# Multi-Order Information for Working Set Selection of Sequential Minimal Optimization

**Qimao Yang**
East China Normal University

**Changrong Li**
CFETS Information Technology

**Jun Guo**
East China Normal University

## Abstract

A new working set selection method for sequential minimal optimization (SMO) is proposed in this paper. Instead of the method adopted in the current version of LIBSVM, which uses the second order information of the objective function to choose the violating pairs, we suggest a new method where a higher order information is considered. It includes the descent degree of the objective function and the stride of variables update. Many experimental results show, in contrast to LIBSVM, the number of iterations obtained by the proposed method is less in the vast majority of cases and the training of support vector machines (SVMs) is sped up. Meanwhile, the convergence of the proposed approach can be guaranteed and its accuracy is at the same level as LIBSVM's.

## 1 Introduction

Support vector machine (SVM) is one of the most popular machine learning algorithms, which has been widely used in many areas including image classification [1], disease detection [2, 3], malware classification [4], and so on. Furthermore, the models that are composed of deep learning and SVM [5–9] often show more significant performance than the deep learning models where softmax layer is connected in the last layer. However, despite the advantages of SVM, it takes much more time to solve large and complex problems.

There is a lot of work proposed to improve the performance of SVMs [10–13]. We have also proposed parallel sequential minimal optimization (SMO) algorithms in the previous work [14, 15], which improve the efficiency by updating several violating pairs synchronously at each iteration. Non-convergence can be avoided by calculating the impact of pre-selection on the remaining working set.

We focus on studying a better approach of working set selection to shorten the time in training phase in this paper. One of the classic working set selection algorithms is called maximal violating pair (MVP) method, which is to choose the violating pairs according to the degree of violation of the Karush-Kuhn-Tucker (KKT) condition [16]. Fan *et al.* [17] proposed an improved method where the reduction of the objective function is considered and second order information is used. Their experimental results showed that it can reduce the number of iterations to achieve faster convergence and better performance. As a famous library for SVMs, LIBSVM also adopts second-order information approach since version 2.8.

In this paper, we propose a new working set selection method, which incorporates the merits of previous algorithms. To optimize the object function, we focus on not only the reduction of the objective value but also more information. Besides, we introduce a relaxation factor $\lambda$ to limit the choice of working set. The factor guarantees that the selected working sets can satisfy the violation of the KKT condition rigorously. From the results of our experiments, we can see whether in the number of iterations or the training time, the proposed algorithm is better than LIBSVM with second order information algorithm.

This paper is organized as follows. Section 2 introduces SVM and SMO algorithms. Several typical working set selection methods are mentioned in Section 3. The proposed new working set selection algorithm is introduced in Section 4, and the specific experiments can be found in Section 5. Finally, Section 6 gives a conclusion.

## 2 Sequential Minimal Optimization

SVM was proposed by Vapnik and Corinna Cortes [18, 19]. Given a training set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{l}$, where $\boldsymbol{x}_i$ is an

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. 

instance of the input vector $\boldsymbol{x}$ and $y_i \in \{-1, 1\}$ is the label of $\boldsymbol{x}_i$, $l$ is the number of training samples. A general form of quadratic programming (QP) problem in SVM is to minimize

$$W(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j Q_{i,j} - \sum_{i=1}^{l}\alpha_i, \qquad (1)$$

subject to the constraints

$$\sum_{i=1}^{l}\alpha_i y_i = 0,$$
$$0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, l,$$

where $Q_{i,j} = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the kernal function and $C$ is the penalty parameter. A decision function can be deduced as follows:

$$f(\boldsymbol{x}) = \sum_{i=1}^{l}\alpha_i^* y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b^*,$$

where $\boldsymbol{\alpha}^*$ is the optimal solution of Eq. (1), and $b^*$ is the optimum value of the bias $b$ that could be calculated by $\boldsymbol{\alpha}^*$. In other words, after getting the value of $\boldsymbol{\alpha}^*$, we could get a perfect decision function.

Several algorithms[20, 21] have been proposed to solve this optimization problem. However, the number of variables in Eq. (1) is equal to the number of training samples, and these algorithms are inefficient when the number of training samples is large. SMO[22] was proposed by Platt in 1998 accordingly, which only contains two variables as its working set. Assume these two variables to be $\alpha_1$, $\alpha_2$, and other variables are fixed, the subproblem is to minimize

$$D(\alpha_1, \alpha_2) = \frac{1}{2}K_{11}\alpha_1^2 + \frac{1}{2}K_{22}\alpha_2^2 + y_1 y_2 K_{12}\alpha_1\alpha_2 - (\alpha_1 + \alpha_2) + y_1\alpha_1 A_1 + y_2\alpha_2 A_2 + \delta \quad (2)$$

subject to the constraints

$$\alpha_1 y_1 + \alpha_2 y_2 = -\sum_{i=3}^{l}\alpha_i y_i = \rho, \qquad (3)$$
$$0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, l,$$

where $K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $A_i = \sum_{j=3}^{l} y_j\alpha_j K_{ij}, i = 1, 2$, and $\delta$,$\rho$ are constants. $\delta$ is a constant term omitted in above subproblem. Therefore, there is only one free variable in Eq. (2) through Eq. (3).

