ 0 \end{array}\right], \quad (3)$$

where Q is the kernel matrix with $Q_{ij} = \varphi(\mathbf{x}_i)^T \varphi\left(\mathbf{x}_j\right)$, the vector $\mathbf{1}_N$ is the N-dimensional vector whose elements are equal to 1, and I_N is the $N \times N$ identity matrix.

To train the LS-SVM, the conjugate gradient (CG) algorithm was employed to solve the linear system (3) [**?** **?**]. The disadvantage of the CG-based training algorithm is that the computational complexity increases exponentially with the size of the linear system. The sequential minimization optimization (SMO) algorithm was also proposed to speed up the calculation [**?**]. Apart from the computational cost on training, the LS-SVM method also requires a large number of support vectors (SVs), which may influence the training performance and the generalization capacity. Too many SVs result in poor generalization on the test data even if it can obtain high accuracy on the training data. Therefore, several optimization methods have been suggested to improve the sparseness of the LS-SVM model. Suykens *et al*. first proposed to remove training samples that have the smallest absolute support values [**?**]. However, this method might eliminate training samples near the decision boundary, which has a negative influence on the training performance. An improved method was proposed in [**?**] where a reduced training set comprised of samples near the decision boundary is used to retrain the LS-SVM. In [**?**], SVs were eliminated by minimizing the output error after few samples have been deleted. However, the method involves the inversion of a matrix that is often singular or near singular. Another sparse training approach is proposed based on a two-step model selection of the kernel and penalty parameters [**?**]. In [**?**], a pruning algorithm is presented using the quadratic Renyi entropy. The training set is firstly divided into several subsets before computing their entropy for all of them. The subset with the larger entropy will be trained as the priority, and the sparse LS-SVM is eventually built on top of the subsets. A survey of sparse training algorithms is presented in [**?**].

B. Compressed sensing

Compressed sensing (CS) has received considerable attention recently for its ability to perform data acquisition and compression simultaneously [**?** **?** **?**]. It can be used to reconstruct a sparse approximation of a compressible signal from far fewer measurements than required by the sampling theorem. This has the advantage of reducing the amount of the data acquisition and computational time. Fewer measurements can be constructed by simply choosing a random *measurement matrix* in most cases, and the recovery of the sparse solution can still be achieved with high probability [**?** **?**].

According to the number of measurement vectors, the compressed sensing framework is categorized into follows:

1) *Single measurement vector* (SMV) model is applied when only one measurement vector is available [**?** **?**];
2) *Multiple measurement vector* (MMV) model considers more than one measurement vector simultaneously, where the solution is a two-dimensional array [**?**];

In this paper, we focus on the application of the SMV to train the LS-SVMs. Consider a signal $\mathbf{s} \in \mathbb{R}^N$ and an orthonormal basis $\Psi = [\boldsymbol{\psi}_1, \cdots, \boldsymbol{\psi}_N]$, where $\boldsymbol{\psi}_n \in \mathbb{R}^N$, for $n = 1, 2, \cdots, N$. Then the signal $\mathbf{s}$ can be expressed as follows:

$$\mathbf{s} = \sum_{n=1}^{N} x_n \boldsymbol{\psi}_n \quad \text{or} \quad \mathbf{s} = \Psi \mathbf{x}, \quad (4)$$

where $\mathbf{x} \in \mathbb{R}^N$ is the weight vector. In SMV model, the aim is to recover this vector $\mathbf{x}$ from few linear measurements $\mathbf{y} \in \mathbb{R}^M$, where $M << N$. Given a linear measurement matrix $\Phi \in \mathbb{R}^{M \times N}$, the measurement vector is given by

$$\mathbf{y} = \Phi \mathbf{s} = \mathcal{D} \mathbf{x}, \quad (5)$$

where $\mathcal{D} = \Phi\Psi$ is known as the dictionary. The SMV model can then be expressed as

$$\min S(\mathbf{x}) \quad \text{subject to} \quad \left\|\mathbf{y} - \mathcal{D}\mathbf{x}\right\|_2 < \epsilon, \quad (6)$$

where $S(\mathbf{x})$ denotes a sparsity measure, and ϵ bounds the amount of additional noise. One simple strategy of solving Eq. (6) is to minimize the l_0-norm of $\mathbf{x}$, i.e., $S(\mathbf{x}) = \|\mathbf{x}\|_0$. The l_0 pseudo-norm is the cardinality or number of

nonzero elements in $\mathbf{x}$. Thus Eq. (6) is rewritten as

$$\min \|\mathbf{x}\|_0 \quad \text{subject to} \quad \left\|\mathbf{y} - \mathcal{D}\mathbf{x}\right\|_2 < \epsilon. \tag{7}$$

