A Simple Decomposition Method for Support Vector Machines

Chih-Wei Hsu and Chih-Jen Lin

Department of Computer Science and
Information Engineering
National Taiwan University
Taipei 106, Taiwan
TEL 886-2-2362-5336 ext. 413
FAX 886-2-2362-8167
cjlin@csie.ntu.edu.tw

Abstract

The decomposition method is currently one of the major methods for solving support vector machines. An important issue of this method is the selection of working sets. In this paper through the design of decomposition methods for bound-constrained SVM formulations we demonstrate that the working set selection is not a trivial task. Then from the experimental analysis we propose a simple selection of the working set which leads to faster convergences for difficult cases. Numerical experiments on different types of problems are conducted to demonstrate the viability of the proposed method.

keywords: Support vector machines, decomposition methods, classification.

1 Introduction

The support vector machine (SVM) is a new and promising technique for classification. Surveys of SVM are, for example, Vapnik (1995, 1998) and Schölkopf et al. (1998). Given training vectors $x_i \in R^n, i = 1, \ldots, l$, in two classes, and a vector $y \in R^l$ such that $y_i \in \{1, -1\}$, the support vector technique requires the solution of the following optimization problem:

$$
\begin{aligned}
\min \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\
& y_i (w^T \phi(x_i) + b) \geq 1 - \xi_i, \\
& \xi_i \geq 0, i = 1, \ldots, l.
\end{aligned} \tag{1.1}
$$

Training vectors x_i are mapped into a higher (maybe infinite) dimensional space by the function ϕ. The existing common method to solve (1.1) is through its dual, a finite quadratic programming problem:

$$
\begin{aligned}
\min \quad & \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \\
& 0 \leq \alpha_i \leq C, \qquad i = 1, \ldots, l \\
& y^T \alpha = 0,
\end{aligned} \tag{1.2}
$$

where e is the vector of all ones, C is the upper bound of all variables, Q is an l by l positive semidefinite matrix, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, and $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ is the kernel.

The difficulty of solving (1.2) is the density of Q because Q_{ij} is in general not zero. In this case, Q becomes a fully dense matrix so a prohibitive amount of memory is required to store the matrix. Thus traditional optimization algorithms such as Newton, Quasi Newton, etc., cannot be directly applied. Several authors (for example, Osuna et al. (1997), Joachims (1998), Platt (1998), and Saunders et al. (1998)) have proposed decomposition methods to conquer this difficulty and reported good numerical results. Basically they separate the index $\{1, \ldots, l\}$ of the training set to two sets B and N, where B is the working set and $N = \{1, \ldots, l\} \backslash B$. If we denote α_B and α_N as vectors containing corresponding elements, the objective value is equal to $\frac{1}{2} \alpha_B^T Q_{BB} \alpha_B - (e_B - Q_{BN} \alpha_N)^T \alpha_B + \frac{1}{2} \alpha_N^T Q_{NN} \alpha_N - e_N^T \alpha_N$. At each iteration, α_N is fixed and the following sub-problem with the variable α_B is solved:

$$
\begin{aligned}
\min \quad & \frac{1}{2} \alpha_B^T Q_{BB} \alpha_B - (e_B - Q_{BN} \alpha_N)^T \alpha_B \\
& 0 \leq (\alpha_B)_i \leq C, i = 1, \ldots, q, \\
& y_B^T \alpha_B = -y_N^T \alpha_N,
\end{aligned} \tag{1.3}
$$

where $\begin{bmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{bmatrix}$ is a permutation of the matrix Q and q is the size of B. The strict decrease of the objective function holds and under some conditions this method converges to an optimal solution.

Usually a special storage using the idea of a cache is used to store recently used Q_{ij}. Hence the computational cost of later iterations can be reduced. However, the computational time is still strongly related to the number of iterations. As the main thing which affects the number of iterations is the selection of working sets, a careful choice of B can dramatically reduce the computational time. This will be the main topic of this paper.

Instead of (1.1), in this paper, we decide to work on a different SVM formulation:

$$
\begin{aligned}
\min \quad & \frac{1}{2}w^T w + \frac{1}{2}b^2 + C\sum_{i=1}^{l} \xi_i \\
& y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \\
& \xi_i \geq 0, i = 1, \ldots, l.
\end{aligned}
\tag{1.4}
$$

Its dual becomes a simpler bound-constrained problem:

$$
\begin{aligned}
\min \quad & \frac{1}{2}\alpha^T (Q + yy^T)\alpha - e^T \alpha \\
& 0 \leq \alpha_i \leq C, \qquad i = 1, \ldots, l.
\end{aligned}
\tag{1.5}
$$

This formulation was proposed and studied by Mangasarian and Musicant (1999), and Friess et al. (1998). We think it is easier to handle a problem without general linear constraints. Later on we will demonstrate that (1.5) is an acceptable formulation in terms of generalization errors though an additional term $b^2/2$ is added to the objective function.

As (1.5) is only a little different from (1.2), in Section 2, naturally we adopt existing working set selections for (1.5). Surprisingly we fail on such an extension. Then through different experiments we demonstrate that finding a good strategy is not a trivial task. That is, experiments and analysis must be conducted before judging whether a working set selection is useful for one optimization formulation or not. Based on our observations, we propose a simple selection which leads to faster convergences in difficult cases. In Section 3, we implement the proposed algorithm as the software BSVM and compare it with SVM^{light} (Joachims, 1998) on problems with different size. After obtaining classifiers from training data, we also compare error rates for classifying test data by using (1.2) and (1.5). We then apply the proposed working set selection to the standard SVM formulation (1.2). Results in Section 4 indicate that this selection strategy also performs very well. Finally in Section 5, we present discussions and conclusions.