The remaining question is how to choose a violating pair to be a working set. According to KKT condition, the optimality condition can be expressed as:

$$\min_{i \in I_{\text{up}}(\boldsymbol{\alpha})} F_i(\boldsymbol{\alpha}) \geq \max_{i \in I_{\text{low}}(\boldsymbol{\alpha})} F_i(\boldsymbol{\alpha})$$

where

$$F_i(\boldsymbol{\alpha}) = y_i\frac{\partial W(\boldsymbol{\alpha})}{\partial \alpha_i} = y_i(\sum_{j=1}^{l} y_i y_j \alpha_j K_{ij} - 1),$$

and

$$I_{\text{up}}(\boldsymbol{\alpha}) = \{i : \alpha_i \leq C, y_i = 1\} \cup \{i : \alpha_i \geq 0, y_i = -1\},$$
$$I_{\text{low}}(\boldsymbol{\alpha}) = \{i : \alpha_i \leq C, y_i = -1\} \cup \{i : \alpha_i \geq 0, y_i = 1\}.$$

A pair of indices $(i, j)$ is said to be a violating pair if the following condition hold:

$$i \in I_{\text{up}}(\boldsymbol{\alpha}), j \in I_{\text{low}}(\boldsymbol{\alpha}), -F_i(\boldsymbol{\alpha}) \geq -F_j(\boldsymbol{\alpha}) + \lambda,$$

where $\lambda$ is a relaxation factor to generalize the model. Hash has proved the following theorem [23]:

**Theorem 1** *$Q$ is assumed to be a positive semi-definite matrix. If and only if the working set $B$ is a violating pair, SMO can ensure the objective function Eq. (1) to be a strictly monotone decreasing function, ie. $\forall k$, $Q(\boldsymbol{\alpha})^{k+1} \leq Q(\boldsymbol{\alpha})^k$.*

## 3 Traditional Working Set Selection

In process of SMO, only one violating pair $(i, j)$ is chosen to be a working set $(\alpha_i, \alpha_j)$ in each iteration. After updating this violating pair, another is been chosen and repeat the above steps. Therefore, studying approaches of working set selection is necessary.

Maximal violating pair is a classic working set selection algorithm, which was proposed by Keerthi *et al.* in 2001 and was used in the earlier LIBSVM tools [16]. MVP selects a pair $(i, j)$ according to the following conditions:

$$i \in \arg\max_t\{-F_t(\boldsymbol{\alpha}) | t \in I_{\text{up}}(\boldsymbol{\alpha})\},$$
$$j \in \arg\min_t\{-F_t(\boldsymbol{\alpha}) | t \in I_{\text{low}}(\boldsymbol{\alpha})\}.$$

MVP algorithm chooses violating pairs through the degree of violation of the KKT condition, which is related to first order approximation of $W(\boldsymbol{\alpha})$.

Working set selection algorithm using second order information, which was has been applied in the LIBSVM tools [17, 23, 24] since version 2.8 has emerged, proposed by Fan *et al.* in 2005. Assume $\boldsymbol{d}^T = [\boldsymbol{d}_B^T, \boldsymbol{0}_N^T]$ is the update vector of Lagrange multipliers and $B$ is a working set, and consider the reduction of the objec-

tive value:

$$\begin{aligned}
&W(\boldsymbol{\alpha}^k+\boldsymbol{d})-W(\boldsymbol{\alpha}^k)\\
=&\nabla W(\boldsymbol{\alpha}^k)^T\boldsymbol{d}+\frac{1}{2}\boldsymbol{d}^T\nabla^2W(\boldsymbol{\alpha}^k)\boldsymbol{d}\\
=&\nabla W(\boldsymbol{\alpha}^k)_B^T\boldsymbol{d}_B+\frac{1}{2}\boldsymbol{d}_B^T\nabla^2W(\boldsymbol{\alpha}^k)_{BB}\boldsymbol{d}_B\\
=&\begin{bmatrix}\nabla W(\alpha_i^k)\,\nabla W(\alpha_j^k)\end{bmatrix}\begin{bmatrix}d_i\\d_j\end{bmatrix}+\frac{1}{2}\begin{bmatrix}d_i\;d_j\end{bmatrix}\begin{bmatrix}Q_{ii}\,Q_{ij}\\Q_{ji}\,Q_{jj}\end{bmatrix}\begin{bmatrix}d_i\\d_j\end{bmatrix}\\
=&(-F_i(\boldsymbol{\alpha}^k)+F_j(\boldsymbol{\alpha}^k))d_j^{'}+\frac{1}{2}(K_{ii}+K_{jj}-2K_{ij})d_j^{'2}\quad(4)
\end{aligned}$$

where $k$ means the $k-$th iteration and $d_i^{'}=y_id_i$. According to Eq. (3), we can conclude $d_i^{'}+d_j^{'}=0$. Obviously, the minimum of Eq. (4) is $-b_{ij}^2/2a_{ij}$, which is taken at $d_j^{'}=-b_{ij}/a_{ij}<0$,

$$a_{ij}=K_{ii}+K_{jj}-2K_{ij}>0 \tag{5}$$

and

$$b_{ij}=-F_i(\boldsymbol{\alpha}^k)+F_j(\boldsymbol{\alpha}^k)>0. \tag{6}$$

In summary, the second order information algorithm is as follows:

**Algorithm 1** *Working set selection using second order information.*
*Step 1. Select* $i\in\arg\max_t\{-F_t(\boldsymbol{\alpha})\,|\,t\in I_{up}(\boldsymbol{\alpha})\}$,
*Step 2. Select*
$j\in\arg\min_t\{-\frac{b_{it}^2}{a_{it}}\,|\,t\in I_{low}(\boldsymbol{\alpha}),-F_i(\boldsymbol{\alpha})\geq-F_t(\boldsymbol{\alpha})\}$.
*Step 3. Return* $B=i,j$.