III. Sparsity-based training algorithm for LS-SVM

In this section, we present a sparsity-base training algorithm for LS-SVM, termed *Sparse Least Squares Support Vector Machines (SLS-SVM)*, to solve sparsity and the computational complexity of LS-SVM. The main difference between SLS-SVM and existing techniques is that the proposed algorithm minimizes the model structure while training the LS-SVM simultaneously. By contrast, most traditional methods need to train the model before finding the sparse support vectors. In Subsection III-A, we present the sparsity-based training algorithm for LS-SVM. In Subsection III-B, we further reduce the computational cost by introducing the random measurement matrix. We finally detail the solver for the sparse representation in Subsection III-C.

A. Sparse representation for LS-SVM

The LS-SVM training algorithm aims to find the optimal parameters for the vector $[\boldsymbol{\alpha}^T, b]^T$ that satisfies Eq. (3). Note that setting a particular element of $\boldsymbol{\alpha}$ to zero is equivalent to removing the corresponding training sample or support vector. Therefore, the goal of finding a sparse LS-SVM model, within a given tolerance of accuracy, can be equated to minimizing the number of non-zero elements from the parameter vector $[\boldsymbol{\alpha}^T, b]^T$. For further discussion, we simplify Eq. (3) as follows. Let $\widehat{\mathbf{z}}$ be a composite vector containing the desired output:

$$\widehat{\mathbf{z}} = \begin{bmatrix} \mathbf{z} \\ 0 \end{bmatrix}. \tag{8}$$

The coefficient matrix is represented by Ψ:

$$\Psi = \left[\begin{array}{c|c} Q + \gamma^{-1} I_N & \mathbf{1}_N \\ \hline \mathbf{1}_N^T & 0 \end{array}\right]. \tag{9}$$

The parameter vector is represented by $\mathbf{x}$ as follows:

$$\mathbf{x} = [\boldsymbol{\alpha}^{\mathrm{T}}, b]^{\mathrm{T}}. \tag{10}$$

Note that each element from $\mathbf{x}$ is associated with one support vector. Now the sparse LS-SVM training is equivalent to finding a sparse representation for the parameter vector $\mathbf{x}$. Only the support vectors associated with non-zero parameters in $\mathbf{x}$ will affect the LS-SVM model. Consequently, the sparse training problem for LS-SVM is cast as follows:

$$\min \ \|\mathbf{x}\|_0 \quad \text{subject to} \quad \left\|\widehat{\mathbf{z}} - \Psi\mathbf{x}\right\|_2 < \epsilon. \tag{11}$$

B. Measurement matrix

The problem with the minimization of Eq. (11) is that the entire training set with all N samples is considered, which leads to costly computation, especially with large training sets.

We now further formulate the sparse LS-SVM training using fewer measurements. Given a linear measurement matrix $\Phi \in \mathbb{R}^{M\times(N+1)}$ ($M << N$), the measurement vector is given by

$$\mathbf{y} = \Phi\widehat{\mathbf{z}}, \tag{12}$$

where $\widehat{\mathbf{z}}$ is the augmented vector in Eq. (8). Given the matrix Ψ in Eq. (9), the optimization problem in (11) can be expressed as follows:

$$\min \ \|\mathbf{x}\|_0 \quad \text{subject to} \quad \left\|\mathbf{y} - \mathcal{D}\mathbf{x}\right\|_2 < \epsilon, \tag{13}$$

where $\mathcal{D} = \Phi\Psi$ is the dictionary. We should note that the dictionary $\mathcal{D}$ is of size $M \times (N + 1)$, where $M << N$. In particular, if the selection matrix with $M = N + 1$ is employed, then all the original measurements will be selected. In this case, solving Eq. (13) is converted to that of Eq. (11). As observed, the computational complexity is now reduced to solving a $M\times(N+1)$-dim linear system. This is much smaller than the original model, which has size $(N + 1) \times (N + 1)$.