The software BSVM is available at the authors' homepage.[‡]

2 Selection of the Working Set

Among existing methods, Osuna et al. (1997), and Saunders et al. (1998) find the working set by choosing elements which violate the KKT condition. Platt (1998) has a special heuristic but his algorithm is mainly on the case when $q = 2$. A systematic way is proposed by Joachims (1998). In his software SVM^{light}, the following problem is solved:

$$\begin{aligned} \min \quad & \nabla f(\alpha_k)^T d \\ & y^T d = 0, \ -1 \leq d \leq 1, && \text{(2.1a)} \\ & d_i \geq 0, \ \text{if } (\alpha_k)_i = 0, \ d_i \leq 0, \ \text{if } (\alpha_k)_i = C, \\ & |\{d_i \mid d_i \neq 0\}| \leq q, && \text{(2.1b)} \end{aligned}$$

where we represent $f(\alpha) \equiv \frac{1}{2}\alpha^T Q \alpha - e^T \alpha$, α_k is the solution at the kth iteration, $\nabla f(\alpha_k)$ is the gradient of $f(\alpha)$ at α_k. Note that $|\{d_i \mid d_i \neq 0\}|$ means the number of components of d which are not zero. The constraint (2.1b) implies that a descent direction involving only q variables is obtained. Then components of α_k with non-zero d_i are included in the working set B which is used to construct the sub-problem (2.7). Note that d is only used for identifying B but not as a search direction. Joachims (1998) showed that the computational time for solving (2.1) is mainly on finding the $q/2$ largest and $q/2$ smallest elements of $y_i \nabla f(\alpha_k)_i, i = 1, \ldots, l$ with $y_i d_i = 1$ and $y_i d_i = -1$, respectively. Hence the cost is at most $O(ql)$ which is affordable in his implementation.

Therefore, following SVM^{light}, for solving (1.5), a natural method to choose the working set B is by a similar problem of (2.1):

$$\begin{aligned} \min \quad & \nabla f(\alpha_k)^T d \\ & -1 \leq d \leq 1, && \text{(2.2a)} \\ & d_i \geq 0, \ \text{if } (\alpha_k)_i = 0, \ d_i \leq 0, \ \text{if } (\alpha_k)_i = C, && \text{(2.2b)} \\ & |\{d_i \mid d_i \neq 0\}| \leq q, && \text{(2.2c)} \end{aligned}$$

where $f(\alpha)$ becomes $\frac{1}{2}\alpha^T(Q + yy^T)\alpha - e^T\alpha$. Note that SVM^{light}'s working set selection violates the feasibility condition $0 \leq \alpha_k + d \leq C$. Therefore, instead of (2.2a)-(2.2c), we can consider other types of constraints such as

$$0 \leq \alpha_k + d \leq C, \ |\{d_i \mid d_i \neq 0\}| = q. \tag{2.3}$$

[‡] BSVM is available at `http://www.csie.ntu.edu.tw/~cjlin/bsvm`

The convergence of using (2.3) is guaranteed under the framework of Chang et al. (2000). For SVM^{light}'s selection (2.2), the convergence is more complicated and has just been proved by Lin (2000) very recently.

Note that solving (2.2) is very easy by calculating the following vector:

$$v_i = \begin{cases} \min(\nabla f(\alpha_k)_i, 0) & \text{if } (\alpha_k)_i = 0, \\ -|\nabla f(\alpha_k)_i| & \text{if } 0 < (\alpha_k)_i < C, \\ -\max(\nabla f(\alpha_k)_i, 0) & \text{if } (\alpha_k)_i = C. \end{cases} \tag{2.4}$$

Then B contains indices of the smallest q elements of v. Interestingly, for the bound-constrained formulation, solving (2.2) is the same as finding maximal violated elements of the KKT condition. Note that elements which violate the KKT condition are

$$\begin{aligned} \nabla f(\alpha_k)_i &< 0 && \text{if } (\alpha_k)_i = 0, \\ \nabla f(\alpha_k)_i &> 0 && \text{if } (\alpha_k)_i = C, \\ \nabla f(\alpha_k)_i &\neq 0 && \text{if } 0 < (\alpha_k)_i < C. \end{aligned}$$

Hence, the q maximal violated elements are the smallest q elements of v. A similar relation between the KKT condition of (1.2) and (2.1b) is not very easy to see as the KKT condition of (1.2) involves the number b which does not appear in (2.1b). An interpretation is given by Laskov (1999) where he considers possible intervals of b and select the most violated points through end points of these intervals. These approaches are reasonable as intuitively we think that finding the most violated elements is a natural choice.

However, unlike SVM^{light}, this selection of the working set does not perform well. In the rest of this paper we name our implementation BSVM. In Table 1, by solving the problem heart from the Statlog collection (Michie, Spiegelhalter, & Taylor, 1994), we can see that BSVM performs much worse than SVM^{light}.

Table 1: Problem heart: comparison in early iterations

BSVM						SVM^{light}					
iter.	obj.	#free	$\#C$	$y_i = 1$	-1	iter.	obj.	#free	$\#C$	$y_i = 1$	-1
1	0.00	0	0	4	6	1	0.00	0	0	5	5
2	-6.68	3	7	10	0	2	-7.22	0	10	5	5
3	-7.69	8	7	0	10	3	-14.49	4	16	5	5
4	-9.57	14	7	10	0	4	-23.24	8	22	5	5
5	-11.12	19	7	0	10	5	-30.78	11	27	5	5
6	-12.58	25	7	10	0	6	-38.33	12	33	5	5
7	-13.91	28	7	0	10	7	-42.86	18	36	5	5
590	-100.94	24	107	9	1	63	-100.88	25	107	7	3

SVM^{light} takes only 63 iterations but BSVM takes 590 iterations. In this experiment, we use $K(x_i, x_j) = e^{-\|x_i - x_j\|^2/n}$, $q = 10$, and $C = 1$. Both methods use similar stopping criteria and the initial solution is zero. The column "#C" presents the number of elements of α_k which are at the upper bound C. Columns $y_i = \pm 1$ report number of elements in B in two different classes. Note that here for easy experiments, we use simple implementation of SVM^{light} and BSVM written in MATLAB.

We observe that in each of the early iterations, all components of the working set are in the same class. The decrease of objective values of BSVM is much slower than that of SVM^{light}. In addition, the number of variables at the upper bound after seven SVM^{light} iterations is more than that of BSVM while BSVM produces iterations with more free variables. We explain this observation as follows. First we make some assumptions for easy description:

1. α and $\alpha + d$ are solutions of the current and next iterations, respectively.
2. B is the current working set, where all elements are from the same class with $y_i = 1$, that is, $y_B = e_B$. In other words, we assume that at one iteration, the situation where all elements of the working are in the same class happens.
3. $\alpha_B = 0$.

