A Novel Model of Working Set Selection for SMO Decomposition Methods

Zhen-Dong Zhao[1] Lei Yuan[2] Yu-Xuan Wang[2] Forrest Sheng Bao[2] Shun-Yi Zhang[1] Yan-Fei Sun[1]
1.Institution of Information & Network Technology,
Nanjing University of Posts and Telecommunications, Nanjing, 210003, CHINA
2.School of Communications and Information Engineering,
Nanjing University of Posts and Telecommunications, Nanjing, 210003, CHINA
{zhaozhendong, lyuan0388, logpie, forrest.bao}@gmail.com {dirzsy, sunyanfei}@njupt.edu.cn

***Abstract*— In the process of training Support Vector Machines (SVMs) by decomposition methods, working set selection is an important technique, and some exciting schemes were employed into this field. To improve working set selection, we propose a new model for working set selection in sequential minimal optimization (SMO) decomposition methods. In this model, it selects B as working set without reselection. Some properties are given by simple proof, and experiments demonstrate that the proposed method is in general faster than existing methods.**

***Index Terms*— support vector machines, decomposition methods, sequential minimal optimization, working set selection**

I. Introduction

In the past few years, there has been huge of interest in Support Vector Machines (SVMs) [1], [2] because they have excellent generalization performance on a wide range of problems. The key work in training SVMs is to solve the follow quadratic optimization problem.

$$\begin{aligned} &\min_{\alpha B} f(\alpha) = \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ \text{subject to}\quad & 0 \le \alpha_i \le C, i = 1, \dots, l \\ & y^T\alpha = 0 \end{aligned} \tag{1}$$

where e is the vector of all ones, C is the upper bound of all variables, and $Q_{ij} = y_i y_j K(x_i, x_j)$, $K(x_i, x_j)$ is the kernel function.

Notable effects have been taken into training SVMs [3], [4], [5], [6]. Unlike most optimization methods which update the whole vector α in each iteration, the decomposition method modifies only a subset of α per iteration. In each iteration, the variable indices are split into a "working set": $B \subseteq \{1, \dots, l\}$ and its complement $N = \{1, \dots, l\} \setminus B$. Then, the sub-problem with variables $x_i, i \in B$, is solved, thereby, leaving the values of the remaining variables $x_j, j \in N$ unchanged. This method leads to a small sub-problem to be minimized in each iteration. An extreme case is the Sequential Minimal Optimization (SMO) [5], [7], which restricts working set to have only two elements. Comparative tests against other algorithms, done by Platt [5], indicates that SMO is often much faster and has better scaling properties.

Since only few components are updated per iteration, for difficult problems, the decomposition method suffers from slow convergence. Better method of working set selection can reduce the number of iterations and hence is an important research issue. Some methods were proposed to solve this problem and to reduce the time of training SVMs [8]. In this paper, we propose a new model to select the working set. In this model, specially, it selects B without reselection. In another word, once $\{\alpha_i, \alpha_j\} \subset B$ are selected, they will not be tested or selected during the following working set selection. Experiments demonstrate that the new model is in general faster than existing methods.

This paper is organized as following. In section II, we give literature review, SMO decomposition method and existing working set selection are both discussed. A new method of working set selection is then presented in section III. In section IV, experiments with corresponding analysis are given. Finally, section V concludes this paper.

II. Literature Review

In this section we discuss SMO and existing working set selections.

A. Sequential Minimal Optimization

Sequential Minimal Optimization (SMO) was proposed by Platt [5], which is an extreme case of the decomposition algorithm where the size of working set is restricted to two. This method is named as *Algorithm 1*. Keerthi improved the performance of this algorithm for training SVMs (classifications and regressions [9], [7]). To take into account the situation that the Kernel matrices are Non-Positive Definite, Pai-Hsuen Chen *et al.* [10], [8] introduce the *Algorithm 2* by restrict proof.

Algorithm 2

Step 1: Find α^1 as the initial feasible solution. Set k=1.

Step 2: If α^k is a stationary point of (1), stop. Otherwise, find a working set $B \equiv \{i, j\}$

Step 3: Let $a_{ij} = K_{ii} + K_{jj} - K_{ij}$.If $a_{ij} > 0$, solve the sub-problem. Otherwise, solve:

$$
\begin{aligned}
\min_{B} \quad & \frac{1}{2}\begin{bmatrix}\alpha_i & \alpha_j\end{bmatrix}\begin{bmatrix}Q_{ii} & Q_{ij}\\ Q_{ji} & Q_{jj}\end{bmatrix}\begin{bmatrix}\alpha_i\\ \alpha_j\end{bmatrix} \\
& + (-e_B + Q_{BN}\alpha_N^k)^T\begin{bmatrix}\alpha_i\\ \alpha_j\end{bmatrix} \\
& + \frac{\tau - \alpha_{ij}}{4}((\alpha_i - \alpha_i^k)^2 + (\alpha_j - \alpha_j^k)^2) \qquad (2)\\
\text{subject to} \quad & 0 \le \alpha_i \le C \\
& y_i\alpha_i + y_j\alpha_j = -y_N^T\alpha_N^k
\end{aligned}
$$

where τ is a small positive number.

Step 4: Set $\alpha_N^{+1} \equiv \alpha_N^k$. Set $k \leftarrow k+1$ and goto step 2.

Working set selection is a very important procedure during training SVMs. We discuss some existing methods in next subsection.

