SMO-Based Pruning Methods for Sparse Least Squares Support Vector Machines

Xiangyan Zeng and Xue-wen Chen, *Senior Member, IEEE*

***Abstract*—Solutions of least squares support vector machines (LS-SVMs) are typically nonsparse. The sparseness is imposed by subsequently omitting data that introduce the smallest training errors and retraining the remaining data. Iterative retraining requires more intensive computations than training a single nonsparse LS-SVM. In this paper, we propose a new pruning algorithm for sparse LS-SVMs: the sequential minimal optimization (SMO) method is introduced into pruning process; in addition, instead of determining the pruning points by errors, we omit the data points that will introduce minimum changes to a dual objective function. This new criterion is computationally efficient. The effectiveness of the proposed method in terms of computational cost and classification accuracy is demonstrated by numerical experiments.**

***Index Terms*—Least squares support vector machine, pruning, sequential minimal optimization (SMO), sparseness.**

I. Introduction

RECENTLY, least squares support vector machines (LS-SVMs) have been investigated for classification and function estimation problems [1]–[3]. Instead of solving a quadratic programming problem as in SVMs, LS-SVMs find the solutions of a set of linear equations [4]. Several numerical algorithms have been proposed for training LS-SVM: Suykens *et al.* propose an iterative algorithm based on conjugate gradient (CG) algorithms [5]; Chu *et al.* improve the efficiency of the CG algorithm using one reduced system of linear equations [6]; and Keerthi and Shevade extend the sequential minimal optimization (SMO) algorithms to solve the linear equations in LS-SVMs [7]. These numerical algorithms are computationally attractive. However, the solutions obtained from these methods are not sparse.

The sparseness is important for a classifier, as it allows for fast and accurate evaluation of new data points [8], [10]. To impose sparseness in LS-SVM solutions, pruning mechanism is generally employed, i.e., the least important points are gradually omitted from training data and the LS-SVM are retrained in terms of the remaining data. Motivated by the fact that the LS-SVM support values are proportional to the errors at the data points, Suykens *et al.* propose to prune the samples that have the smallest absolute support values [8], [9]. This is a simple and direct criterion to determine which points are omitted. Arguing that omitting data with small errors in the previous pass does not reliably predict what the errors will be after the samples have been omitted, De Kruif *et al.* propose to select the samples that bear the smallest errors when they are omitted in the next pass [10]. When no regularization is applied, i.e., only the margin is maximized without the consideration of approximation errors, the derived criterion is the absolute support value, divided by the diagonal element of the inverse of the kernel matrix. Since the kernel matrix needs to be inverted, the computational load to determine the pruning points has greatly increased using this method. The similar cases are other pruning methods for Gaussian process where calculation of the error induced by omitting a sample point involves an inverse matrix [11]. A recent paper provides a comparison of various pruning algorithms and concludes that pruning based on absolute support values is still most attractive if one takes into account both the computational costs and classification accuracy [12].

The computational load of pruning algorithms includes the costs of both determining the pruning points and retraining LS-SVMs. In these previous researches, however, the importance of retraining costs has not been studied in detail. In Suykens' CG algorithm [5], there are no intermediate variables cached. While Kruif *et al.* argue that the retraining can be done to only part of the data, their method is based on the assumption that the kernel matrix is decomposed using Cholesky decomposition [10]. However, it is impractical to store and decompose kernel matrices for large scale problems.

In this paper, we propose a new pruning scheme for sparse LS-SVMs. The pruning method is based on Keerthi's SMO formulation, which has been successfully applied to find nonsparse LS-SVM solutions [10]. We formulate SMO techniques for sparse LS-SVM solutions, since SMO is suitable for large scale problems and is convenient for timely data updating. In addition, a new criterion is proposed to select data points that introduce least changes to a dual objective function. The effectiveness of the proposed method in terms of computational cost and classification accuracy is demonstrated by numerical experiments.

The paper is organized into five sections. Section II introduces LS-SVMs. The proposed pruning algorithm is described in Section III. In Section IV, we present the experimental results. Finally, conclusions are drawn in Section V.

II. LS-SVM Classifiers

Given a training set of N data points $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)\}$ where $\mathbf{x}_i \in R^d$ is the ith input vector and $y_i \in \{\pm 1\}$ is the corresponding target. We employ the idea of

Manuscript received September 14, 2004; revised February 28, 2005. The work of X. Chen was supported by the U.S. Army Research Laboratory and the U.S. Army Research Office under Contract DAAD19-03-1-0123.

X. Zeng is with the Department of Electrical and Computer Engineering, California State University, Northridge, CA 91003 USA.