We will show that it is more possible that components with $y_i = -1$ will be selected in the next working set. Note that $d_N = 0$ and since $\alpha_B = 0$, $d_B \geq 0$. Since $d \geq 0$, $y_B = e_B$, $(Q + yy^T)_{ij} = y_i y_j (e^{-\|x_i - x_j\|^2/n} + 1)$ and $(e^{-\|x_i - x_j\|^2/n} + 1) > 0$,

$$((Q + yy^T)d)_i = \sum_{j \in B} y_i y_j (e^{-\|x_i - x_j\|^2/n} + 1) d_j \geq 0, \;\text{ if } y_i = 1, \tag{2.5}$$

$$\leq 0, \;\text{ if } y_i = -1. \tag{2.6}$$

We also know that

$$\nabla f(\alpha + d) = \nabla f(\alpha) + (Q + yy^T)d.$$

In early iterations, most elements of α are at zero. If for those nonzero elements, v_i of (2.4) are not too small, (2.2) is essentially finding the smallest elements of $\nabla f(\alpha + d)$. For the next iteration, since $v_i = 0, i \in B$, elements in B will not be included the working set again. For elements in N, from (2.5) and (2.6), we have

$$\nabla f(\alpha + d)_i \geq \nabla f(\alpha)_i, \;\text{ if } y_i = 1,$$
$$\nabla f(\alpha + d)_i \leq \nabla f(\alpha)_i, \;\text{ if } y_i = -1.$$

Therefore, if $((Q + yy^T)d)_i$ is not small, in the next iteration, elements with $y_i = -1$ tend to be selected because $\nabla f(\alpha + d)_i$ becomes smaller. For the example in Table 1, we really observe that the sign of most $\nabla f(\alpha)_i$ is changed in every early iteration. The explanation is similar if $y_B = -1$.

We note that (2.5) and (2.6) hold because of the RBF kernel. For another popular kernel: the polynomial kernel $(x_i^T x_j/n)^d$, if all attributes of data are scaled to $[-1,1]$, $(x_i^T x_j/n)^d + 1 > 0$ still holds. Hence (2.5) and (2.6) remain valid.

Next we explain that when the above situation happens, in early iterations, very few variables can reach the upper bound. Since we are solving the following sub-problem:

$$\begin{aligned} \min \quad & \frac{1}{2}\alpha_B^T(Q_{BB} + y_B y_B^T)\alpha_B - (e_B - (Q_{BN} + y_B y_N^T)\alpha_N)^T \alpha_B \\ & 0 \le \alpha_B \le C, \end{aligned} \tag{2.7}$$

it is like that the following primal problem is solved:

$$\begin{aligned} \min \quad & \frac{1}{2}w^T w + \frac{1}{2}b^2 + C\sum_{i \in B} \xi_i \\ & y_i(w^T\phi(x_i) + b) \ge 1 - \sum_{j \in N} Q_{ij}\alpha_j - \xi_i, i \in B, \\ & \xi_i \ge 0, i \in B. \end{aligned} \tag{2.8}$$

Table 2: Problem heart: a new comparison in early iterations

BSVM				SVM^{light}			
iter.	obj.	#free	$\#C$	iter.	obj.	#free	$\#C$
1	0.00	0	0	1	0.00	0	0
2	-7.19	1	9	2	-7.22	0	10
3	-13.68	5	14	3	-14.49	4	16
4	-21.99	9	20	4	-23.24	8	22
5	-28.77	12	24	5	-30.78	11	27
6	-36.21	10	33	6	-38.33	12	33
7	-41.93	15	36	7	-42.86	18	36
388	-100.94	24	107	63	-100.88	25	107

If components in B are in the same class, (2.8) is a problem with separable data. Hence $\xi_i = 0, i \in B$ implies that α_B are in general not equal to C. Thus the algorithm has difficulties to identify correct bounded variables. In addition, the decrease of the objective function becomes slow. On the other hand, the constraint $y^T d = 0$ in (2.1) of SVM^{light} provides an excellent channel for selecting the working set from two different classes. Remember that SVM^{light} selects the largest $q/2$ and smallest $q/2$ elements of $y_i \nabla f(\alpha_k)_i, i = 1, \ldots, l$ with $y_i d_i = -1$ and $y_i d_i = 1$, respectively. Thus when most $(\alpha_k)_i$ are zero, $d_i \ge 0$ so the largest elements of $y_i \nabla f(\alpha_k)_i$ are mainly from indices with $y_i = -1$ and $\nabla f(\alpha_k)_i < 0$. Conversely, the smallest elements of $y_i \nabla f(\alpha_k)_i$ are from data with $y_i = 1$ and $\nabla f(\alpha_k)_i < 0$. This can be seen in the columns of the number of $y_i = \pm 1$ in Table 1.

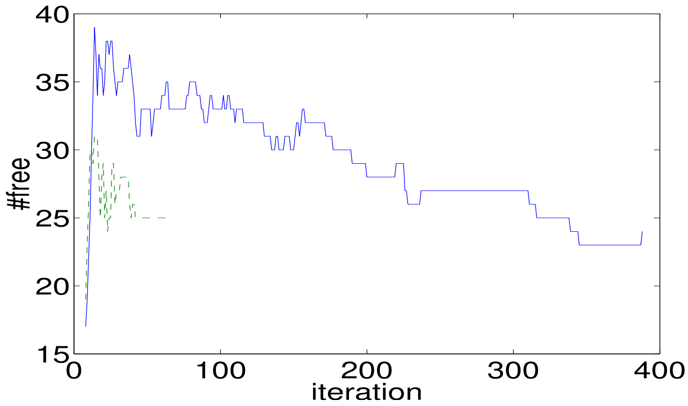


Figure 1: # of free variables (line: BSVM, dashed: SVM^{light})

Another explanation is from the first and second order approximations of the optimization problem. If $[\alpha_B, \alpha_N]$ is the current solution which is considered as a fixed vector and d is the variable, problem (1.3) is equivalent to solving

$$\begin{aligned} \min \quad & \frac{1}{2} d^T Q_{BB} d - (e_B - Q_{BN}\alpha_N - Q_{BB}\alpha_B)^T d \\ & 0 \leq \alpha_B + d \leq C, \\ & y_B^T d = 0. \end{aligned} \tag{2.9}$$

Since

$$-(e_B - Q_{BN}\alpha_N - Q_{BB}\alpha_B) = \nabla f(\alpha)_B,$$

(2.1) is like to select the best q elements such that the linear part of (2.9) is minimized. Similarly, (2.7) is equivalent to solving

$$\begin{aligned} \min \quad & \frac{1}{2} d^T (Q_{BB} + y_B y_B^T) d - (e_B - (Q_{BN} + y_B y_N^T)\alpha_N - (Q_{BB} + y_B y_B^T)\alpha_B)^T d \\ & 0 \leq \alpha_B + d \leq C. \end{aligned} \tag{2.10}$$

Clearly a main difference between (2.9) and (2.10) is that y_B involves in a term $d^T(y_B y_B^T)d = (y_B^T d)^2$ in the objective value of (2.10) but for (2.9), y_B appears in one of its constraints: $y_B^T d = 0$. Therefore, since we are now using a linear approximation for selecting the working set, for (2.10), $d^T(y_B y_B^T)d$ is a quadratic term which is not considered in (2.2). Thus (2.2) does not know that $(y_B^T d)^2$ should not be too large. On the other hand, for (2.1), $y_B^T d = 0$ remains as a constraint so it is like that $(y_B^T d)^2$ is implicitly minimized. In one word, (2.1) and (2.2) both come from the linear approximation but one contains more information than the other.