B. Existing Working Set Selections

Currently a popular way to select the working set B is via the "maximal violating pair", we call it *WSS 1* for short. This working set selection was first proposed in [6], and is used in, for example, the software LIBSVM [3]. Instead of using first order approximation which *WSS 1* has used, a new method which consider the more accurate second order information be proposed [8], we call it *WSS 2*. By using the same i as in *WSS 1*, *WSS 2* check only $O(l)$ possible B's to decide j. Experiments indicate that a full check does not reduce iterations of using *WSS 2* much [10], [8].

But,for the linear kernel, sometimes K is only positive semi-definite, so it is possible that $K_{ii}+K_{jj}-2K_{ij}=0$. Moreover, some existing kernel functions (e.g., sigmoid kernel) are not the inner product of two vectors, so K is even not positive semi-definite. Then $K_{ii}+K_{jj}-2K_{ij}<0$ may occur .

For this reason, Chen *et al.* [8], propose a new working set selection named *WSS 3*.

WSS 3

Step 1: Select

$$
\overline{a_{ts}} \equiv \begin{cases} a_{ts} & \text{if} \quad a_{ts} > 0 \\ \tau & \text{otherwise} \end{cases}
$$

$$
i \in arg\max_t\{-y_t\nabla f(\alpha^k)_t | t \in I_{up}(\alpha^k)\}, \qquad (3)
$$

Step 2:

Consider Sub(B) defined and select

$$
\begin{aligned}
j \in arg\min_t\{ & -\frac{b_{it}^2}{\overline{a_{it}}} | t \in I_{low}(\alpha^k), \\
& -y_t\nabla f(\alpha^k)_t < -y_i\nabla f(\alpha^k)_i\} \qquad (4)
\end{aligned}
$$

Step 3: Return $B = \{i, j\}$
where

$$
\begin{aligned}
a_{ij} &\equiv K_{ii} + K_{jj} - 2K_{ij} > 0 \\
b_{ij} &\equiv -y_i\nabla f(\alpha^k)_i + y_j\nabla f(\alpha^k)_j > 0
\end{aligned}
$$

III. Our New Method

A. Interesting Phenomena

Interestingly, when we test the datasets by LIBSVM [3] which employs *Algorithm 2* and *WSS 3*, some phenomena attract us. Fig. 1 illustrates two of them.

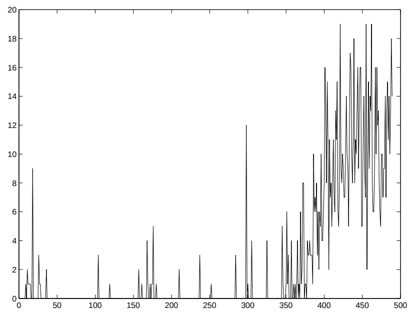

Fig. 1. The illustration of frequency of α reselection. Using dataset A1A, and kernel method RBF. Abscissa indicates the indices of α, and ordinate indicates the number of times a certain α is picked in the training process

The first phenomenon is, lots of α have not been selected at all in the training process. Because of the using of "Shrinking" [4], some samples will be "shrunk" during the training procedure, thus, they will never be selected and optimized. At the same time, we notice another interesting phenomenon that several α are selected to optimize the problem again and again, while others remain untouched. But does this kind of reselection necessary?

B. Working Set Selection Without Reselection(WSS-WR)

After investigating the phenomena above, if we limit the reselection of α, we could effectively reduce the time for training SVMs, with an acceptable effect on the ability of generalization. Thus, we propose our new method of working set selection where a certain α can only be selected once.

Before introducing the novel working set selection, we give some definitions:

Definition 1: $T^{k+1}, k \in \{1, \cdots, \frac{l}{2}\}$ is denoted as optimized set, in which $\forall\alpha \in T$ has been selected and optimized once in working set selection.

Definition 2: $C^{k+1} \subset \{1, \ldots, l\}\backslash T^k$ is called available set, in which $\forall\alpha \in C$ has never been selected.

For optimization problem (1), $\alpha \equiv C \cup T \cup B$.

Our method can be described as following: In iteration k, a set $B^k \equiv \{\alpha_i^k, \alpha_j^k\}$ will be selected from the new method which considers the more accurate second order information available set C^k by *WSS 3*, after optimization, both sample in this set will be put into T^{k+1}, that is to say, $T^{k+1} = T^k \cup B^k$, $C^{k+1} = C^k \setminus B^k$. In other words, once a working set is selected and optimized, all samples in it will not be examined anymore in the following selection. The relationship between sets B, C, T are shown in the Fig. 2

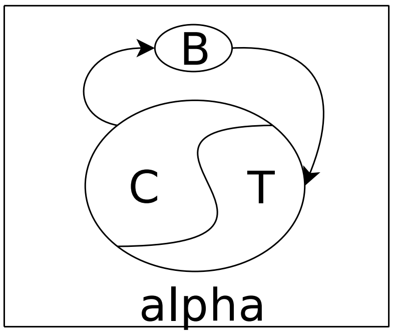

Fig. 2. The Model of Working Set Selection Without Reselection (*WSS-WR*)