X. Chen is with the Electrical Engineering and Computer Science Department, University of Kansas, Lawrence, KS 66045 USA (e-mail: xwchen@ku.edu).

Digital Object Identifier 10.1109/TNN.2005.852239

mapping the data points into a high-dimensional Hilbert space using a nonlinear function $\varphi(\cdot)$. In addition, the dot product in that high-dimensional space is equivalent to a kernel function in the input space $K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$. A linear classifier in the new space takes the form

$$y(\mathbf{x}) = \operatorname{sign}(w \cdot \varphi(\mathbf{x}) + b). \tag{1}$$

In order to obtain the parameters w and b, LS-SVM formulates the problem as

$$\min\left(\frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{2}C\sum_i e_i^2\right) \tag{2}$$

subject to the equality constraint $y_i(\mathbf{w} \cdot \varphi(\mathbf{x}_i) + b) = 1 - e_i$, or an equivalent constraint used in [7] and [10]: $y_i - (\mathbf{w} \cdot \varphi(\mathbf{x}_i) + b) = e_i$, where C is a regularization factor and e_i is the difference between the desired output y_i and the actual output. For simplicity, we consider the problem without a bias term, as did in [7], [13], and [14]. The Lagrangian for problem (2) is

$$\Re(\mathbf{w}, e_i; \alpha_i) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{2}C\sum_i e_i^2 + \sum_i \alpha_i\left[y_i - \mathbf{w}\cdot\varphi(\mathbf{x}_i) - e_i\right] \tag{3}$$

where α_i are Lagrangian multipliers. The Karush–Kuhn–Tucker (KKT) conditions for optimality are

$$\begin{cases} \frac{\partial \Re}{\partial w} = 0 & \mathbf{w} = \sum_i \alpha_i \varphi(\mathbf{x}_i) \\ \frac{\partial \Re}{\partial e_i} = 0 & \alpha_i = C e_i \\ \frac{\partial \Re}{\partial \alpha_i} = 0 & y_i - \mathbf{w}\cdot\varphi(\mathbf{x}_i) - e_i = 0 \end{cases}. \tag{4}$$

If the bias term is used in the Lagrangian (2), the KKT condition will lead to a constraint $\sum_i \alpha_i = 0$. In this case, at least two Lagrangian multipliers must be updated. Using the constraint without the bias term makes it possible to update one component per iteration. In the numerical solution, the KKT conditions are reduced to a linear system

$$\left(\mathbf{K} + \frac{\mathbf{I}}{C}\right)\boldsymbol{\alpha} = \mathbf{y} \tag{5}$$

where $\mathbf{y} = [y_1, y_2, \cdots, y_N]^T$, $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_N]^T$, and $\mathbf{K} \in R^{N\times N}$ is the kernel matrix. LS-SVM greatly simplifies the problem by characterizing the solution as a linear system. For large scale problems, linear system (5) can be efficiently solved using CG method [5], [6]. A drawback of LS-SVM is that the sparseness is lost in its solution: from the KKT conditions (4), every data point contributes to the model and the relative importance of a data point is reflected by its support value. The sparseness is usually imposed by a pruning process [8].

III. SMO Based Pruning Methods for Sparse LS-SVMs

Pruning is to remove less meaningful data and retrain LS-SVM on the basis of the remaining points. In fact, it is an iterative scheme where in each step one has to solve a KKT system [9]. Although the size of the linear system decreases from step to step, pruning is much more costly than training a nonsparse LS-SVM. A crucial concern for the pruning algorithms is the computational cost, which is related to the determination of the pruning points and the implementation of the iterative retraining. In the following, we describe a fast pruning scheme based on SMO formulation and a new criterion for determination of the pruning points.

A. SMO Algorithms for LS-SVM

SMO decomposes the optimization problem and solves the smallest possible optimization problem at every step. In standard SVM, it works by optimizing two Lagrangian multipliers at a time with the others fixed [15]. When applied to pruning LS-SVM solutions, SMO can be simplified to optimize one multiplier at a time [7]. We now detail the SMO formulation in the LS-SVM pruning process. By substituting the KKT conditions (4) into the Lagrangian (3), the dual problem is to maximize the flowing objective function

$$\max(L(\boldsymbol{\alpha})) = -\frac{1}{2}\sum_j\sum_i \alpha_i\alpha_j Q(\mathbf{x}_i, \mathbf{x}_j) + \sum_i \alpha_i y_i \tag{6}$$

where $\mathbf{Q}(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j) + \sigma_{ij}/C$, and $\sigma_{ij} = 1$ if $i = j$ and 0 otherwise. The KKT conditions for the dual problem are $\partial L/\partial\alpha_i = 0$, which lead to $y_i - \sum_j \alpha_j Q(\mathbf{x}_i, \mathbf{x}_j) = 0, \forall i$.