Based on this observation, we try to modify (2.2) by selecting a d which contains the best $q/2$ elements with $y_i = 1$ and the best $q/2$ elements with $y_i = -1$. The new result is in Table 2 where a substantial improvement is obtained. Now after seven iterations, both algorithms reach solutions which have the same number of components at the upper bound. Objective values are also similar.

However, in Table 2, BSVM still takes more iterations than SVM^{light}. We observe very slow convergence in final iterations. To improve the performance, we analyze the algorithm in more detail. In Figure 1, we present the number of free variables in each iteration. It can be clearly seen that BSVM goes through points which have more free variables. Since the weakest part of the decomposition method is that it cannot consider all variables together in each iteration (only q are selected), a larger number of free variables causes more difficulty. In other words, if a component is correctly identified at C or 0, there is no problem of numerical accuracy. However, it is in general not easy to decide the value of a free variable if all free variables are not considered together. Comparing to the working set selection (2.1) of SVM^{light}, our strategy of selecting $q/2$ elements with $y_i = 1$ and $q/2$ elements with $y_i = -1$ is not very natural. Therefore, in the middle of the iterative process more variables are not correctly identified at the upper bound so the number of free variables becomes larger. This leads us to conjecture that we should keep the number of free variables as small as possible. A possible strategy to achieve this is by adding some free variables in the previous iteration to the current working set.

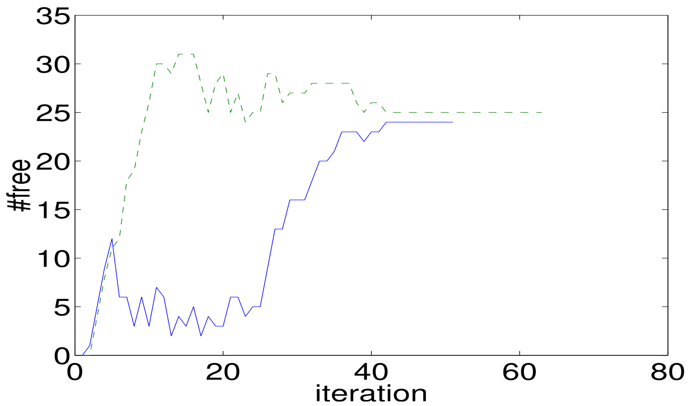


Figure 2: # of free variables (line: BSVM, dashed: SVM^{light})

The second observation is on elements of the working set. When we use (2.2) to select B, in final iterations, components of the working set are shown in Table 3. In Table 4, working sets of running SVM^{light} are presented. From the last column of Table 3, it can be

Table 3: Working sets of final iterations: BSVM

iter.	B	# in iter.+2
382	[16 30 53 23 5 228 130 200 7 90]	8
383	[168 24 245 197 119 51 134 108 7 90]	8
384	[16 23 30 53 266 200 130 228 51 90]	8
385	[24 195 25 119 197 108 134 7 51 90]	8
386	[16 30 53 5 23 200 228 130 51 7]	8
387	[197 245 24 119 195 108 90 134 51 228]	
388	[16 30 23 53 5 200 130 7 134 108]	

Table 4: Working set in final iterations: SVM^{light}

iter.	B	# in iter.+2
57	[245 16 108 90 119 230 7 195 51 49]	7
58	[266 130 51 230 195 25 5 30 53 197]	3
59	[245 16 108 119 90 195 30 266 49 24]	5
60	[168 7 130 228 19 200 230 73 5 197]	3
61	[245 16 108 30 119 230 19 7 228 5]	5
62	[25 30 19 119 7 168 53 73 195 197]	
63	[245 108 90 16 24 7 30 25 195 168]	

seen that the working set of the kth iteration is very close to that of the $(k+2)$nd iteration. However, in Table 4, this situation is not that serious. For this example, at final solutions, there are 24 free variables by using (1.5) and 25 by (1.2). For BSVM, in the second half iterations, the number of free variables is less than 30. Note that in final iterations, the algorithm concentrates on deciding the value of free variables. Since in each iteration we select 10 variables in the working set, after the sub-problem (2.7) is solved, gradient at these 10 elements become zero. Hence for the next iteration the solution of (2.2) mainly comes from the other free variables. This explains why working sets of the kth and $(k+2)$nd iterations are so similar. Apparently a selection like that in Table 3 is not an appropriate one. We mentioned earlier that the weakest part of the decomposition method is that it cannot consider all variables together. Now the situation is like two groups of variables are never considered together so the convergence is extremely slow.

Based on these observations, we propose Algorithm 2.1 for the selection of the working set. We have $q/2$ elements from a problem of (2.2) but the other $q/2$ elements are from free components with the largest $-|\nabla f(\alpha_k)_i|$. Since free variables in the previous working set satisfy $\nabla f(\alpha_k)_i = 0$, if there are not too many such elements, most of them are again included in the next working set. There are exceptional situations where all (α_k) are at bounds. When this happens, we choose $q/2$ best elements with $y_i = 1$ and $q/2$ best elements