In the previous two algorithms, compared to MVP, the second order information algorithm has fewer iterations and spends less time according to experimental results. The possible explanation may be that MVP uses first order information and is equivalent to the method of steepest gradient descent. The second order contains more information than the first order so it can converge faster.
However, second order information algorithm is similar to the greedy algorithm. It only guarantees that the current pair can reduce the object value maximally in the current iteration, but in the next iteration the reduction of the object value may be extremely little.

## 4 A New Working Set Selection Algorithm

In this section, we propose a new working set selection algorithm. The solution of the greedy algorithm in each iteration is the locally optimal solution, so is second order information algorithm. To solve this problem, we construct a new optimization function which is not only focus on the objective function. Thus, the violating pairs obtained by this method could avoid locally optimal solutions. We name this algorithm multi-order information algorithm.
First of all, we hope this method can also focus on the update step size of the multiplier in some extent. In order to tie the update step size of the multiplier $d_j^{'}$ with the decrease of the objective value $(W(\boldsymbol{\alpha}^k+\boldsymbol{d})-W(\boldsymbol{\alpha}^k))$, we construct the optimization function as follows:

$$\min H=(W(\boldsymbol{\alpha}^k+\boldsymbol{d})-W(\boldsymbol{\alpha}^k))\left|d_j^{'}\right|. \tag{7}$$

By multiplying the update step size of the multiplier and the decrease of the objective value, SMO avoids focusing only on minimizing Eq. (4) when selecting a working set. For removing the calculation of the absolute value in the above expression and simplifying the calculation, we construct an equivalent function $G$ as follows:

$$\max G=H^2=(W(\boldsymbol{\alpha}^k+\boldsymbol{d})-W(\boldsymbol{\alpha}^k))^2(d_j^{'})^2. \tag{8}$$

We can substitute $(W(\boldsymbol{\alpha}^k+\boldsymbol{d})-W(\boldsymbol{\alpha}^k))$ in Eq. (8) with Eq. (4):

$$\max G=(b_{ij}d_j^{'}+\frac{1}{2}a_{ij}d_j^{'2})^2(d_j^{'})^2. \tag{9}$$

Take partial derivative of Eq. (9) with respect to $d_j^{'}$:

$$\begin{aligned}
\frac{\partial G}{\partial d_j^{'}}=&2(b_{ij}d_j^{'}+\frac{1}{2}a_{ij}d_j^{'2})(b_{ij}+a_{ij}d_j^{'})(d_j^{'})^2\\
&+2(b_{ij}d_j^{'}+\frac{1}{2}a_{ij}d_j^{'2})^2(d_j^{'})\\
=&(d_j^{'})^3(\frac{3}{2}a_{ij}^2d_j^{'2}+5a_{ij}b_{ij}d_j^{'}+4b_{ij}^2).
\end{aligned}$$

The optimal solutions of $d_j^{'}$ are as follow:

$$\begin{cases}
d_{j1}^{'} &= -2b_{ij}/a_{ij},\\
d_{j2}^{'} &= -4b_{ij}/3a_{ij},\\
d_{j3}^{'} &= 0.
\end{cases}$$

Eq. (9) get the maximum, at

$$d_j^{'}=-\frac{4b_{ij}}{3a_{ij}}.$$

At this time, Eq. (7) gets the minimum value:

$$H=-\frac{16b_{ij}^3}{27a_{ij}^2}.$$

Therefore, $d_j^{'}$, the update step size of the multiplier, is equal to $-4b_{ij}/3a_{ij}$. And the decrease of the objective value can be deduced from the formula below:

$$\begin{aligned} & W(\boldsymbol{\alpha}^k + \boldsymbol{d}) - W(\boldsymbol{\alpha}^k) \\ =& (-F_i(\boldsymbol{\alpha}^k) + F_j(\boldsymbol{\alpha}^k))d_j^{'} + \frac{1}{2}(K_{ii} + K_{jj} - 2K_{ij})d_j^{'2} \\ =& b_{ij}(-\frac{4b_{ij}}{3a_{ij}}) + \frac{1}{2}a_{ij}(-\frac{4b_{ij}}{3a_{ij}})^2 \\ =& -\frac{4b_{ij}^2}{9a_{ij}}. \end{aligned}$$

Some differences between multi-order information algorithm and second order information algorithm can be discovered from the outcome of these formulas. From the perspective of the update step size of the multiplier, the former one descends by $4b_{ij}/3a_{ij}$ while the latter one declines by $b_{ij}/a_{ij}$. From the aspect of the decrease of the objective value, the former one descends by $4b_{ij}^2/9a_{ij}$ while the latter one declines by $b_{ij}^2/2a_{ij}$. Although these two results are estimated values, they still give us important directions. The consequences hints that multi-order information algorithm sacrifices some of the reduced value of the objective function in exchange for a bigger update step size of the multiplier. In other words, our algorithm avoids focusing only on the objective function.