1) Working Set Selection Without Reselection (WSS-WR):
Step 1: Select

$$\overline{a_{ts}} \equiv \begin{cases} a_{ts} & \text{if} \quad a_{ts} > 0 \\ \tau & \text{otherwise} \end{cases}$$

$$i \in arg\max_t \{ -y_t \nabla f(\alpha^k)_t | t \in I_{up}(\alpha^k) \cap t \notin T^k \}, \quad (5)$$

$$j \in arg\min_t \{ -\frac{b_{it}^2}{\overline{a_{it}}} | t \in I_{low}(\alpha^k) \cap t \notin T^k, \\ -y_t \nabla f(\alpha^k)_t < -y_i \nabla f(\alpha^k)_i \}. \quad (6)$$

Step 2: $T^{k+1} = T^k \cup B^k$, $C^{k+1} = C^k \setminus B^k$
Step 3: Return $B = \{i, j\}$
where

$$a_{ij} \equiv K_{ii} + K_{jj} - 2K_{ij} > 0$$
$$b_{ij} \equiv -y_i \nabla f(\alpha^k)_i + y_j \nabla f(\alpha^k)_j > 0$$

We name this new method as Working Set Selection Without Reselection (*WSS-WR*), in which all α will not be reselected.

C. Some Properties of WSS-WR Model

WSS-WR has some special properties. In this subsection, we simply prove some features of this new model.

Theorem 1: The values of all selected $\{\alpha_i, \alpha_j\} \subset B$ will always be 0.

Proof: Firstly, each α can only be selected once, which means once a α is chosen for optimization, the value of it has never been modified before; secondly, all α will be initialized as 0 at the beginning of the algorithm. Thus, the values of all selected $\{\alpha_i, \alpha_j\} \subset B$ will be 0. ■

Theorem 2: The algorithm terminates after a maximum of $\lceil \frac{l}{2} \rceil$ iterations.

Proof: Firstly, the algorithm terminates if there is no sample left in C^k or certain optimization conditions are reached; secondly, in each iteration, two samples will be selected and deleted from the available set. Thus, under the worst situation, after $\lceil \frac{l}{2} \rceil$ iterations, there will be no samples left in the active set, the algorithm then terminates. ■

Lemma 1: In the Model of *WSS-WR*, $I_{up} \equiv I_1$ and $I_{low} \equiv I_4$.

Proof: According to the Theorem 1, the $\alpha \equiv 0$ which is chosen by *WSS-WR*. And Keerthi [9] define the I_{up}, I_{low} as:

$$I_0 \equiv \{i : 0 < \alpha_i < C\}$$
$$I_1 \equiv \{i : y_i = +1, \alpha_i = 0\} \quad (7)$$
$$I_2 \equiv \{i : y_i = -1, \alpha_i = C\}$$
$$I_3 \equiv \{i : y_i = +1, \alpha_i = C\}$$
$$I_4 \equiv \{i : y_i = -1, \alpha_i = 0\} \quad (8)$$

and $I_{up} \equiv \{I_0 \cup I_1 \cup I_2\}, I_{low} \equiv \{I_0 \cup I_3 \cup I_4\}$
Thus $I_{up} \equiv I_1$ and $I_{low} \equiv I_4$ in *WSS-WR* model. ■

Lemma 2: In the Model of *WSS-WR*, $\alpha_1^{new} = \alpha_2^{new}$.

Proof: During the SMO algorithm searches through the feasible region of the dual problem and minimizes the objective function (1).

Because $\sum_{i=1}^{N} y_i \alpha_i = 0$, we have

$$y_1\alpha_1^{new} + y_2\alpha_2^{new} = y_1\alpha_1^{old} + y_2\alpha_2^{old}$$

Since

$$\alpha_1^{old} \in I_{up}, \alpha_2^{old} \in I_{low}$$

Therefore

$$y_1\alpha_1^{new} + y_2\alpha_2^{new} = 0$$
$$\Rightarrow y_1\alpha_1^{new} = -y_2\alpha_2^{new}$$

According to the *Lemma 1*

$$y_1 = -y_2$$
$$So\ \alpha_1^{new} = \alpha_2^{new}$$

■

IV. Computational Comparison

In this section we compare the performance of our model against *WSS 3*. The comparison between *WSS 1* and *WSS 3* have been done by Rong-En Fan *et al.* [8].

A. Data and Experimental Settings

Some small datasets (around 1,000 samples) including nine binary classification and two regression problems are investigated under various settings. And the large (more than 30,000 instances) classification problems are also taken into account. We select splice from the Delve archive (http://www.cs.toronto.edu/∼delve). Problems german.numer, heart and australian are from the Statlog collection [11]. Problems fourclass is from [12] and be transformed to a two-class set. The datasets diabetes, breast-cancer, and mpg are from the UCI machine learning repository [13]. The dataset mg from [14]. Problems a1a and a9a are compiled in Platt [5] from the UCI "adult" dataset. Problems w1a and w8a are also from Platt [5]. The problem IJCNN1 is from the first problem of IJCNN 2001 challenge [15].