Table 5: RBF kernel and $C = 1$

		BSVM					SVM^{light} (without shrinking)				
Problem	n	SV(BSV)	Mis.	Obj.	Iter.	Time	SV(BSV)	Mis.	Obj.	Iter.	Time
australian	690	245(189)	98	-201.64	200	0.39	245(190)	98	-201.64	504	0.99
diabetes	768	447(434)	168	-413.57	131	0.44	447(435)	168	-413.56	105	0.39
german	1000	599(514)	189	-502.84	247	1.41	599(512)	189	-502.77	311	1.24
heart	270	130(107)	36	-100.94	53	0.08	132(107)	36	-100.88	66	0.09
vehicle	846	439(414)	210	-413.27	171	0.79	439(414)	210	-413.02	332	0.94
satimage	4435	380(367)	56	-286.85	121	8.40	378(365)	60	-285.35	102	5.36
letter	15000	569(542)	104	-451.18	235	26.24	564(531)	103	-446.98	349	24.38
shuttle	43500	6188(6184)	1117	-5289.17	3036	484.36	6164(6152)	1059	-5241.41	1120	449.27
dna	2000	695(540)	50	-427.91	334	8.99	697(533)	50	-427.38	253	6.14
segment	2310	392(376)	7	-239.99	149	2.53	381(366)	7	-233.17	108	1.58
fourclass	862	411(403)	168	-383.87	189	0.64	408(401)	167	-383.58	124	0.41
adult1	1605	691(586)	228	-567.85	293	2.33	691(584)	228	-567.79	266	2.17
adult4	4781	1888(1655)	700	-1637.58	841	22.07	1888(1651)	700	-1637.56	976	23.11
web1	2477	441(85)	54	-116.59	402	2.30	451(85)	54	-116.45	557	3.70
web4	7366	845(288)	123	-326.12	850	14.86	869(281)	126	-326.07	876	19.82
web7	24692	2021(927)	301	-939.70	2045	171.58	2084(909)	301	-939.71	2384	210.90

Table 6: RBF kernel and $C = 1000$

		BSVM					SVM^{light} (without shrinking)				
Problem	n	SV(BSV)	Mis.	Obj.	Iter.	Time	SV(BSV)	Mis.	Obj.	Iter.	Time
australian	690	222(55)	28	-81212.20	12609	5.99	222(55)	28	-81199.76	101103	199.12
diabetes	768	377(273)	128	-302471.03	21908	10.83	377(273)	128	-302463.71	75506	166.09
german	1000	508(15)	3	-39910.32	16874	12.55	509(15)	3	-39908.72	43411	113.96
heart	270	101(1)	0	-4815.87	1230	0.35	101(1)	0	-4815.72	5772	7.56
vehicle	846	284(172)	47	-179603.67	10238	5.98	284(173)	47	-179593.77	177188	418.53
satimage	4435	136(30)	8	-44228.55	3717	15.94	136(30)	8	-44147.44	32466	249.04
letter	15000	149(53)	6	-55845.30	7636	110.93	152(52)	6	-55325.74	27290	815.87
shuttle	43500	1487(1470)	137	-1188526.85	4446	318.04	1491(1468)	137	-1188414.63	24587	2713.42
dna	2000	404(0)	0	-1454.02	1263	7.18	408(0)	0	-1453.15	778	6.84
segment	2310	27(8)	5	-9165.45	160	1.02	25(7)	5	-9087.98	606	2.92
fourclass	862	91(76)	2	-54642.66	505	0.55	89(74)	2	-53885.73	11191	20.83
adult1	1605	649(24)	14	-41925.38	9098	10.65	656(20)	14	-41925.47	17339	64.83
adult4	4781	1811(162)	97	-279114.93	85103	276.57	1854(144)	97	-279116.00	163209	1702.13
web1	2477	354(16)	9	-19276.98	1090	3.03	373(12)	9	-19276.98	2036	11.07
web4	7366	770(56)	31	-64313.28	3280	27.15	820(48)	31	-64313.24	5679	87.26
web7	24692	1660(231)	126	-260147.98	9884	633.83	1814(213)	126	-260147.88	24538	1483.80

Let r be the number of free variables at α_k
If $r > 0$, then
    Select indices of the largest $\min(q/2, r)$ elements in v, where $(\alpha_k)_i$ is free, into B
    Select the $(q - \min(q/2, r))$ smallest elements in v into B.
else
    Select the $q/2$ smallest elements with $y_i = 1$ and $q/2$ smallest elements with $y_i = -1$ in v into B.

Algorithm 2.1: Selection of the working set

with $y_i = -1$ following the discussion for results in Table 2.

The motivation of this selection is described as follows: consider minimizing $f(\alpha) = \frac{1}{2}\alpha^T A\alpha - e^T\alpha$, where A is a positive semidefinite matrix and there are no constraints. This problem is equivalent to solving $\nabla f(\alpha) = A\alpha - e = 0$. If the decomposition method is used, B_{k-1} is the working set at the $(k-1)$st iteration, and A is written as $\begin{bmatrix} A_{B_{k-1}} \\ A_{N_{k-1}} \end{bmatrix}$, we have $A_{B_{k-1}}\alpha_k = e$. Therefore, similar to what we did in (2.2), we can let B_k, the next working set, contain the smallest q elements of $\nabla f(\alpha_k) = A\alpha_k - e$. In other words, elements violate KKT condition are selected. Thus B_k will not include any elements in B_{k-1} where $\nabla f(\alpha_k)_i = 0$, for all $i \in B_{k-1}$. However, $A_{B_{k-1}}\alpha_k - e = 0$ only holds at the kth iteration. When α is updated to α_{k+1}, the equality fails again. Hence this is like a zigzaging process. From the point of view of solving a linear system, we think that considering some inequalities and equalities together is a better method to avoid the zigzaging process. In addition, our previous obervations suggest the reduction of the number of free variables. Therefore, basically we select the $q/2$ most violated elements from the KKT condition and the $q/2$ most satisfied elements at which α_i is free.

Using Algorithm 2.1, BSVM takes about 50 iterations which is fewer than that of SVM^{light}. Comparing to the 388 iterations presented in Table 2, the improvement is dramatic. In Figure 2, the number of free variables in both methods are presented. It can be clearly seen that in BSVM, the number of free variables is kept small. In early iterations, each time q elements are considered and some of them move to the upper bound. For free variables, Algorithm 2.1 tends to consider them again in subsequent iterations so BSVM has more opportunities to push them to the upper bound. Since now the feasible region is like a box, we can say that BSVM walks closer to walls of this box. We think that in general this is a good property as the decomposition method faces more difficulties on handling free variables.