Moreover, in order to take into account the degree of violation of the KKT condition of the multiplier, we add relaxation factor $\lambda$ to our algorithm as a filter for $j$. Thus, the selection range of $j$ is limited for the same $i$.

Thus the new working set selection select is shown as follows:

$$i \in \arg\max_t\{-F_t(\boldsymbol{\alpha}) \,|\, t \in I_{\text{up}}(\boldsymbol{\alpha})\},$$

$$j \in \arg\min_t\{-\frac{b_{it}^3}{a_{it}^2} \,|\, t \in I_{\text{low}}(\boldsymbol{\alpha}), -F_i(\boldsymbol{\alpha}) \geq -F_t(\boldsymbol{\alpha}) + \lambda\}.$$

In a word, after the relaxation factor is added to the constraint of $j$, $B(= i, j)$ will not be selected if it does not reach the required degree of violation of the KKT condition.

Another problem needs to be solved is that if a Gaussian kernel is regarded as a kernel function, $a_{ij}$ will always greater than 0. In contrast, if the kernel function is a linear kernel or other types of kernel functions, $a_{ij}$ may be equal to 0 or even less than 0. We solve the problem by the method provided by Chen *et al.*, adding an additional term $\tau$ to Eq. (2). Then, the optimal objective value is $-b_{ij}^3/\tau^2$.

Therefore, the description of a generalized SMO-type decomposition algorithm using multi-order information for the working set selection can be expressed as follows:

**Algorithm 2** *An SMO-type decomposition algorithm using new working set selection method.*
*Given a data set* $T = \{\boldsymbol{x_i}, y_i\}_{i=1}^l$, *a penalty parameter* $C$ *and a hyperparameter* $g$ *in RBF.* $\boldsymbol{x_i}$ *is the eigenvector of the sample, and* $y_i \in \{-1, +1\}$ *is the label.*
*Step 1. Find* $\boldsymbol{\alpha}^0 = \mathbf{0}$ *as the initial feasible solution. Set* $k = 0$.
*Step 2. If* $a_{it} \leq 0$, *set* $a_{it} = \tau$ *( where* $\tau$ *is a small positive number ).*
*Step 3. Select*
$i \in \arg\max_t\{-F_t(\boldsymbol{\alpha}) \,|\, t \in I_{up}(\boldsymbol{\alpha})\}$,
$j \in \arg\min_t\{-\frac{b_{it}^3}{a_{it}^2} \,|\, t \in I_{low}(\boldsymbol{\alpha}), -F_i(\boldsymbol{\alpha}) \geq -F_t(\boldsymbol{\alpha}) + \lambda\}$.
*Step 4. Select two variables,* $\alpha_i^k$ *and* $\alpha_j^k$, *and fix other variables, for the optimization problem Eq.* (2), *find the optimal solution* $\alpha_i^{k+1}$, $\alpha_j^{k+1}$. $\boldsymbol{\alpha}^k$ *is updated to* $\boldsymbol{\alpha}^{k+1}$.
*Step 5. If there is any variable violation of the KKT condition in* $\boldsymbol{\alpha}^{k+1}$, *set* $k = k + 1$, *go to 2.*
*Step 6. Get the solution* $\boldsymbol{\alpha}^{k+1}$.

In the above algorithm, the parameters $C$ and $g$ can be cross-validated to obtain the optimal value. In our experiments, we find that using this algorithm will effectively reduce the number of iterations of selecting violating pairs in most cases. In next section, we would like to show the result of our experiments.

## 5 Experiments

In this section, we show some experimental results for comparing the proposed multi-order information with second order information, which is currently the most common working set selection algorithm. LIBSVM, widely used in scientific research and industrial production, is an integrated software proposed by Chang *et al.* [24], which has been using second order information algorithm as its working set selection algorithm since version 2.8. To be fair, we only modify the part of the working set selection algorithm of LIBSVM. We use LIBSVM to represent second order information algorithm in the following paragraphs.

Shrinking and cache are two optimization tools in LIBSVM. The experiments prove that the value of $\alpha_i$ will not change when the $\alpha_i$ reaches the boundary position ($\alpha_i = 0$ or $\alpha_i = C$) during the training process. Under this condition, LIBSVM will remove these multipliers from the working set. The application of cache in LIBSVM greatly reduces training time. The cache can cache the majority of $Q_{i,j}$ used recently, which can be read directly by LIBSVM instead of complex calculation. However, a large amount of memory will be occupied by matrix $Q$ when the training set contains many instances, which is not suitable for prac-

tical applications. Cached $Q_{i,j}$ elements can be deduced by LIBSVM through the Least Recently Used (LRU) algorithm. In addition, the shrinking technology can also help to improve the hit rate of cache since $\alpha_i$ at the boundary location has been removed. In a word, shrinking can affect the number of iterations while cache can not.

All data sets chose are available at the web site of LIBSVM [25]. In order to make the comparison more intuitive, several data sets used in the [17] have been selected. In addition, the data sets used by our experiments also contain numbers of different features and sizes. Besides, the large problems used in [17] are also covered to verify the validity of our algorithm. These data sets are shown in Table 1. In the following paragraphs, we use MOI to represent multi-order information for the working set selection.