For most datasets, each attribute is linearly scaled to $[-1, 1]$ except a1a, a9a, w1a, and w8a, because they take two values, 0 and 1. All data are available at http://www.csie.ntu.edu.tw/

Kernel	Problems Type	log_2^C	log_2^γ
RBF	Classification	-5,15,2	3,-15,-2
	Regression	-1,15,2	3,-15,-2
Linear	Classification	-3,5,2	
	Regression	-3,5,2	
Polynomial	Classification	-3,5,2	-5,-1,1
	Regression	-3,5,2	-5,-1,1
Sigmoid	Classification	-3,12,3	-12,3,3
	Regression	$\gamma = \frac{1}{\#Features}$	-8,-1,3

TABLE I

PARAMETERS USED FOR VARIOUS KERNELS: VALUES OF EACH PARAMETER ARE FROM A UNIFORM DISCRETIZATION OF AN INTERVAL. WE LIST THE LEFT, RIGHT END POINTS AND THE SPACE FOR DISCRETIZATION. FOR EXAMPLE:-5,15,2. FOR $\log 2^C$ MEANS $\log 2^C = -5, -3, \ldots, 15$

$\sim$cjlin/libsvmtools/. We set $\tau = 10^{-12}$ both in *WSS 3* and *WSS-WR*.

Because different SVMs parameters and kernel parameters affect the training time, it is difficult to evaluate the two methods under every parameter setting. For a fair comparison, we use the experimental procedure which Rong-En Fan *et al.* [8] used:

1. "Parameter selection" step: Conduct five-fold cross validation to find the best one within a given set of parameters.

2. "Final training" step: Train the whole set with the best parameter to obtain the final model.

Since we concern the performance of using different kernels, we thoroughly test four commonly used kernels:

1. RBF kernel:

$$K(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2}$$

2. Linear kernel:

$$K(x_i, x_j) = x_i^T x_j$$

3. Polynomial kernel:

$$K(x_i, x_j) = (\gamma(x_i^T x_j + 1))^d$$

4. Sigmoid kernel:

$$K(x_i, x_j) = tanh(\gamma x_i^T x_j + d)$$

Parameters used for each kernel are listed in Table I. It is important to check how *WSS-WR* performs after incorporating shrinking and caching strategies. We consider various settings:

1. With or without shrinking.

2. Different cache size: First a 100MB cache allows the whole kernel matrix to be stored in the computer memory. Second, we allocate only 100KB memory, so cache miss may happen and more kernel evaluations are needed. The second setting simulates the training of large-scale sets whose kernel matrices cannot be stored.

B. Numerical Experiments

1) Comparison of Cross Validation Accuracy of Classification: First, the grid method is applied. Cross validation accuracy is compared during "parameters selection" and "final training".

method	a1a	w1a	aust.	spli.	brea.
WSS-WR	83.4268	97.7796	86.2319	86	97.3646
WSS 3	83.8006	97.9814	86.8116	86.8	97.2182
method	diab.	four.	germ.	heart	
WSS-WR	77.9948	100	75.8	85.1852	
WSS 3	77.474	100	77.6	84.4444	

TABLE II

ACCURACY COMPARISON BETWEEN *WSS 3* AND *WSS-WR* (RBF)

method	a1a	w1a	aust.	spli.	brea.
WSS-WR	83.053	97.4162	85.5072	80.4	97.2182
WSS 3	83.8629	97.8199	85.5072	79.5	97.2182
method	diab.	four.	germ.	heart	
WSS-WR	76.4323	77.8422	75.3	83.7037	
WSS 3	77.2135	77.6102	77.9	84.4444	

TABLE III

ACCURACY COMPARISON BETWEEN *WSS 3* AND *WSS-WR* (LINEAR)

We test various situations concerning all the commonly used kernels, with and without shrinking technique and 100MB/100KB cache. The Table II- V show that cross validation accuracy between *WSS-WR* and *WSS 3* method are almost the same. To be specifically, $|Accuracy_{\textit{WSS-WR}} - Accuracy_{\textit{WSS 3}}| < 0.026$. And in most of the datasets, the accuracy of *WSS 3* outperform that of *WSS-WR*. There are also several datasets in which *WSS-WR* performs even more accurate than *WSS 3*. Besides, great improvement are made both in the number of iterations as well as the consumption of time.

2) Comparison of Cross Validation Mean Squared Error of Regression: The Table VI shows that cross validation mean squared error does not differ much in regression between *WSS-WR* and *WSS 3*.

3) Iteration and time ratios between WSS-WR and WSS 3: After comparison of cross validation accuracy of classification and cross validation mean squared error of regression, We

method	a1a	w1a	aust.	spli.	brea.
WSS-WR	83.3645	97.8603	86.087	82.1	97.6574
WSS 3	83.3645	97.6181	85.7971	82.7	97.511
method	diab.	four.	germ.	heart	
WSS-WR	76.6927	78.6543	75.1	84.0741	
WSS 3	77.0833	79.6984	75.2	84.8148	

TABLE IV

ACCURACY COMPARISON BETWEEN *WSS 3* AND *WSS-WR* (POLYNOMIAL)

method	a1a	w1a	aust.	spli.	brea.
WSS-WR	83.6137	97.6988	85.7971	80.4	97.0717
WSS 3	83.4268	97.8603	85.6522	80.5	97.2182
method	diab.	four.	germ.	heart	
WSS-WR	77.2135	77.8422	75.8	84.8148	
WSS 3	77.2135	77.8422	77.8	84.0744	

TABLE V

ACCURACY COMPARISON BETWEEN *WSS 3* AND *WSS-WR* (SIGMOID)

Kernel Methods	RBF		Linear	
Method	*WSS-WR*	*WSS 3*	*WSS-WR*	*WSS 3*
MPG	6.9927	6.4602	12.325	12.058
MG	0.01533	0.014618	0.02161	0.02138
Kernel Methods	Polynomial		Sigmoid	
Method	*WSS-WR*	*WSS 3*	*WSS-WR*	*WSS 3*
MPG	0.25	0.5	14.669	14.22
MG	0.019672	0.018778	0.023228	0.023548

TABLE VI
CROSS VALIDATION MEAN SQUARED ERROR

illustrate the iteration and time ratios between *WSS-WR* and *WSS 3*.