Table 7: Polynomial kernel and $C = 1$

		BSVM					SVM^{light} (without shrinking)				
Problem	n	SV(BSV)	Mis.	Obj.	Iter.	Time	SV(BSV)	Mis.	Obj.	Iter.	Time
australian	690	283(241)	95	-227.98	112	0.35	284(238)	95	-227.93	188	0.46
diabetes	768	539(532)	231	-499.41	128	0.47	538(532)	230	-499.24	99	0.37
german	1000	626(528)	202	-516.42	238	1.43	625(527)	203	-516.24	236	1.03
heart	270	177(151)	38	-131.81	51	0.10	177(151)	38	-131.80	62	0.09
vehicle	846	436(415)	212	-422.69	94	0.69	443(403)	212	-422.18	217	0.69
satimage	4435	2126(2109)	839	-1827.68	308	37.20	2131(2104)	839	-1827.21	391	22.10
letter	15000	1038(1019)	172	-783.91	196	35.11	1040(1015)	172	-783.05	223	28.17
shuttle	43500	17699(17698)	7248	-15560.58	3705	1083.50	17700(17694)	7248	-15560.17	3008	1118.19
dna	2000	1102(775)	464	-906.09	229	13.94	1103(772)	464	-905.58	216	7.66
segment	2310	581(566)	19	-379.98	149	3.28	582(566)	19	-379.20	130	1.90
fourclass	862	485(480)	195	-444.25	228	0.53	485(479)	195	-443.76	125	0.41
adult1	1605	762(623)	241	-599.84	234	2.26	761(623)	242	-599.55	216	1.97
adult4	4781	2007(1716)	721	-1684.41	743	22.36	2011(1716)	721	-1684.09	647	19.64
web1	2477	539(88)	57	-123.85	442	2.73	552(88)	57	-123.35	492	3.57
web4	7366	1110(284)	176	-373.41	1009	18.85	1143(273)	176	-372.90	1458	29.92
web7	24692	2647(1027)	577	-1214.22	2474	266.33	2755(1005)	577	-1213.73	3993	326.41

Table 8: Polynomial kernel and $C = 1000$

		BSVM					SVM^{light} (without shrinking)				
Problem	n	SV(BSV)	Mis.	Obj.	Iter.	Time	SV(BSV)	Mis.	Obj.	Iter.	Time
australian	690	217(64)	31	-88213.03	6987	3.41	216(64)	31	-88208.01	40630	82.39
diabetes	768	387(311)	138	-332791.89	6324	3.28	387(311)	138	-332791.88	16976	35.81
german	1000	484(17)	3	-43186.83	14083	10.54	485(17)	3	-43186.40	30546	78.50
heart	270	105(1)	0	-6118.35	760	0.25	105(1)	0	-6117.89	2943	3.64
vehicle	846	339(258)	81	-247281.87	2002	1.65	340(258)	81	-247279.97	6716	15.04
satimage	4435	307(232)	54	-200995.69	427	9.51	307(232)	54	-200995.34	1292	14.22
letter	15000	258(144)	50	-142509.32	1396	32.35	259(144)	50	-142508.03	3718	120.14
shuttle	43500	4626(4613)	615	-3789837.97	3312	399.35	4630(4606)	615	-3789835.86	1327	400.72
dna	2000	1051(0)	0	-17293.01	1097	14.64	1062(0)	0	-17292.62	493	8.99
segment	2310	34(9)	4	-11416.39	185	1.12	37(9)	4	-11403.03	1610	7.12
fourclass	862	364(359)	177	-360720.39	435	2.22	365(358)	177	-360720.34	6817	34.94
adult1	1605	668(24)	14	-41990.62	5830	7.53	672(20)	14	-41990.39	9898	37.59
adult4	4781	1759(167)	100	-288125.96	60938	201.75	1799(153)	100	-288125.21	102111	1027.01
web1	2477	356(36)	16	-41713.78	672	2.43	362(32)	16	-41713.26	658	4.13
web4	7366	805(87)	43	-102103.18	2387	22.61	881(78)	43	-102102.62	2923	49.03
web7	24692	1683(294)	147	-343024.14	6470	432.96	1857(270)	147	-343023.74	15697	998.56

3 Computational Experiments

In this section, we describe the implementation of BSVM and present the comparison between BSVM (Version 1.1) and SVM^{light} (Version 3.2). Results show that BSVM converges faster than SVM^{light} for difficult cases. The computational experiments for this section were done on a Pentium III-500 using the gcc compiler.

SVM^{light} uses the following conditions as the termination criteria:

$$\begin{aligned} (Q\alpha)_i - 1 + by_i &\geq -\epsilon, && \text{if } \alpha_i < \epsilon_a, \\ (Q\alpha)_i - 1 + by_i &\leq \epsilon, && \text{if } \alpha_i > C - \epsilon_a, \\ -\epsilon \leq (Q\alpha)_i - 1 + by_i &\leq \epsilon, && \text{otherwise}, \end{aligned} \tag{3.1}$$

where $\epsilon_a = 10^{-12}$. To have a fair comparison, we use similar criteria in BSVM:

$$\begin{aligned} -\epsilon \leq ((Q + yy^T)\alpha)_i - 1 &\leq \epsilon, && \text{if } 0 < \alpha_i < C, \\ ((Q + yy^T)\alpha)_i - 1 &\geq -\epsilon, && \text{if } \alpha_i = 0, \\ ((Q + yy^T)\alpha)_i - 1 &\leq \epsilon, && \text{if } \alpha_i = C. \end{aligned} \tag{3.2}$$

Note that now there is no b in the above conditions. For both SVM^{light} and BSVM, we set $\epsilon = 10^{-3}$.

We solve the sub-problem (2.7) by modifying the software TRON by Lin and Moré (1999) [§]. TRON is designed for large sparse bound-constrained problems. Here the sub-problem is a very small fully dense problem so we cannot directly use it.