Define

$$F_i = -\frac{\partial L}{\partial \alpha_i} = -y_i + \sum_{j=1}^{N} \alpha_j Q(\mathbf{x}_i, \mathbf{x}_j). \tag{7}$$

Then, the optimality condition is violated if there exists any index point i that $F_i \neq 0$.

The SMO algorithm works by optimizing only one α_i at a time keeping the others fixed, i.e., $\boldsymbol{\alpha}$ is adjusted by a step t as follows:

$$\alpha_i^{\text{new}} = \alpha_i + t; \text{ and } \alpha_j^{\text{new}} = \alpha_j, \ \forall j \neq i. \tag{8}$$

The update of α_i causes the change of all the F_j

$$F_j^{\text{new}} = F_j + (\alpha_i^{\text{new}} - \alpha_i)\, Q(x_i, x_j), \ \forall j \tag{9}$$

and, thus, the dual objective function value.

To derive the optimal step t and the stopping conditions, we define the dual as a function of t

$$f(t) = L(\alpha^{\text{new}}(t)) \quad \text{and} \quad f(0) = L(\alpha). \tag{10}$$

Maximizing $f(t)$ leads to the optimal step

$$t = \frac{-F_i}{Q(x_i, x_i)} \tag{11}$$

and the change to the dual is

$$f(\alpha_i^{\text{new}}) - f(\alpha_i) = \frac{F_i^2}{2Q(x_i, x_i)}. \tag{12}$$

Therefore, at each step of the SMO algorithm, one chooses a point i that has the largest value of $F_i^2/2Q(x_i, x_i)$ and update

α_i using (8) and (11). The Duality gap is used as a termination criterion for the iterative algorithm

$$Dgap == \Re - L = \sum_i \left(\alpha_i \left(F_i - \frac{\alpha_i}{2C} \right) + \frac{C}{2} e_i^2 \right) \leq \varepsilon L \tag{13}$$

where $\Re$ is the primal cost defined in (3), L is the dual objective function value, and ε is a positive constant. The error term can be computed by

$$e_i = y_i - \mathbf{w} \cdot \varphi(\mathbf{x}_i) = \frac{\alpha_i}{C} - F_i. \tag{14}$$

Compared to CG methods, SMO is more computationally efficient for large scale problems. F_j and $Q(x_j, x_j)$ are cached in SMO algorithm, which allows for implementing the efficiently.

B. Pruning Algorithms

The criterion for determination of pruning points is a crucial factor in pruning process. In this section, we detail a new criterion that is directly based on the dual objective function and easy to compute in SMO formulation.

To derive the proper criterion for pruning, the dual objective function (6) is rewritten using the definition of F_i

$$L(\boldsymbol{\alpha}) = \frac{1}{2} \sum_i \alpha_i (y_i - F_i). \tag{15}$$

If a sample k is removed from the training set, the new objective function is given as

$$L(\boldsymbol{\alpha}') = \frac{1}{2} \sum_{i \neq k} \alpha_i \left(y_i - F_i' \right) \tag{16}$$

where $\boldsymbol{\alpha}' = (\alpha_1, \cdots, \alpha_{k-1}, \alpha_{k+1}, \cdots, \alpha_N)$ and $F_i' = F_i - \alpha_k Q(x_i, x_k)$.

Along with the idea of SMO, we consider that the removal of a sample k does not directly affect the support values of other samples, but it introduces the update of all F_i, which leads to a difference in the objective function

$$\begin{aligned} d(L) &= \frac{1}{2} \sum_i \alpha_i (y_i - F_i) - \frac{1}{2} \sum_{i \neq k} \alpha_i \left(y_i - F_i' \right) \\ &= \frac{1}{2} \sum_{i \neq k} \alpha_i (y_i - F_i) + \frac{1}{2} \alpha_k (y_k - F_k) \\ &\quad - \frac{1}{2} \sum_{i \neq k} \alpha_i \left(y_i - F_i' \right) \\ &= \frac{1}{2} \sum_{i \neq k} \alpha_i \left(F_i' - F \right) + \frac{1}{2} \alpha_k (y_k - F_k) \\ &= - \frac{1}{2} \left(\sum_i \alpha_i \alpha_k Q(x_i, x_k) - \alpha_k^2 Q(x_k, x_k) \right) \\ &\quad + \frac{1}{2} \alpha_k (y_k - F_k) \\ &= \frac{1}{2} \alpha_k^2 Q(x_k, x_k) - \alpha_k F_k. \end{aligned} \tag{17}$$

To minimize the change of the dual objective function due to the pruning, we omit the sample k that has the smallest value of $(1/2)\alpha_k^2 Q(x_k, x_k) - \alpha_k F_k$. This is consistent with the optimization goal of LS-SVM to maximize the objective function. Since F_j and $Q(x_j, x_j)$ are cached in SMO algorithm, the retraining is expected to be more computationally efficient with the proposed criterion.