Different kernel functions will cause different performance in the same data set. The most commonly used RBF has been considered as the kernel function to process our experiments. Under RBF, we determined the $C$ (in (1)) and kernel parameters $g$ by 10-fold cross validation. We set up several cases of comparison experiments, including the case of no shrinking and cache, the case of using shrinking and 100k cache and the case of using shrinking and 40M cache, which is still consistent with [17]. The training time and the number of iterations are calculated by averaging the results of experiments, which has been performed three times.

We define *Ratio* to measure the quality of MOI and LIBSVM:

$$Ratio = \frac{\text{MOI}}{\text{LIBSVM}} - 1$$

As is shown in the Fig. 1, the number of iterations of almost all data sets has dropped. Some of them have a dramatic decline. For example, heart.txt and svmguide1.txt and w1a.txt are decreased by 17.93%, 13.89% and 27.36% respectively.

However, in Fig. 2, the decline in the respect of time used is not as obvious as in the aspect of the number of iterations. We have found that MOI algorithm has a lower hit rate of cache than LIBSVM through analyzing the record obtained from running the program. One of the reasons is that the number of iterations of MOI is generally lower. Another reason is that there is almost no cache shortage in the program running. Besides, MOI algorithm consumes more time on computing $Q_{i,j}$ than LIBSVM in some data sets, like svmguide1.txt and w1a.txt. After our analysis, we find that this phenomenon is related to the hit rate of cache under shrinking. Due to the existence of shrinking, the same "violating pair" requires different amounts of $Q_{i,j}$ elements in different shrinking stages, and shrinking makes the actual training scale smaller,

Table 1: The data sets used in our experiments

| SMALL PROBLEMS | SIZE | FEATURE |
|---|---|---|
| a1a.txt | 1,605 | 123 |
| a2a.txt | 2,265 | 123 |
| a3a.txt | 3,185 | 123 |
| a4a.txt | 4,781 | 123 |
| australian.txt | 690 | 14 |
| diabetes.txt | 768 | 8 |
| fourclass.txt | 862 | 2 |
| german.numer.txt | 1,000 | 24 |
| gisette | 6,000 | 5,000 |
| heart.txt | 270 | 13 |
| letter.scale | 15,000 | 16 |
| mushrooms.txt | 8124 | 112 |
| splice.txt | 1,000 | 60 |
| svmguide1.txt | 3,089 | 4 |
| w1a.txt | 2,477 | 300 |
| w2a.txt | 3,470 | 300 |
| w3a.txt | 4,912 | 300 |
| w4a.txt | 7,366 | 300 |

| LARGE PROBLEMS | SIZE | FEATURE |
|---|---|---|
| a9a.txt | 32,561 | 123 |
| connect-4 | 67,557 | 126 |
| covtype.libsvm.binary.scale | 581,012 | 54 |
| ijcnn1 | 49,990 | 22 |
| real-sim | 72,309 | 20,958 |
| skin_nonskin | 245,057 | 3 |
| w8a.txt | 49,749 | 300 |