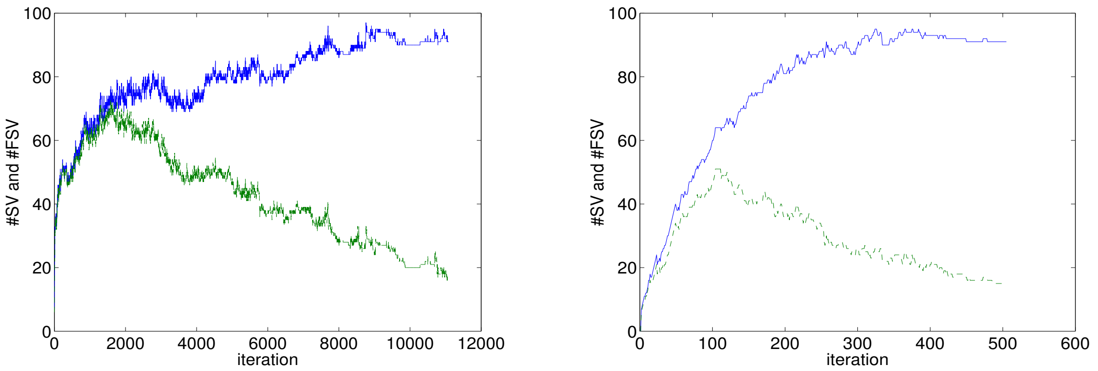
As pointed out in existing work of decomposition methods, the most expensive step in each iteration is the evaluation of the q columns of the matrix Q. In other words, we maintain the vector $Q\alpha$ so in each iteration, we have to calculate $Q(\alpha_{k+1} - \alpha_k)$ which involves q columns of Q. To avoid the recomputation of these columns, existing methods use the idea of a cache where recently used columns are stored. In BSVM, we now have a very simple implementation of the least-recently-used caching strategy. In the future, we plan to optimize its performance using more advanced implementation techniques. For experiments in this section, we use 160MB as the cache size for both BSVM and SVM^{light}.

We test problems from different collections. Problems australian to segment are from the Statlog collection (Michie et al., 1994). Problem fourclass is from Ho and Kleinberg (1996). Problems adult1 and adult4 are compiled by Platt (1998) from the UCI "adult" data set (Murphy & Aha, 1994). Problems web1 to web7 are also from Platt. Note that all problems from Statlog (except dna) and fourclass are with real numbers so we scale them to $[-1, 1]$.

[§] TRON is available at `http://www.mcs.anl.gov/~more/tron`

Table 9: RBF kernel: Using SVM^{light} with shrinking

		$C=1$					$C=1000$				
Problem	n	SV(BSV)	Mis.	Obj.	Iter.	Time	SV(BSV)	Mis.	Obj.	Iter.	Time
australian	690	245(190)	98	-201.64	504	0.75	223(55)	28	-81199.77	119658	176.44
diabetes	768	447(435)	168	-413.56	105	0.38	377(273)	128	-302463.70	75141	115.26
german	1000	599(512)	189	-502.77	311	1.17	509(15)	3	-39908.72	43411	93.13
heart	270	132(107)	36	-100.88	66	0.09	101(1)	0	-4815.72	5772	7.01
vehicle	846	439(414)	210	-413.02	332	0.82	284(173)	47	-179593.77	177188	268.53
satimage	4435	378(365)	60	-285.35	102	5.35	136(30)	8	-44147.44	32466	71.68
letter	15000	564(531)	103	-446.98	349	20.36	152(52)	6	-55325.74	27290	135.39
shuttle	43500	6164(6152)	1059	-5241.41	1120	478.28	1488(1467)	137	-1188414.66	24699	1243.39
dna	2000	697(533)	50	-427.38	253	6.15	408(0)	0	-1453.15	778	5.93
segment	2310	381(366)	7	-233.17	108	1.56	25(7)	5	-9087.98	606	1.87
fourclass	862	408(401)	167	-383.58	124	0.41	89(74)	2	-53885.74	11866	19.41
adult1	1605	691(584)	228	-567.79	266	2.11	656(20)	14	-41925.47	17339	49.85
adult4	4781	1888(1651)	700	-1637.56	976	19.25	1854(144)	97	-279116.00	163209	1109.83
web1	2477	451(85)	54	-116.45	557	3.05	373(12)	9	-19276.98	2036	7.87
web4	7366	869(281)	126	-326.07	876	15.22	808(48)	31	-64313.24	5547	56.35
web7	24692	2084(909)	301	-939.71	2384	154.10	1819(213)	126	-260147.88	24643	971.40



(a) SVM^{light} (b) BSVM

Figure 3: Problem fourclass: number of SV(line) and free SV(dashed)

Some of these problems have more than 2 classes so we treat all data not in the first class as in the second class. Problems dna, adult, and web are with binary representation so we do not conduct any scaling.

We test problems by using RBF and polynomial kernels. For the RBF kernel, we use $K(x_i, x_j) = e^{-\|x_i - x_j\|^2/n}$ for Statlog problems and fourclass. We use $K(x_i, x_j) = e^{-0.05\|x_i - x_j\|^2}$ for adult and web problems following the setting in Joachims (1998). For the polynomial kernel, similarly we have $K(x_i, x_j) = (x_i^T x_j/n)^3$ and $K(x_i, x_j) = (0.05 x_i^T x_j)^3$. For each kernel, we test $C = 1$ and $C = 1000$. Usually $C = 1$ is a good initial guess. As it is difficult to find out the optimal C, a procedure is to try different Cs and compare error rates obtained by cross validation. In Saunders et al. (1998), they point out that plotting a graph of error rate on different Cs will typically give a bowl shape, where the best value of C is somewhere in the middle. Therefore, we think it may be necessary to solve problems with large Cs so $C = 1000$ is tested here. In addition, the default C of SVM^{light} is 1000.

Numerical results using $q = 10$ are presented in Tables 5 to 8. The column "SV(BSV)" represents the number of support vectors and bounded support vectors. The column "Mis." is the number of misclassified training data while the "Obj." and "Iter." columns are objective values and the number of iterations, respectively. Note that here we present the objective value of the dual (that is, (1.2) and (1.5)). We also present the computational time (in *seconds*) in the last column. SVM^{light} implements a technique called "shrinking" which drops out some variables at the upper bound during the iterative process. Therefore, it can work on a smaller problem in most iterations. Right now we have not implemented similar techniques in BSVM so in Tables 5–8 we present results by SVM^{light} *without* using this shrinking technique. Except this option, we use all default options of SVM^{light}. Note that here we do not use the default optimizer of SVM^{light} (version 3.2) for solving (1.2). Following the suggestion by Joachims (2000), we link SVM^{light} with LOQO (Vanderbei, 1994) to achieve better stability. To give an idea of effects on using shrinking, in Table 9 we present results of SVM^{light} using this technique. It can be seen that shrinking is a very useful technique for large problems. How to effectively incorporate shrinking in BSVM is an issue for future investigation.