For each kernel, we give two figures showing results of "parameter selection" and "final training" steps, respectively. We further separate each figure to two situations: without/with shrinking, and present five ratios between using *WSS-WR* and using *WSS 3*:

$$ratio1 \equiv \frac{\text{time of } \textit{WSS-WR} \text{ (100M cache,shrinking)}}{\text{time of } \textit{WSS 3} \text{ (100M cache,shrinking)}}$$

$$ratio2 \equiv \frac{\text{time of } \textit{WSS-WR} \text{ (100M cache,nonshrinking)}}{\text{time of } \textit{WSS 3} \text{ (100M cache,nonshrinking)}}$$

$$ratio3 \equiv \frac{\text{time of } \textit{WSS-WR} \text{ (100K cache,shrinking)}}{\text{time of } \textit{WSS 3} \text{ (100K cache,shrinking)}}$$

$$ratio4 \equiv \frac{\text{time of } \textit{WSS-WR} \text{ (100K cache,nonshrinking)}}{\text{time of } \textit{WSS 3} \text{ (100K cache,nonshrinking)}}$$

$$ratio5 \equiv \frac{\text{Total iteration of } \textit{WSS-WR}}{\text{Total iteration of } \textit{WSS 3}}$$

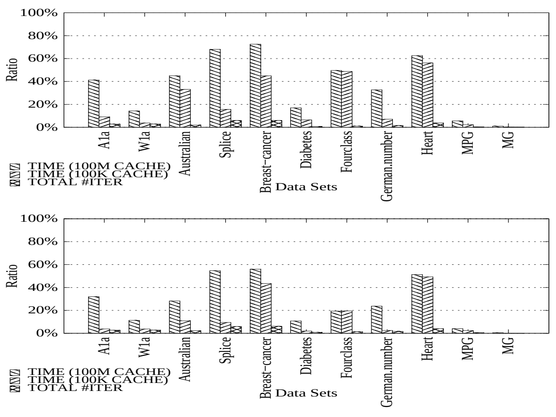

Fig. 3. Iteration and time ratios between *WSS-WR* and *WSS 3* using the RBF kernel for the "parameter selection" step (top: with shrinking, bottom: without shrinking).

The number of iterations is independent of the cache size. In the "parameter selection" step, time (or iterations) of all parameters is summed up before calculating the ratio. In general the "final training" step is very fast, so the timing result may not be accurate. Hence we repeat this step 4 times to obtain more reliable timing values. Fig. 3- 10 present obtain ratios, They are in general less than 1. We can conclude that using *WSS-WR* is in general better than using *WSS 3*.

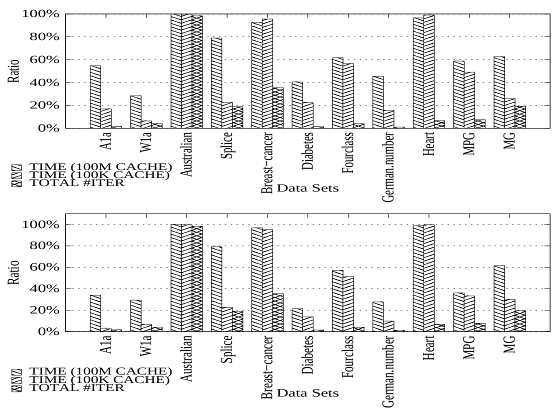

Fig. 4. Iteration and time ratios between *WSS-WR* and *WSS 3* using the RBF kernel for the "final training" step (top: with shrinking, bottom: without shrinking).

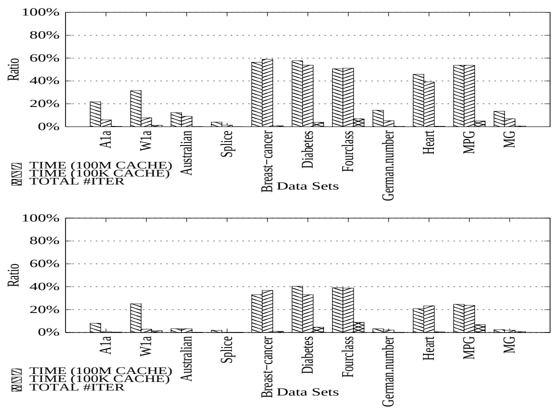

Fig. 5. Iteration and time ratios between *WSS-WR* and *WSS 3* using the Linear kernel for the "parameter selection" step (top: with shrinking, bottom: without shrinking).

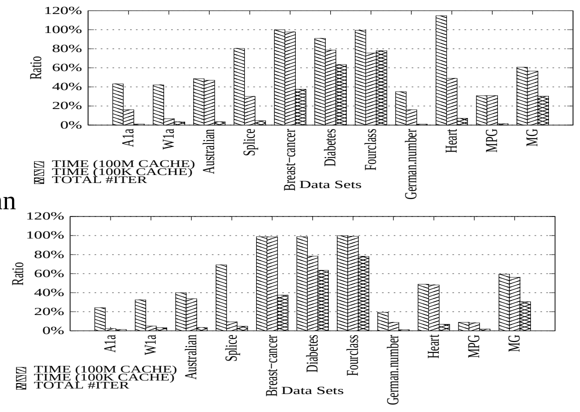

Fig. 6. Iteration and time ratios between *WSS-WR* and *WSS 3* using the Linear kernel for the "final training" step (top: with shrinking, bottom: without shrinking).