Note that pruning one sample and then retraining is not efficient for large scale problems. Therefore, the retraining is done after a set of samples are removed. Defining the number of samples left out in a retraining loop as pruning step (PS), we give the pruning algorithm as follows.

Step 1) Train the initial nonsparse LS-SVMs using the SMO formulation as described in Section III-A.

Step 2) Repeat Step 3) and Step 4) until the defined termination condition is satisfied.

Step 3) Repeat the following inner loop by ps times:

- remove a sample k from the training set using criterion (17);
- update F_i, $\forall i \neq k$, of the remaining samples in the training set using $F_i' = F_i - \alpha_k Q(x_i, x_k)$, where k is the omitted data point.

Step 4) Retrain the LS-SVM using the SMO formulation based on the support values $\boldsymbol{\alpha}$ and the updated $\mathbf{F}$ of the remaining data set, where $\boldsymbol{\alpha}' = (\alpha_1, \cdots, \alpha_{k-1}, \alpha_{k+1}, \cdots, \alpha_N)$, and $\mathbf{F}' = (F_1', \cdots, F_{k-1}', F_{k+1}', \cdots, F_N')$.

The pruning iteration is terminated when a fixed percentage of training data is omitted, or when a criterion exceeds a predefined threshold. The termination criterion can be cross-validation error or an error term associate with the support value of the omitted point. To simplify the comparison of different pruning methods, we use the computational costs in pruning a fixed percentage of data. In conventional pruning schemes, the PS (3) only omits selected data points without updates to the remaining samples. Our algorithm updates F_i of the remaining samples after each point is omitted. Therefore, the criterion used to determine the pruning points is computed based on the timely updated $\boldsymbol{F}$. For the retraining, we adopt the alpha seeding technique to make use of the results from the present loop to initialize the next one [16]. The initial estimates of the alpha values in the LS-SVM training are set not to zero but to the support values of the previous training.

IV. Numerical Experiments

In this section, the proposed method is implemented and tested on three benchmark datasets: Image, Splice and Waveform. Detailed information about the datasets can be found at http://ida.first.gmd.de/~raetsch/data/benchmarks.htm. The radial basis function $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2)$ is used as the kernel function. For each dataset, we use the parameter σ^2 suggested in [6] and [7]. We will compare the proposed pruning methods to CG methods in terms of both computational cost and classification accuracy.

TABLE I
COMPUTATIONAL COSTS OF NONSPARSE LS-SVM MEASURED BY NUMBER OF KERNEL EVALUATIONS: EACH UNIT CORRESPONDS TO 10^6 EVALUATIONS. $\sigma^2_{\text{image}} = 2.7183$, $\sigma^2_{\text{splice}} = 29.9641$, AND $\sigma^2_{\text{waveform}} = 1.8221$

Data Sets	C=1		C=10		C=100		C=1000	
	SMO	CG	SMO	CG	SMO	CG	SMO	CG
Image Training-size=1300, dimension=18	7.19	30.42	34.84	89.57	263.75	284.76	2000.73	948.09
Splice Training-size=1000, dimension=60	3.15	7.50	5.84	21.00	11.78	43.00	37.39	60.50
Waveform Training-size=400, dimension=21	0.69	1.36	3.56	3.84	16.02	9.6	33.52	15.6

TABLE II
COMPUTATIONAL COSTS OF PRUNING 780 SAMPLES FROM IMAGE DATASET MEASURED BY NUMBER OF KERNEL EVALUATIONS: EACH UNIT CORRESPONDS TO 10^6 EVALUATIONS. THE ABSOLUTE SUPPORT VALUE (SMO-SV) AND THE PROPOSED CRITERION (SMO-NEW) ARE USED IN SMO PRUNING WITH $C = 10$

Pruning step	60	50	40	30	25	20	15	10	5
SMO-new	26.86	28.23	33.11	37.53	40.39	42.91	46.03	54.13	65.26
SMO-sv	65.54	70.52	86.08	99.01	106.40	119.77	137.75	155.51	186.73
CG	299.52	345.27	461.45	587.73	679.52	870.25	1133.92	1652.99	3104.92

A. Computational Costs of Pruning

In the following experiments, the SMO and CG algorithms are used to train initial nonsparse LS-SVM and then the corresponding pruning schemes are applied to omitting the less important points. For the CG algorithm, selection of pruning points is based on the absolute support values and alpha seeding is also utilized in the CG algorithm [17]. For the SMO algorithm, two criteria are tested for choosing pruning points. One is the absolute support value (SMO-sv) and the other is the proposed criterion (SMO-new). Although the same algorithm is used for SMO-sv and SMO-new, different samples are selected and omitted by the two criteria. Therefore, the retraining time will be different for these two methods due to the convergence speed. All the three schemes do not need additional computational cost to determine pruning points.