C. Sparse solver via orthogonal matching pursuit

A variety of algorithms can be applied to find the sparse solution, such as Orthogonal Matching Pursuit (OMP) [**?**] and Non-convex local optimization [**?**]. Herein, we use the OMP algorithm to solve the SMV problem. The reason is two-fold: firstly, we are able to control the number of selected SVs via OMP. According to OMP, it starts from an empty set and adds a new atom iteratively for the sparse representation. If the solver halts at the K-th iteration, there will be K non-zero element in the solution. As a result, when OMP is applied to solve Eq. (13), we will have $\|\mathbf{x}\|_0 = K$ at the K-th iteration. Accordingly, there will be K support vectors being selected.

The second reason is the less computational cost. For the proposed algorithm, at each iteration, the OMP algorithm performs a Cholesky factorization, which requires $O(K^2)$ operations, where K is the number of support vectors to be selected. Furthermore, SLS-SVM selects one support vector iteratively. The selection procedure can be regarded as a forward selection of K support vectors from the entire N training samples. Therefore, K loops are required to select K support vectors. The computational complexity for the SLS-SVM algorithm is $O(K^3)$.

By contrast, the training algorithms in [**?**] train an LS-SVM using all N training examples. The LS-SVM model is trained by optimizing the kernel and penalty parameters. Therefore, the complexity is $O(N^3)$. In [**?**] , the pruning operation is applied to delete training examples until some criteria are reached. This process is computationally expensive as each deletion requires $O(P^2)$ operations, where P ($N > P > K$) is the number of training examples being compared. In addition, K loops are required to achieve the sparse model. Therefore, the

computational complexity is $O(K \times P^2)$. The comparison of the computational complexity between the proposed algorithm and other sparse training methods [**?** **?**] is summarised in Table I. Overall, compared to conventional methods, faster convergence is expected from the proposed algorithm.

TABLE I
COMPUTATIONAL COMPLEXITY OF VARIOUS TRAINING ALGORITHMS FOR LS-SVM, WHERE N IS THE NUMBER OF TRAINING SAMPLES, K IS THE NUMBER OF SELECTED SUPPORT VECTORS, AND P IS THE NUMBER OF SUPPORT VECTORS TO BE REMOVED.

	SLS-SVM	[**?**]	[**?**]
Loops	K	N	K
Complexity	$O(K^3)$	$O(N^3)$	$O(K \times P^2)$

IV. EXPERIMENTAL RESULTS

This section presents the experimental results and comparisons of the proposed algorithm with existing LS-SVM training techniques. The employed classification data sets and the evaluation of the training algorithm are presented in Subsection IV-A. The performance of the SLS-SVM is then evaluated in Subsection IV-B. The comparison results with conventional training algorithms are presented in Subsection IV-C.

A. Experimental methods

Four classification problems are chosen from UCI repository [**?**] for experimental evaluation (see Table II). Each data set is partitioned into two subsets: a training set and a test set. The training set is used to train and optimize the LS-SVM model. The test set is used for the evaluation of the generalization performance of the model. The sizes of the training and test sets are 67% and 33%, respectively.

TABLE II
EMPLOYED DATA SET. A PARTITIONING INTO TRAINING AND TEST SET IS ALSO GIVEN FOR EACH DATA SET.

Data Set	Input	Training	Test
Cancer	9	466	233
Heart	13	180	90
Thyroid	5	145	70
Wine	13	120	58

The performance of the training algorithms is evaluated using the classification accuracy. The resulting model structure is measured by the number of remaining support vectors, i.e.,

$$k = K/(N+1) \times 100\%, \tag{14}$$

where K is the number of selected support vectors, and N is the number of training samples. Thus, a larger value for k means that more support vectors are selected during training. For the proposed SLS-SVM algorithm, the percentage of selected measurements is denoted

$$m = M/(N+1) \times 100\%, \tag{15}$$

where M is the number of selected measurements.

B. Performance analysis of SLS-SVM

In this subsection, we first investigate the effect of the remaining support vectors on the generalization ability. Second, we test the performance of the SLS-SVM method based on the measurement matrix with different numbers of selected measurements.

1) Support vectors: The number of support vectors is critical to the performance of the SLS-SVM algorithm. For instance, a large model with more support vectors may result in a fast convergence to a local minimum, but exhibits poor generalization performance because of overfitting. Meanwhile, a too small model may not be able to find the proper fit to the data. Therefore, in this subsection we analyse how the number of support vectors impacts the performance of SLS-SVM. The remaining model structure is set to k = 20%, 30%, 50%, and 100%, and the percentage of measurements is fixed at m =100% with the measurement matrix. Again, using the measurement matrix with m =100% is equivalent to solving the sparse representation for Eq. (11).