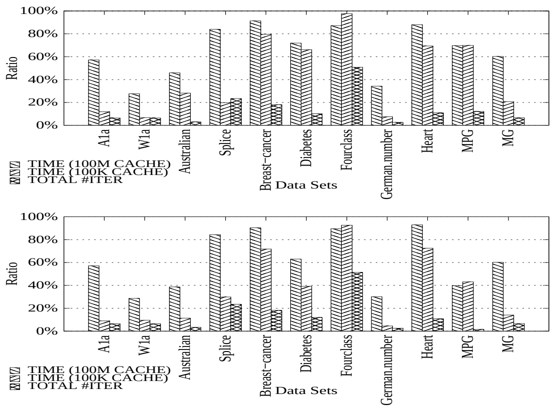

Fig. 7. Iteration and time ratios between *WSS-WR* and *WSS 3* using the Polynomial kernel for the "parameter selection" step (top: with shrinking, bottom: without shrinking).

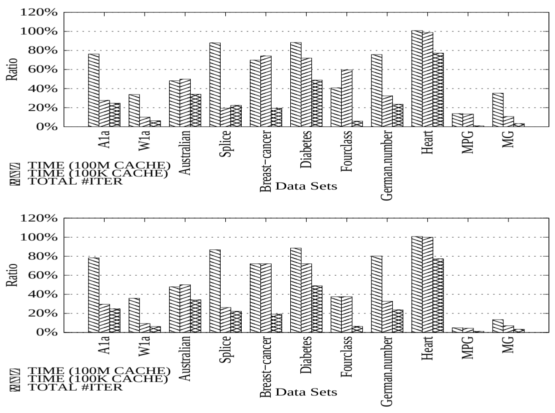

Fig. 8. Iteration and time ratios between *WSS-WR* and *WSS 3* using the Polynomial kernel for the "final training" step (top: with shrinking, bottom: without shrinking).

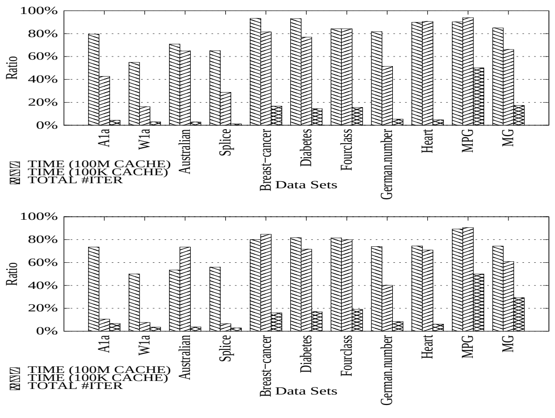

Fig. 9. Iteration and time ratios between *WSS-WR* and *WSS 3* using the Sigmoid kernel for the "parameter selection" step (top: with shrinking, bottom: without shrinking).

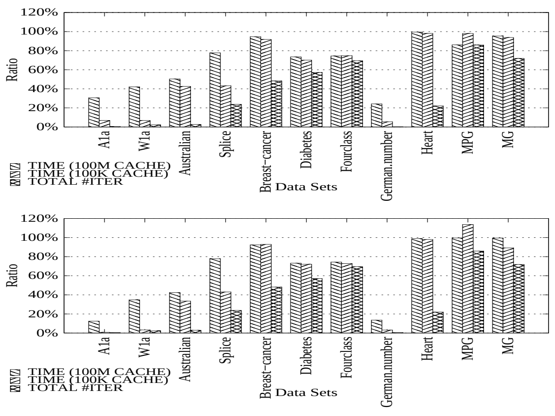

Fig. 10. Iteration and time ratios between *WSS-WR* and *WSS 3* using the Sigmoid kernel for the "final training" step (top: with shrinking, bottom: without shrinking).

WSS-WR				
method	RBF	Linear	Poly.	Sigm.
100M, shrinking	56.8936	15.6038	9.3744	15.9825
100M, nonshrinking	56.8936	15.6038	9.7964	16.1321
100K, shrinking	59.9929	17.7689	9.4823	16.7169
100K, nonshrinking	59.8346	15.3692	8.9024	16.8676
WSS 3				
	RBF	Linear	Poly.	Sigm.
100M, shrinking	400.7368	49.5343	34.0337	29.0977
100M, nonshrinking	505.7596	62.5636	34.3321	32.2544
100K, shrinking	1655.7840	232.5998	140.9247	103.3014
100K, nonshrinking	1602.4966	533.9386	93.3340	220.2574

TABLE VII

W1A, COMPARISON OF CACHING AND SHRINKING

4) Caching and Shrinking techniques in WSS-WR: According to the analysis of the experimental results, in *WSS-WR* model, the caching and shrinking techniques do not have much more effects on SMO decomposition method. The data in Table VII, VIII certify the above conclusion.