As a basis for the comparisons, Table I shows the computational costs of nonsparse LS-SVM solutions of SMO and CG algorithms at different values of parameter C. Since the extremely small and large C values are usually of little interest, we list the computational costs at 1, 10, 100 and 1000. The results are consistent with Keerthi's conclusion that SMO algorithm is efficient at medium C values, but its increase in computational costs at large C values is sharper than CG algorithm [7].

An important factor influencing the computational costs of pruning is the PS. Generally, a smaller PS is computationally less efficient, because given a fixed number of samples to be pruned away, smaller PS results in a larger number of retraining loops. We have carried out experiments to test the increase of pruning costs with a decrease of the PS. As an example, the costs of pruning 780 samples from Image dataset at different PS are shown in Table II, where C is set to 10. In SMO pruning, each loop of retraining is to modify α_j based on the cached F_j, $j = 1, 2, \cdots, N$ and, thus, converges faster for smaller PS. On the other hand, CG pruning does not cache intermediate results and most calculation needs to be repeated. Therefore, the increase in the computational cost of CG pruning at small PS values is much sharper. Furthermore, the SMO-new algorithm is more than two times faster than the SMO-sv at all the PS.

TABLE III
CLASSIFICATION ERROR RATES OF CG PRUNING ON INDEPENDENT TEST DATASETS (THE ERROR RATES WITH THE FULL TRAINING DATASETS ARE TAKEN AS 1). 70% OF THE TRAINING DATA ARE REMOVED AT DIFFERENT PS. $C = 10$

Pruning step / Dataset	1%	2%	5%	7%	10%	14%
Image	1.099	1.331	1.562	5.266	12.783	12.785
Splice	1.320	1.284	1.348	1.529	1.4837	1.7283
Waveform	1.182	1.234	1.256	1.321	1.369	1.347

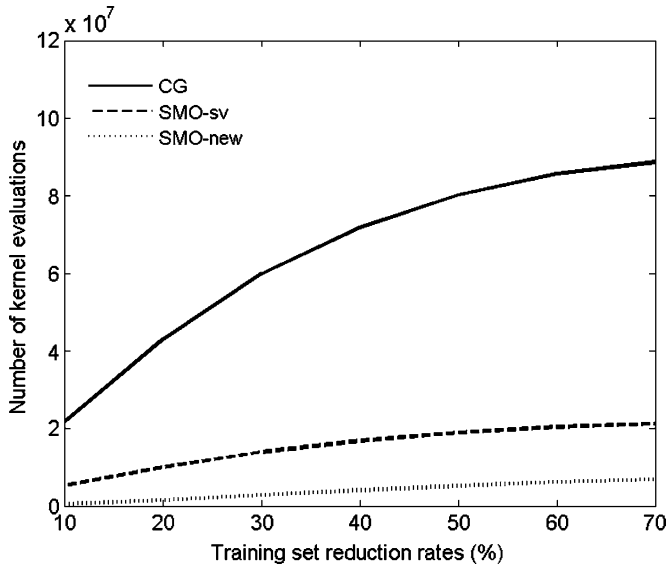

Fig. 1. Computational costs of pruning image dataset with $C = 1$.

Although larger PS lead to fewer numbers of retraining and, thus, lower computational costs, the classification performance often degrades with the increase of PS. Table III shows the classification performance of different PS, where 70% of the training data are removed and the error rates of the remaining datasets have been divided by the error rates of the full training datasets. With a reasonable PS of 5% of training data, we compare the performance of the three methods on the three datasets. Starting from the complete nonsparse LS-SVM solution, we omitted 10%, 20%, $\cdots$, 70% of training data sequentially. Generally, SMO algorithms outperform the CG algorithm and SMO-new is the fastest as shown in Fig. 1 which gives a plot of the computational costs of pruning Image dataset with $C = 1$.