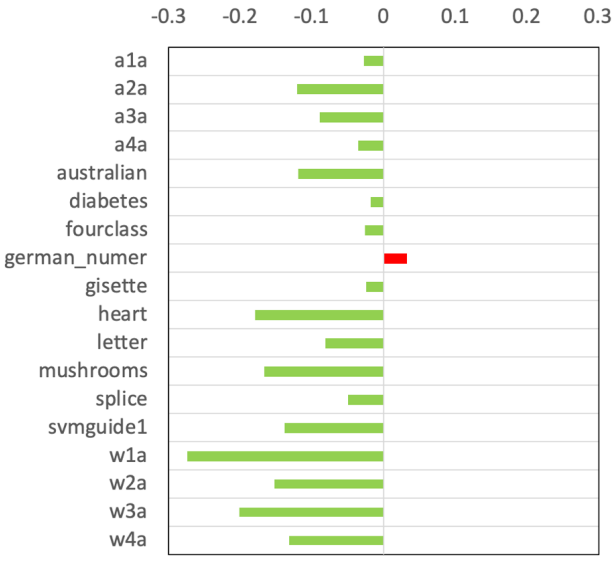


Figure 1: MOI vs. LIBSVM($Ratio(iters)$) with shrinking

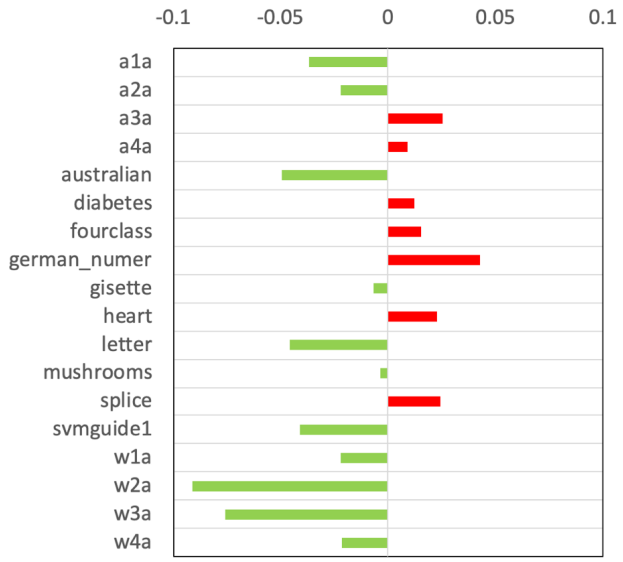


Figure 2: MOI vs. LIBSVM($Ratio(time)$) with shrinking and 40M cache

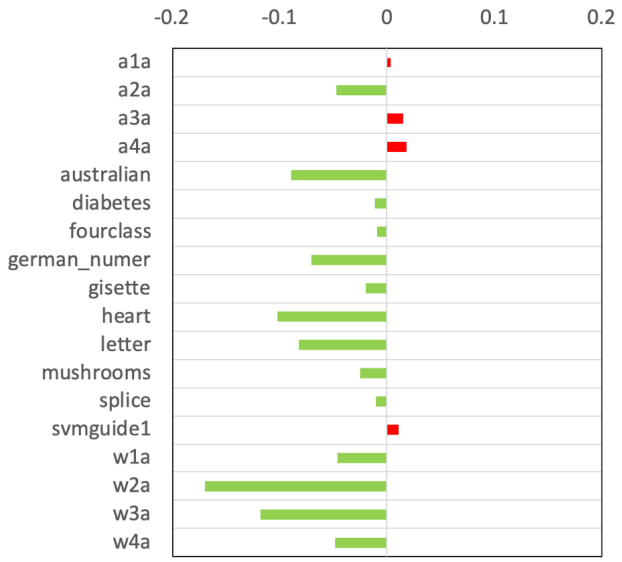


Figure 3: MOI vs. LIBSVM($Ratio(time)$) with shrinking and 100k cache

thus time-consuming of calculating is also getting less. We have analyzed several data sets and found that the problem is mainly caused by the first phase of shrinking. Because the hit rate of cache of the first phase of shrinking is lower than LIBSVM, MOI will calculate more $Q_{i,j}$ elements. At other phases, the impact is smaller.

From Fig. 3 (the size of cache does not affect the number of iterations, so the number of iterations is still referred to Fig. 1), the data can be used to initially verify our previous conjecture. In the 100k cache, due to insufficient cache, each data set has a situation where the cache loads new content. Due to the loading of caches, the number of iterations increases and the likelihood of recalculating $Q_{i,j}$ elements increases. Compared with the 40M cache, MOI shows a greater advantage.

To better compare the performance of MOI and LIBSVM under different cache size, we have also done experiments with 100M cache. As shown in Fig. 4, MOI is still more efficient than LIBSVM in most cases.

Besides, MOI has almost shown an overwhelming advantage (Fig. 5 and Fig. 6) without the use of shrinking and cache. Regardless of the number of iterations or the consuming of time, on almost all data sets, MOI can significantly reduce the number of iterations and training time. This also confirms our previous inference: MOI has a fewer number of iterations than LIBSVM in almost all cases.

Therefore, through the previous experiments and analysis, we can conclude that the overall performance of MOI is better than LIBSVM in that MOI can significantly reduces the number of iterations. The utilization of shrinking and cache is not as good as that of LIBSVM. The smaller the cache is, the more obvious the advantage is. When cache is large enough, MOI still performs well in most data sets.

Next, we will do an experiment to solve large problems with a similar procedure. We still choose RBF as the kernel function and use 20% of the training set to obtain the optimal parameters by a similar method as before. The contrast is set between shrinking+cache and non-shrinking+non-cache. Cache size is set to 800M for connect-4, covtype, and skin_nonskin, 350M for the rest. The results are shown in Table 2, Table 3, Table 4, and Table 5 respectively. The validity of our previous conclusions is further validated by these experimental results.

In addition, the models trained by different working set selections are tested on their accuracy. According to almost the same accuracy, it is shown that our algorithm can effectively speed up the SVM and ensuring accuracy.

## 6 Conclusion

A novel working set selection algorithm is presented in this paper. Not only the descent value of the objective function, but also the update step size of the multipliers and the degree of violation of "violating pairs" are used to select working set. MOI algorithm is based on the combination of these elements. The experimental results show the overall performance of MOI algorithm is better than that of LIBSVM.