5) Experiments of Large Classification Datasets: Next, the experiment with large classification sets is handled by a similar procedure. As the parameter selection is time consuming, we adjust the "parameter selection" procedure to a 16-point searching. The cache size are 300MB and 1MB. The experiments employ RBF and Sigmoid kernel methods, along with that Sigmoid kernel in general leads the worst ratio between

WSS-WR				
method	RBF	Linear	Poly.	Sigm.
100M, shrinking	97.8465	24.0803	15.9311	31.7092
100M, nonshrinking	97.8465	24.0803	15.9311	31.7092
100K, shrinking	99.8329	26.4382	15.0456	30.8630
100K, nonshrinking	92.9560	25.7199	15.0456	28.5457
WSS 3				
method	RBF	Linear	Poly.	Sigm.
100M, shrinking	299.3279	168.0513	46.5298	38.7958
100M, nonshrinking	413.3258	764.8448	52.9702	42.9117
100K, shrinking	1427.4005	505.8935	201.4057	60.0467
100K, nonshrinking	3811.7786	1213.8522	336.8386	71.2662

TABLE VIII

GERMAN.NUMER, COMPARISON OF CACHING AND SHRINKING

300MB cache			RBF kernel			
Datasets			Shrinking		No-Shrinking	
Problem	#data	#feat.	Iter.	Time	Iter.	Time
a9a	32,561	123	0.0522	0.2306	0.3439	0.8498
w8a	49,749	300	0.0370	0.0327	0.0149	0.2106
IJCNN1	49,990	22	0.1187	0.5889	0.3474	0.7422
300MB cache			Sigmoid kernel			
Datasets			Shrinking		No-Shrinking	
Problem	#data	#feat.	Iter.	Time	Iter.	Time
a9a	32,561	123	0.0522	0.2282	0.3439	0.8603
w8a	49,749	300	0.0380	0.0443	0.0339	0.4554
IJCNN1	49,990	22	0.1309	0.5928	0.3883	0.8147
1MB cache Nonshrinking			RBF		Sigmoid	
Problem	#data	#feat.	Iter.	Time	Iter.	Time
a9a	32,561	123	0.0522	0.0687	0.3439	0.3799
w8a	49,749	300	0.0370	0.0365	0.0149	0.3279
IJCNN1	49,990	22	0.1187	0.1673	0.3474	0.4093

TABLE IX

LARGE PROBLEMS: ITERATION AND TIME RATIOS BETWEEN *WSS-WR* AND *WSS 3* FOR 16-POINT "PARAMETER SELECTION".

WSS-WR and *WSS 3*. Table IX gives iteration and time ratios. Comparing the results among small problems, we safely draw the conclusion that ratios of time and iteration in large datasets are less than those of them in small ones, especially, in the situation of small size of cache.

C. Convergence Graph of WSS-WR

For the reason that *WSS-WR* employs *Algorithm 2* and second order information method, which *WSS 3* used, we just compare the convergence rates between them.

We set $C = 1, \gamma = 0, \epsilon = 10^{-3}$ during the whole comparison.

First, the evaluation is made on the following datasets: a1a, w1a, australian, splice, breast-cancer, diabetes, fourclass, german.numer and heart.

The illustrations in Fig. 11 are the first step evaluation, we compare the convergence ratios on nine datasets between *WSS-WR* and *WSS 3* with RBF kernel. Other charts are omitted here for short.

For further analysis, we choose two datasets: w1a and breast-cancer to made evaluations by using diversity kernel methods (Linear, RBF, Polynomial, Sigmoid). Fig. 12 compare the convergence on W1a between *WSS-WR* and *WSS 3*. The illustrations of breast-cancer are omitted here for short.

From the Fig. 11, 12. The convergence rates of *WSS-WR* and *WSS 3* are exactly the same at the beginning of the procedure. In addition, *WSS-WR* will be terminated soon when it reaches the optimum, while *WSS 3*, on the contrary, will hold the objective value or make insignificant progress with much more time consumed. Thus, it is reasonable to conclude that *WSS-WR* is much more efficient.

D. Discussion

With the above analysis, our main observations and conclusions from Fig. 3-10 and Table VII, VIII, IX are in the following:

1) The improvement of reducing the number of iterations is significant, which can be obtained from the illustrations.

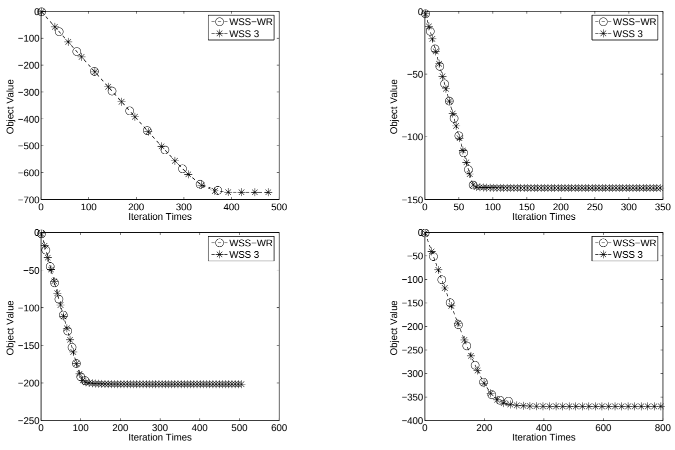

Fig. 11. The comparison of convergence on several datasets between *WSS-WR* and *WSS 3* with RBF kernel. (Datasets in order are: a1a, w1a, australian, splice)