The differences between the three methods are considerably large for small to medium C values. The computational costs with different C values of 1, 10, 100, and 1000 are summarized in Table IV. For instance, in the case of Image dataset with reduction rates $= 10\%$ and C $= 1$, SMO-new algorithm is 40 times faster than CG and is 10 times faster than SMO-sv. On Waveform dataset of small size, CG pruning is slightly faster than SMO pruning at large C values. Thus, the cost superiority

TABLE IV
COMPUTATIONAL COSTS OF PRUNING BENCHMARK DATASETS MEASURED BY NUMBER OF KERNEL EVALUATIONS: EACH UNIT CORRESPONDS TO 10^6 EVALUATIONS. THE PS IS EQUAL TO %5 TRAINING SAMPLES. THE ABSOLUTE SUPPORT VALUE (SMO-SV) AND THE PROPOSED CRITERION (SMO-NEW) ARE USED IN SMO PRUNING

			Training set reduction rates by pruning (%)						
			10	20	30	40	50	60	70
Image	C=1	SMO-new	0.52	1.51	2.82	4.04	5.20	6.17	6.91
		SMO-sv	5.42	10.15	13.99	16.89	18.99	20.41	21.23
		CG	21.62	42.96	59.86	71.76	80.17	85.67	88.73
	C=10	SMO-new	2.39	6.53	11.47	17.04	21.83	25.72	28.42
		SMO-sv	17.09	31.44	42.32	51.49	57.59	61.41	64.21
		CG	72.88	139.58	192.06	230.11	256.34	272.71	281.53
	$C=10^2$	SMO-new	15.35	46.54	85.79	126.17	173.07	210.69	236.67
		SMO-sv	50.97	112.37	143.55	187.49	237.15	278.65	309.13
		CG	244.69	459.31	630.60	755.37	840.22	889.61	916.24
	$C=10^3$	SMO-new	175.07	404.95	747.03	1150.51	1484.63	1783.62	2015.27
		SMO-sv	208.94	488.20	898.85	1251.61	1660.12	1977.85	2235.67
		CG	837.92	1563.43	2114.39	2505.31	2764.51	2915.75	2999.41
Splice	C=1	SMO-new	1.38	3.00	4.65	5.91	6.78	7.34	7.64
		SMO-sv	2.95	5.35	7.21	8.59	9.56	10.19	10.59
		CG	6.80	12.94	17.67	21.19	23.68	25.12	25.97
	C=10	SMO-new	2.29	5.03	7.34	9.26	10.72	11.71	12.29
		SMO-sv	5.11	9.15	12.32	14.74	16.47	17.59	18.24
		CG	18.79	34.09	46.72	55.51	60.84	63.66	65.21
	$C=10^2$	SMO-new	4.31	9.07	12.76	15.60	17.57	19.02	19.79
		SMO-sv	8.27	14.81	20.19	24.01	26.72	28.24	29.18
		CG	42.53	78.36	105.20	121.96	130.63	134.82	136.75
	$C=10^3$	SMO-new	6.30	13.47	18.48	22.31	24.91	26.90	27.85
		SMO-sv	10.91	19.85	27.12	31.84	36.67	38.61	39.80
		CG	57.68	105.41	138.14	160.39	173.63	178.30	180.44
Waveform	C=1	SMO-new	0.09	0.28	0.45	0.62	0.77	0.88	0.95
		SMO-sv	0.62	1.13	1.52	1.81	2.03	2.18	2.27
		CG	1.02	2.00	2.80	3.42	3.86	4.13	4.29
	C=10	SMO-new	0.87	1.96	3.14	4.20	4.95	5.48	5.85
		SMO-sv	1.98	3.39	4.51	5.55	6.39	6.96	7.34
		CG	3.89	7.26	9.96	11.93	13.26	14.03	14.45
	$C=10^2$	SMO-new	5.09	11.75	18.01	22.86	26.54	29.07	30.72
		SMO-sv	6.35	13.29	19.58	24.76	28.74	31.42	33.56
		CG	10.25	18.91	25.47	30.17	33.38	35.23	36.20
	$C=10^3$	SMO-new	15.40	33.41	48.28	59.16	67.50	72.96	76.38
		SMO-sv	17.22	35.86	50.37	61.96	70.11	76.11	79.56
		CG	16.97	30.98	41.37	48.47	53.05	55.69	57.03

of SMO to CG is more significant in pruning than it is in nonsparse training. Considering the relative low efficiency of the nonsparse SMO at large C values and on small size datasets, we note that the proposed pruning algorithm performs well in these cases.