The advantage is even more significant in the case

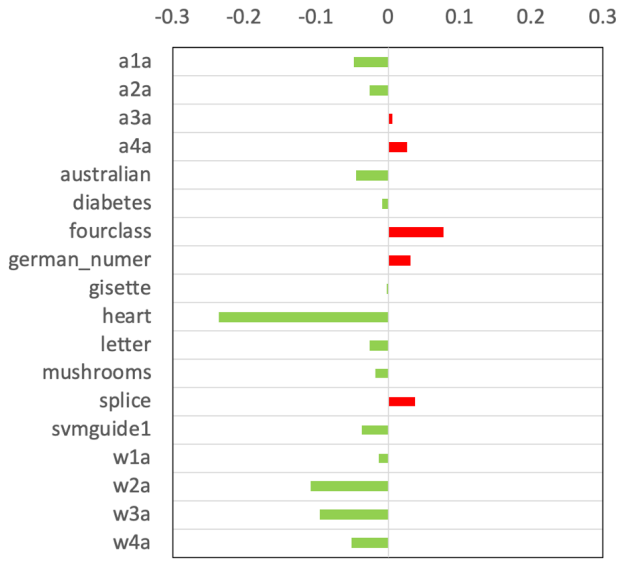


Figure 4: MOI vs. LIBSVM($Ratio(time)$) with shrinking and 100M cache

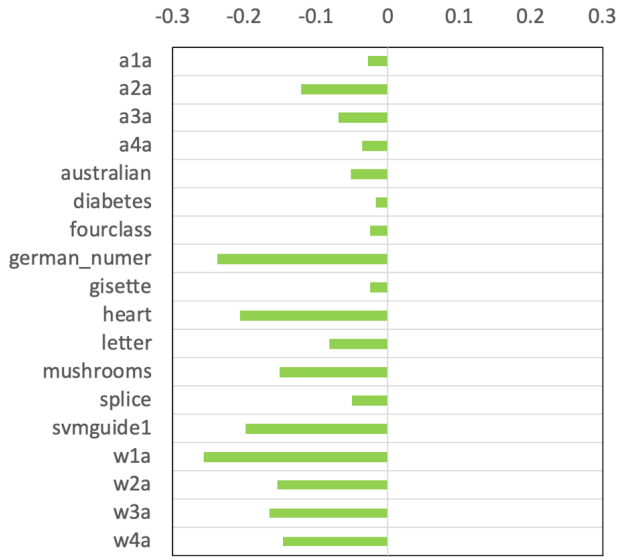


Figure 5: MOI vs. LIBSVM($Ratio(iters)$) no shrinking or cache

of insufficient cache. There has been an evidently improvement in the decrease of the number of iterations on many data sets. If the problem of hit rate of cache is solved, it is anticipated that the performance of MOI will be improved more. In addition, we also expect that the combination of MOI and our previous parallel SMO algorithm will lead to greater improvement.

The big data era brings huge computational challenges to machine learning. It is very necessary to study

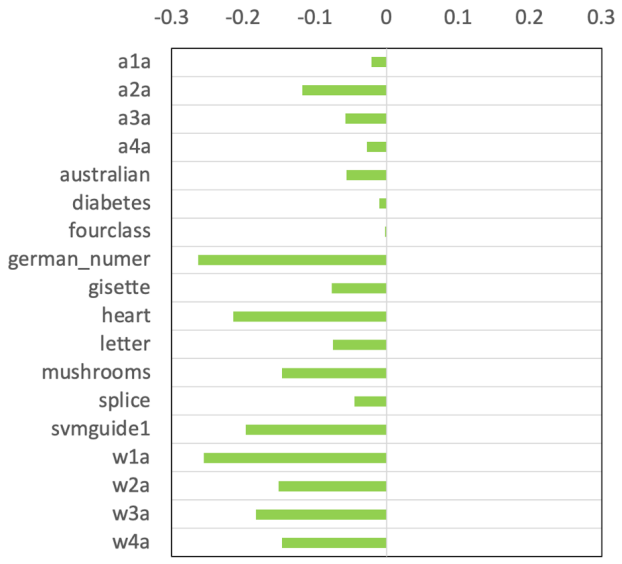


Figure 6: MOI vs. LIBSVM($Ratio(time)$) no shrinking or cache

Table 2: $Ratio(iters)^*$ between MOI and LIBSVM

| | **MOI** | **LIBSVM** | **RATIO** |
|---|---|---|---|
| a9a.txt | 166,053 | 182,170 | -8.85% |
| connect-4 | 514,333 | 620,017 | -17.05% |
| covtype | 124,114 | 128,524 | -3.43% |
| ijcnn1 | 28,957 | 33,050 | -12.38% |
| real-sim | 32,893 | 32,642 | 0.77% |
| skin_nonskin | 6,180 | 8,027 | -23.01% |
| w8a.txt | 18,679 | 22,152 | -15.68% |

∗With shrinking and cache.

Table 3: $Ratio(time)^*$ between MOI and LIBSVM

| | **MOI** | **LIBSVM** | **RATIO** |
|---|---|---|---|
| a9a.txt | 205.712342 | 215.283361 | -4.45% |
| connect-4 | 2046.773589 | 2311.639732 | -11.46% |
| covtype | 3039.733288 | 2947.665176 | 3.12% |
| ijcnn1 | 42.05839 | 41.988279 | 0.17% |
| real-sim | 855.686756 | 857.788728 | -0.25% |
| skin_nonskin | 24.732979 | 25.466065 | -2.88% |
| w8a.txt | 40.337401 | 44.246501 | -8.83% |

∗With shrinking and cache.

the speedup of SVM, since SVM is a particularly effective algorithm for pattern recognition in dealing with nonlinear and high-dimensional features. More importantly, the working set selection algorithm can be used not only in SVM algorithm, but also in some

Table 4: *Ratio*(*iters*)* between MOI and LIBSVM

| | **MOI** | **LIBSVM** | **RATIO** |
|---|---|---|---|
| a9a.txt | 163,300 | 182,988 | -10.76% |
| connect-4 | 513,794 | 616,542 | -16.67% |
| covtype | 121,999 | 124,138 | -1.72% |
| ijcnn1 | 27,589 | 32,490 | -15.08% |
| real-sim | 32,543 | 32,779 | -0.72% |
| skin_nonskin | 6,506 | 7,758 | -16.14% |
| w8a.txt | 18,355 | 21,937 | -16.33% |

∗No shrinking or cache.