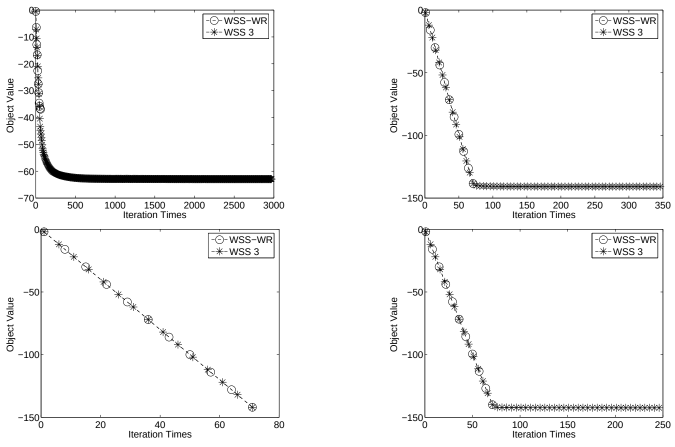

Fig. 12. The comparison of convergence on W1a between *WSS-WR* and *WSS 3*, using diversity kernels: RBF, Linear, Polynomial, Sigmoid

2) Using *WSS-WR* dramatically reduces the cost of time. The reduction is more dramatic for the parameter selection step, where some points have low convergence rates.
3) *WSS-WR* outperforms *WSS 3* in most datasets, both in the "parameter selection" step and in "final training" steps. Unlike *WSS 3*, the training time of *WSS-WR* does not increase when the amount of memory for caching drops. This property indicates that *WSS-WR* is useful under such situation where the datasets is too large for the kernel matrices to be stored, or where there is not enough memory.
4) The shrinking technique of LIBSVM [3] was introduced by Pai-Hsuen Chen *et al.*, [10] to make the decomposition method faster. But in the view of Table VII, VIII, experiments of regression and large classification problems, shrinking technique almost does not shorten the training time by using *WSS-WR*.
5) Fig. 3-10 indicate that the relationship of the five ratios

can be described as follows:

$$ratio5 < ratio4 < ratio3 < ratio2 < ratio1$$

Though this may not be valid in all datasets.

V. Conclusions

By analyzing the available working set selection methods and some interesting phenomena, we have proposed a new working set selection model–Working Set Selection Without Reselection (*WSS-WR*). Subsequently, full-scale experiments were given to demonstrate that *WSS-WR* outperforms *WSS 3* in almost datasets during the "parameters selection" and "final training" step. Then, we discussed some features of our new model, by analyzing the results of experiments. A theoretical study on the convergence of *WSS-WR* and to continually improve this model are our future work.

VI. Acknowledgements

This work is under the support of the National High Technology Research and Development Program of China (863 Program) under Grant NO.2006AA01Z232. Thanks to Chih-Chung Chang and Chih-Jen Lin for their powerful software, LIBSVM.

References

[1] B. Boser, I. Guyon, and V. Vapnik, “A training algorithm for optimal margin classifiers,” *In Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 1992.

[2] C. Cortes and V. Vapnik, “Support-vector network,” *Machine Learning*, 20 1995.

[3] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[4] T. Joachims, “Making large-scale support vector machine learning practical,” in *Advances in Kernel Methods: Support Vector Machines*, A. S. B. Schölkopf, C. Burges, Ed. MIT Press, Cambridge, MA, 1998.

[5] J. C. Platt, “Fast training of support vector machines using sequential minimal optimization,” in *Advances in Kernel Methods - Support Vector Learning*, A. J. S. Bernhard Schölkopf, Christopher J. C. Burges, Ed. MIT Press, Cambridge, MA, 1998.

[6] E. Osuna, R. Freund, and F. Girosi, “Training support vector machines: An application to face detection,” *In Proceedings of CVPR’97*, 1997.

[7] J.C.Platt, “Using sparseness and analytic qp to speed training of support vector machines,” in *Advances in Neural Information Processing Systems 11*, S. S. M.S.Kearns and d. D.A.Cohn, Eds. MIT Press, Cambridge, MA, 1999.

[8] R.-E. Fan, P.-H. Chen, and C.-J. Lin, “Working set selection using second order information for training support vector machines,” *Journal of Machine Learning Research*, 6 2005.

[9] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, “Improvements to platt’s smo algorithm for svm classifier design,” *Neural Computation*, 13 2001.

[10] P.-H. Chen, R.-E. Fan, and C.-J. Lin, “A study on smo-type decomposition methods for support vector machines,” *IEEE Transactions on Neural Networks*, 2006, to appear. http://www.csie.ntu.edu.tw/~cjlin/papers/generalSMO.pdf.

[11] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, “Machine learning, neural and statistical classification,” *Prentice Hal*, 1994, data available at http://www.ncc.up.pt/liacc/ML/statlog/datasets.html.

[12] T. K. Ho and E. M. Kleinberg, “Building projectable classifiers of arbitrary complexity,” *In Proceedings of the 13th International Conference on Pattern Recognition*, August 1996.

[13] C. L. Blake and C. J. Merz, “Uci repository of machine learning databases,” Tech. Rep., 1998, available at http://www.ics.uci.edu/~mlearn/MLRepository.html.

[14] G. W. Flake and S. Lawrence, “Efficient svm regression training with smo,” *Machine Learning*, 46 2002.

[15] D. Prokhorov, “Ijcnn 2001 neural network competition,” Tech. Rep., 2001, available. http://www.geocities.com/ijcnn/nnc_ijcnn01.pdf.