It is interesting to analyze the change of the computational costs at different reduction rates. Subtracting the adjacent columns gives the computational costs of pruning 10% data at different stages. From Table III, for the CG algorithm with C = 1 and the image dataset, the cost is 21.62 for pruning the first 10% data, 21.34 for the second 10% data, 16.9 for the third 10% data, and 3.06 for the last 10% data. Thus, the cost of pruning the first 10% training data is the most expensive due to the large number of training data. Similar results are observed at most C values for SMO-sv algorithms. For SMO-new algorithms, however, the cost of pruning the first 10% data is less expensive in most cases. Fast convergence of retraining in the proposed method indicates that the omitted samples have little information. In pruning algorithms, the samples omitted in the $(i-1)$th loop should be less important than those omitted in the ith loop. Therefore, the $(i-1)$th loop converges faster than the ith loop, if we ignore the data size difference. Since the number of training data is decreased sequentially, the computational costs determined by the previous two factors have different peak points in different cases. For example, for image dataset at C = 10, the SMO-sv algorithm has a cost peak of 17.09 at 10% and the SMO-new algorithm has a peak of 5.57 at 40%. Since both algorithms start from the same nonsparse solution, the later peak points indicate that the samples are omitted by the order of importance and, thus, the computational costs are less affected by the training data size. The results demonstrate that the proposed criterion outperforms the absolute support value in selecting least important samples to omit.

TABLE V
CLASSIFICATION ERROR RATES OF INDEPENDENT TEST DATASETS. THE TRAINING SETS ARE REDUCED AT A PS OF 5% TRAINING SAMPLES. THE OPTIMAL PARAMETER C IS DETERMINED BY 20-FOLD CROSS VALIDATION

Error rate(%)		Training set reduction rates by pruning (%)							
		0	10	20	30	40	50	60	70
Image	SMO-new	2.07	2.07	2.07	2.07	2.07	1.98	1.98	2.47
	SMO-sv	2.07	2.07	2.17	2.37	2.22	2.22	2.22	2.67
	CG	2.07	2.07	2.07	2.07	1.98	2.07	2.17	4.85
Splice	SMO-new	10.34	10.34	10.43	10.34	10.39	10.39	11.40	13.79
	SMO-sv	10.34	10.11	10.16	10.20	10.80	10.98	11.67	17.56
	CG	10.34	10.34	10.39	10.62	10.85	10.94	11.67	13.83
Waveform	SMO-new	10.08	10.08	10.13	10.19	10.13	10.30	10.56	10.43
	SMO-sv	10.08	10.15	10.36	10.39	10.54	10.67	10.34	10.54
	CG	10.08	10.08	10.21	10.19	10.28	10.39	10.52	10.47

B. Classification Performance

Next, we briefly compare the generalization performance of the three pruning methods. The 20-fold cross-validation error is carried out to determine the optimal parameter C for the initial nonsparse LS-SVM. Starting from the complete nonsparse LS-SVM solution, 10%, 20%, $\cdots$, 70% samples are sequentially omitted from the training set at a PS of 5% training samples. The generalization performance is evaluated by the error rates of an independent test set for each dataset, as shown in Table V. From the results, we note that the accuracy differences among the three methods are very small. Considering the computational costs, SMO pruning based on the proposed criterion is apparently the most effective. For all the three methods, the classification accuracies start to degrade after a number of PS. To improve the accuracy, we can update the regularization parameter as pointed out in [9].

The sparseness imposed by the pruning methods can be naturally provided by SVM. Next we briefly compare the LS-SVM pruning method and the SMO algorithm of SVM in terms of classification accuracy, computational costs and training set reduction ability. For the pruning method, the computational cost is measured by the number of kernel evaluations involved in the initial training and the pruning process. For SVM, the training set reduction rate is determined by $1-R$, where R is the ratio between the number of support vectors and the number of training data.

From Table VI, we can see that the SMO-new method can possibly reduce the needed training samples significantly [for example, for waveform data, the SMO-new method needs only 10% training samples, which is much smaller compared to standard SVM methods (34%)] and achieves approximately the same classification accuracy as SVM methods.

TABLE VI
CLASSIFICATION ACCURACY, COMPUTATIONAL COSTS, AND TRAINING SET REDUCTION ABILITY OF THE LS-SVM PRUNING METHOD AND THE SVM METHOD

Error rate(%)	Error Rate		Training set reduction rates (%)		Number of kernel evaluations	
	SMO-new	SVM	SMO-new	SVM	SMO-new	SVM
Image	2.47	2.38	70.00	79.00	63.26	170.85
Splice	10.39	10.11	50.00	37.80	16.91	20.98
Waveform	11.23	10.93	90.00	64.00	5.45	5.09

V. CONCLUSION

In this paper, we propose a new pruning scheme for LS-SVM. Our algorithm is based on SMO formulation, which makes it possible to use the results of previous passes and leads to efficient retraining. Furthermore, a new criterion is proposed to determine the pruning samples in the SMO algorithm. The new criterion minimizes the change of the dual objective function introduced in pruning, which is easy to calculate and leads to a faster retraining than the absolute support values. Simulation experiments have been carried out on three benchmark datasets, which demonstrate that the proposed method is significantly faster than the pruning based on CG algorithm for large scale problems.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their valuable comments.