Table III shows the performance of the proposed algorithm with respect to the remaining model structure. As can be observed, the proposed method achieves a better classification accuracy on the training sets with increasing number of support vectors. The performance on the training sets is 93.43%, 95.50%, 97.13%, and 99.28% for k = 20%, 30%, 50% and 100%, respectively. However, a larger k leads to a longer training time. In other words, when more support vectors are selected, higher computational cost is required to train the SLS-SVM algorithm.

On the other hand, the generalization ability of the trained model is not guaranteed to be improved with more support vectors. For instance, the average classification rate on the test set is 94.06%, 95.02%, 95.25%, 94.84% for k = 20%, 30%, 50% and 100%, respectively. One reason is that the model with more support vectors overfits the training data, and thus its accuracy is reduced on the test sets. Overall, the results show that selecting fewer support vectors leads to comparably well generalization performance and requires less computation cost.

2) Measurement matrix: A measurement matrix is employed in the proposed SLS-SVM algorithm, which is used to reduce the computational complexity. In this subsection, we analyse the robustness of the proposed algorithm to the measurement matrix. Without loss of generality, we adapt the random measurement matrix herein. Meanwhile we set the percentage of selected measurements to m = 20%, 40%, 70% and 100%. To

TABLE III
SUMMARY OF THE CLASSIFICATION ACCURACY (%) AND TRAINING TIME (SEC) FROM THE PROPOSED SLS-SVM METHOD. VARIOUS NUMBERS OF SELECTED SUPPORT VECTORS ARE CONSIDERED.

Data sets	k	20%	30%	50%	100%
Cancer	Training	98.71	99.62	**100**	**100**
	Test	**98.71**	**98.71**	**98.71**	**98.71**
	Time	**1.04±0.23**	2.08±0.75	7.03±3.11	54.68±7.58
Heart	Training	86.11	88.33	92.22	**100**
	Test	**87.78**	86.67	86.67	86.67
	Time	**0.07±0.01**	0.14±0.05	0.35±0.09	2.57±0.73
Thyroid	Training	93.56	96.07	97.98	**98.36**
	Test	93.85	96.32	97.10	**97.87**
	Time	**0.18±0.02**	0.35±0.07	0.71±0.22	4.32±1.19
Wine	Training	95.35	97.97	98.31	**98.76**
	Test	95.90	98.36	**98.51**	96.12
	Time	**0.06±0.01**	0.10±0.01	0.35±0.09	2.53±0.86
Average	Training	93.43	95.50	97.13	**99.28**
	Test	94.06	95.02	**95.25**	94.84
	Time	**0.34±0.07**	0.67±0.22	2.11±0.88	16.02±2.59

make the comparison fair, the remaining model is fixed at k =30%. Tables IV shows the performances of the SLS-SVM algorithms with the random measurements.

Several observations can be made from these results. Firstly, the SLS-SVM method achieves better training accuracy with larger m. Given the random measurement matrix, the average classification rate on the training set is 91.24%, 93.72%, 94.98%, and 95.35% for m = 20%, 40%, 70% and 100%, respectively. Obviously, higher training accuracy requires more measurements. Selecting only a very small number of measurements may not provide sufficient information for training. On the other hand, the proposed SLS-SVM algorithm achieves quite similar generalization performance with different sizes of the measurement matrix. For instance, for m = 20%, 40%, 70%, and 100%, the proposed algorithm generates on average 93.89%, 95.56%, 95.43%, and 95.17% classification accuracy on the test sets. The average difference of 1.28% classification rate on the test sets of the four problems is not significant. Second, in terms of the training time, more measurements require longer training time. When m =100%, the proposed method requires nearly 50% extra time to train the LS-SVM model compared to that of m = 20%. Overall, using fewer measurements, the proposed algorithm not only reduces the computation requirement, but also achieves similar generalization performance.

C. Comparison with sparse training algorithms

In this subsection, the proposed SLS-SVM algorithm is compared with another two sparse training algorithms, namely Selection [**?**] and Pruning [**?**]. To make a fair comparison, we implemented the proposed SLS-SVM using the least number of support vectors achieved by its counterpart. The standard LS-SVM algorithm is also considered for benchmarking. Furthermore, in SLS-SVM, the percentage of selected measurements is set to m =40% with the random measurement matrix. Table V shows the remaining model structure, the training accuracy, and the test accuracy obtained with different methods.