Table 5: *Ratio*(*time*)* between MOI and LIBSVM

| | **MOI** | **LIBSVM** | **RATIO** |
|---|---|---|---|
| a9a.txt | 3133.660363 | 3521.985043 | -11.03% |
| connect-4 | 22861.17247 | 27523.58663 | -16.94% |
| covtype | 8606.373334 | 9177.976153 | -6.23% |
| ijcnn1 | 621.445731 | 730.09983 | -14.88% |
| real-sim | 3301.891496 | 3312.284518 | -0.31% |
| skin_nonskin | 478.026983 | 566.385271 | -15.60% |
| w8a.txt | 468.012072 | 563.566522 | -16.96% |

∗No shrinking or cache.

of other algorithms where gradient descent optimization is used. We would like to try to apply MOI to other machine learning algorithms in our future works.

## References


[1] E. Pasolli, F. Melgani, D. Tuia, F. Pacifici, and W. Emery, "SVM Active Learning Approach for Image Classification Using Spatial Information," *IEEE Transactions on Geoscience and Remote Sensing*, 2014.

[2] M. Kachuee, M. M. Kiani, H. Mohammadzade, and M. Shabany, "Cuff-less high-accuracy calibration-free blood pressure estimation using pulse transit time," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015.

[3] T. Bodnar and M. Salathé, "Validating models for disease detection using twitter," in *WWW 2013 Companion - Proceedings of the 22nd International Conference on World Wide Web*, 2013.

[4] W. Li, J. Ge, and G. Dai, "Detecting Malware for Android Platform: An SVM-Based Approach," in *Proceedings - 2nd IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2015 - IEEE International Symposium of Smart Cloud, IEEE SSC 2015*, 2016.

[5] Y. Tang, "Deep learning using support vector machines," *CoRR*, vol. abs/1306.0239, 2013.

[6] X.-X. Niu and C. Y. Suen, "A novel hybrid cnn-svm classifier for recognizing handwritten digits," *Pattern Recogn.*, vol. 45, no. 4, pp. 1318–1325, Apr. 2012.

[7] K. Mori, M. Matsugu, and T. Suzuki, "Face recognition using svm fed with intermediate output of cnn for face detection," in *Iapr Conference on Machine Vision Applications*, 2009, pp. 410–413.

[8] Z. Wang, Z. Wang, H. Zhang, and X. Guo, "A novel fire detection approach based on cnn-svm using tensorflow," in *Intelligent Computing Methodologies*, D.-S. Huang, A. Hussain, K. Han, and M. M. Gromiha, Eds. Cham: Springer International Publishing, 2017, pp. 682–693.

[9] D.-X. Xue, R. Zhang, H. Feng, and Y.-L. Wang, "Cnn-svm for microvascular morphological type recognition with data augmentation," *Journal of Medical and Biological Engineering*, vol. 36, no. 6, pp. 755–764, Dec 2016.

[10] S. Shalev-Shwartz, Y. Singer, and N. Srebro, "Pegasos: Primal estimated sub-gradient solver for svm," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. New York, NY, USA: ACM, 2007, pp. 807–814.

[11] H. P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik, "Parallel support vector machines: The cascade svm," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. MIT Press, 2005, pp. 521–528.

[12] L. Cao, S. Keerthi, C. Ong, J. Qiu Zhang, U. Periyathamby, X. Ju Fu, and H. P Lee, "Parallel sequential minimal optimization for the training of support vector machines," vol. 17, pp. 1039–49, 08 2006.

[13] B. Li, Q. Wang, and J. Hu, "A fast svm training method for very large datasets," pp. 1784 – 1789, 07 2009.

[14] X. Wang and J. Guo, *An Algorithm for Parallelizing Sequential Minimal Optimization.* Springer Berlin Heidelberg, 2013.

[15] W. Wei, C. Li, and J. Guo, "Improved parallel algorithms for sequential minimal optimization of classification problems," in *Proceedingds of the*

*20th IEEE International Conference on High Performance Computing and Communications*, 2018, pp. 6–13.

[16] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to platt's smo algorithm for svm classifier design," *Neural Comput.*, vol. 13, no. 3, pp. 637–649, Mar. 2001.

[17] 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*, vol. 6, no. 4, pp. 1889–1918, 2005.

[18] V. N. Vapnik, *Statistical Learning Theory*. Wiley-Interscience, 1998.

[19] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Berlin, Heidelberg: Springer-Verlag, 1995.

[20] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," in *Neural Networks for Signal Processing*, 1997, pp. 276–285.

[21] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in kernel methods*, 1998, pp. 169–184.

[22] J. C. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," in *Advances in Kernel Methods-support Vector Learning*, 1998, pp. 212–223.

[23] P. hsuen Chen, R. en Fan, and C. jen Lin, "A study on smo-type decomposition methods for support vector machines," *IEEE Transactions on Neural Networks*, vol. 17, pp. 893–908, 2006.

[24] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *Acm Transactions on Intelligent Systems Technology*, vol. 2, no. 3, pp. 1–27, 2011.

[25] LIBSVM Data [Online]. Available: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`