REFERENCES

[1] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 1999.
[2] T. Van Gestel, J. A. K. Suykens, D.-E. Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, B. De Moor, and J. Vandewalle, "Financial time series prediction using least squares support vector machines within the evidence framework," *IEEE Trans. Neural Netw.*, vol. 12, no. 4, pp. 809–821, Jul. 2001.
[3] J. A. K. Suykens, T. Van Gestel, D. Baestaens, and J. Vandewalle, "A support vector machine formulation to PCA analysis and its kernel version," *IEEE Trans. Neural Netw.*, vol. 14, no. 2, pp. 447–450, Mar. 2003.
[4] L. V. Ferreira, E. Kaszkurewicz, and A. Bhaya, "Solving systems of linear equations via gradient systems with discontinuous righthand sides: application to LS-SVM," *IEEE Trans. Neural Netw.*, vol. 16, no. 2, pp. 501–505, Mar. 2005.
[5] J. A. K. Suykens, L. Lukas, P. Van Dooren, B. De Moor, and J. Vandewalle, "Least squares support vector machine classifiers: a large scale algorithm," in *Proc. Eur. Conf. Circuit Theory and Design (ECCTD'99)*, Stresa, Italy, 1999, pp. 839–842.
[6] W. Chu, C. J. Ong, and S. S. Keerthi, "An improved conjugate gradient scheme to the solution of least squares SVM," *IEEE Trans. Neural Netw.*, vol. 16, pp. 498–501, 2005.
[7] S. S. Keerthi and S. K. Shevade, "SMO algorithm for least squares SVM formulations," *Neural Computat.*, vol. 15, pp. 487–507, 2003.
[8] J. A. K. Suykens, L. Lukas, and J. Vandewalle, "Sparse approximation using least squares support vector machines," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS'00)*, Geneva, Switzerland, May 2000, pp. II757–II760.
[9] J. A. K. Suykens, J. De Barbanter, L. Lukas, and J. Vandewalle, "Weighted least squares support vector machines: robustness and sparse approximation," *Neurocomput.*, vol. 48, no. 1–4, pp. 85–105, 2002.
[10] B. J. de Kruif and T. J. de Vries, "Pruning error minimization in least squares support vector machines," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 696–702, May 2003.
[11] L. Csato and M. Opper, "Sparse online Gaussian process," *Neural Computat.*, vol. 14, no. 3, pp. 641–668, 2002.
[12] L. Hoegaerts, J. A. K. Suykens, J. Vandewalle, and B. De Moor, "A comparison of pruning algorithms for sparse least squares support vector machines," in *Proc. 11th Int. Conf. ICONIP*, Calcutta, India, Nov. 22–25, 2004.
[13] C. Saunders, A. Gammerman, and V. Vovk, "Ridge regression learning algorithm in dual variables," in *Proc. 15th Int. Conf. Machine Learning*, Madison, WI, 1998, pp. 515–521.
[14] T. Van Gestel, J. A. K. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, and J. Vandewalle, "Bayesian framework for least squares support vector machine classifiers, Gaussian processes, and kernel discriminant analysis," *Neural Computat.*, vol. 14, no. 5, pp. 1115–1147, 2002.
[15] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines,", Tech. Rep. MSR-TR-98-14, 1998.
[16] D. Decoste and K. Wagstaff, "Alpha seeding for support vector machines," in *Proc. Int. Conf. Knowledge Discovery Data Mining*, 2000, pp. 345–349.
[17] B. Hamers, J. A. K. Suykens, and B. De Moor, "A comparison of iterative methods for least squares support vector machine classifiers," ESAT-SISTA, K. U. Leuven, Leuven, Belgium, Internal Rep. 01-110, 2001.

Xiangyan Zeng received the B.E. and M.E. degrees in computer applications from Hefei University of Technology, Hefei, China, and the M.E. degree in electrical engineering and the Ph.D. degree in computer science, both from the University of the Ryukyus, Japan.

She is currently a Postdoctoral Fellow in the Department of Electrical and Computer Engineering, California State University, Northridge. Her research interests include machine learning, pattern recognition, and image processing.

Xue-wen Chen (M'00–SM'04) received the Ph.D. degree from Carnegie Mellon University, Pittsburgh, PA, in 2001.

He is Currently an Assistant Professor of computer science at the University of Kansas, Lawrence. His research interest includes machine learning, statistical modeling, and bioinformatics.