Compared to conventional training algorithms, the SLS-SVM achieves a significant improvement in terms of classification accuracy using the same number of support vectors. On average, the SLS-SVM achieves a classification accuracy of 95.45% on the test sets, which is better than the accuracy of Selection (93.21%) and Pruning (94.25%) methods. Furthermore, the SLS-SVM algorithm with average 37.50% support vectors performs comparably well to the standard LS-SVM. The average accuracy of the proposed method is improved by 3.62% on the test sets compared to the standard LS-SVM method.

V. CONCLUSION

In this paper, we proposed a LS-SVM training algorithm based on sparse signal representation. The kernel matrix in LS-SVM is regarded as a dictionary in the sparse representation, hence the goal for training LS-SVM is converted to simply finding the sparse parameter for the classifier. A measurement matrix is also introduced to further reduce the computational complexity. The main difference between SLS-SVM and existing techniques is that the proposed algorithm is capable of selecting important support vectors and training the LS-SVM model simultaneously. On the other hand, most traditional methods need to train the model before finding the support vectors. Experimentally, the proposed training algorithm leads to a quick convergence and a sparse model.

REFERENCES

TABLE IV

AVERAGE CLASSIFICATION ACCURACY (%) AND TRAINING TIME (SEC) OF THE PROPOSED SLS-SVM ALGORITHM, AS A FUNCTION OF THE SIZE OF THE RANDOM MEASUREMENT MATRIX.

Data sets	m	20%	40%	70%	100%
Cancer	Training	96.57±3.25	97.00±2.25	**99.14±0.65**	99.00±0.32
	Test	**99.14±0.43**	**99.14±0.43**	**99.14±0.43**	**99.14±0.43**
	Time	**1.22±0.57**	1.67±0.75	2.06±0.68	2.10±0.87
Heart	Training	83.89±4.21	86.11±2.01	87.78±0.91	**88.22±0.51**
	Test	85.56±1.01	**87.78±0.08**	87.32±0.06	86.67±0.03
	Time	**0.11±0.05**	0.13±0.07	0.14±0.08	0.14±0.07
Thyroid	Training	93.38±5.57	94.45±4.73	95.55±3.89	**96.07±3.66**
	Test	93.57±3.63	**96.87±2.28**	96.51±1.40	96.58±1.33
	Time	**0.16±0.02**	0.23±0.05	0.28±0.05	0.35±0.07
Wine	Training	91.10±6.20	97.31±5.19	97.45±3.53	**98.10±0.91**
	Test	97.30±2.51	98.46±1.20	**98.77±1.36**	98.28±1.08
	Time	**0.06±0.01**	0.08±0.01	0.08±0.01	0.10±0.01
Average	Training	91.24±4.81	93.72±3.55	94.98±2.25	**95.35±1.35**
	Test	93.89±1.90	**95.56±0.99**	95.43±0.81	95.17±0.72
	Time	**0.39±0.16**	0.53±0.22	0.64±0.21	0.67±0.26

TABLE V

SUMMARY OF REMAINING SUPPORT VECTORS AND CLASSIFICATION ACCURACY (%) FOR VARIOUS SPARSE TRAINING ALGORITHMS OF LS-SVM.

		LS-SVM	Selection [?]	Pruning [?]	SLS-SVM
Cancer	k	100	38.74	**25.20**	**25.20**
	Training	96.35	-	**98.00±0.80**	96.80±2.31
	Test	98.71	94.10±3.38	97.90±0.80	**99.14±0.43**
Heart	k	100	62.97	**46.70**	**46.70**
	Training	88.67	-	**91.70±2.10**	86.91±1.72
	Test	81.67	83.44±2.86	85.19±4.60	**87.67±1.67**
Thyroid	k	100	40.96	**32.80**	**32.80**
	Training	93.40	-	**97.30±1.40**	95.51±2.51
	Test	91.45	96.46±1.77	**96.70±3.20**	96.32±1.67
Wine	k	100	50.76	**45.30**	**45.30**
	Training	98.60	-	**99.60±0.50**	97.41±1.51
	Test	95.50	**98.83±1.34**	97.20±1.40	98.67±0.67
Average	k	100	48.36	**37.50**	**37.50**
	Training	94.26	-	**96.65±1.20**	94.16±2.01
	Test	91.83	93.21±2.34	94.25±2.50	**95.45±1.11**