From Tables 5 to 8, we can see that results obtained by BSVM, no matter number of support vectors, number of misclassified data, and objective values, are very similar to those by SVM^{light}. This suggests that using BSVM, a formula with an additional term $b^2/2$ in the objective function, does not affect the training results much. Another interesting property is that the objective value of BSVM is always smaller than that of SVM^{light}. This is due to the properties that $y^T\alpha = 0$ in (1.2) and the feasible region of (1.2) is a subset of that of (1.5). To further check the effectiveness of using (1.5), in Tables 10 and 11, we present error rates by 10-fold cross validation or classifying original test data. Among all problems, test data of dna, satimage, letter, and shuttle are available so we present error rates by classifying them. Results suggest that for these problems, using (1.5) produces a classifier which is as good as that of using (1.2). In addition, we can see that the best rate may happen at different values of C.

When $C = 1$, BSVM and SVM^{light} take about the same number of iterations. However, it can be clearly seen that when $C = 1000$, both decomposition methods take many more

Table 10: SVM^{light}: Accuracy rate by 10-fold cross validation or classifying test data

C	1	5	10	50	100	500	1000	5000	10000
australian	85.36	86.23	85.65	85.51	85.94	84.06	82.17	80.29	77.97
diabetes	76.80	77.19	76.67	76.53	76.01	74.97	74.32	73.02	72.63
fourclass	80.29	80.99	82.73	94.90	97.33	99.42	99.77	99.77	99.88
german	75.40	75.00	75.90	72.70	69.70	69.00	68.80	69.10	68.70
heart	81.85	80.74	80.37	78.89	77.41	75.56	75.18	74.07	74.07
segment	99.65	99.70	99.70	99.74	99.78	99.78	99.74	99.70	99.83
vehicle	74.95	78.97	79.56	81.67	82.39	85.35	85.23	86.17	85.34
adult1	83.11	82.99	82.56	81.43	81.43	78.94	78.82	78.82	78.82
adult4	83.75	83.29	82.72	80.15	78.96	77.83	77.52	77.33	77.35
w1a	97.50	97.94	98.10	97.90	97.70	97.30	97.30	97.30	97.30
w4a	97.92	98.48	98.49	98.48	98.51	98.32	98.36	98.32	98.32
w7a	98.48	98.78	98.80	98.74	98.64	98.44	98.46	98.46	98.46
dna	96.12	97.22	97.72	97.89	97.89	97.89	97.89	97.89	97.89
satimage	98.85	99.00	99.10	99.10	99.20	99.05	99.05	98.75	98.65
letter	99.32	99.48	99.50	99.54	99.60	99.80	99.84	99.86	99.82
shuttle	97.78	98.45	98.99	99.59	99.62	99.77	99.64	99.80	99.81

Table 11: BSVM: Accuracy rate by 10-fold cross validation or classifying test data

C	1	5	10	50	100	500	1000	5000	10000
australian	85.36	86.23	85.65	85.51	85.94	84.06	82.17	80.15	77.97
diabetes	76.93	77.19	76.93	76.53	76.01	74.97	74.32	73.02	72.63
fourclass	80.29	81.22	82.38	94.32	97.33	99.42	99.77	99.77	99.88
german	75.30	75.00	75.90	72.70	69.50	69.00	68.80	69.10	68.70
heart	82.22	80.74	80.00	78.52	77.41	75.93	75.18	74.07	74.07
segment	99.65	99.70	99.70	99.74	99.78	99.78	99.74	99.74	99.83
vehicle	74.95	78.97	79.68	81.79	82.50	85.23	85.23	86.41	85.34
adult1	83.18	82.99	82.62	81.43	81.43	78.94	78.82	78.82	78.82
adult4	83.75	83.29	82.72	80.17	78.96	77.83	77.52	77.30	77.30
w1a	97.50	97.94	98.10	97.90	97.70	97.30	97.30	97.30	97.30
w4a	97.92	98.48	98.49	98.48	98.51	98.32	98.32	98.32	98.32
w7a	98.48	98.78	98.80	98.74	98.64	98.44	98.46	98.46	98.46
dna	96.12	97.30	97.64	97.81	97.81	97.81	97.81	97.81	97.81
satimage	98.85	99.00	99.10	99.10	99.20	99.05	99.05	98.75	98.65
letter	99.32	99.48	99.50	99.56	99.60	99.80	99.84	99.86	99.82
shuttle	97.62	98.21	98.99	99.59	99.62	99.77	99.64	99.80	99.81

iterations. For most problems we have tested, not only those presented here, we observe slow convergence of decomposition methods when C is large. There are several possible reasons which cause this difficulty. We think that one of them is that when C is increased, the number of free variables in the iterative process is increased. In addition, the number of free variables at the final solution is also increased. Though both the numbers of support and bounded support vectors are decreased when C is increased, in many cases, bounded variables when $C = 1$ become free variables when $C = 1000$. When C is increased, the

Table 12: Iterations and q: BSVM, $C = 1000$

	$q = 10$	$q = 20$	$q = 30$	$q = 40$
australian	12609	2840	1398	688
diabetes	21908	3716	1044	531
german	16874	6511	3558	2120
heart	1230	356	186	89
vehicle	10238	2186	748	309

Table 13: Iterations and q: SVM^{light}, $C = 1000$

	$q = 10$	$q = 20$	$q = 30$	$q = 40$
australian	101103	71121	62120	34491
diabetes	75506	43160	37158	41241
german	43411	31143	28084	24529
heart	5772	3321	3068	1982
vehicle	177188	113357	107344	90370

separating hyperplane tries to to fit as many training data as possible. Hence more points (i.e. more free α_i) tend to be at two planes $w^T\phi(x) + b = \pm 1$. Since the decomposition method has more difficulties on handling free variables, if the problem is ill-conditioned, more iterations are required. As our selection of the working set always try to push free variables to be bounded variables, the number of free variables is kept small. Therefore, the convergence seems faster. It can be clearly seen that for almost all cases in Tables 6 and 8, BSVM takes fewer iterations than SVM^{light}.

Problem fourclass in Table 6 is the best example to show the characteristic of BSVM. For this problem, at the final solution, the number of free variables is small. In the iterative process of decomposition methods, many free variables of iterates are in fact bounded variables at the final solution. BSVM considers free variables in subsequent iterations so all bounded variables are quickly identified. The number of free variables is kept small so the slow local convergence does not happen. However, SVM^{light} goes through an iterative process with more free variables so it takes a lot more iterations. We use Figure 3 to illustrate this observation in more detail. It can be seen in Figure 3(a) that the number of free variables in SVM^{light} is increased to about 70 in the beginning. Then it is difficult do identify whether they should be at the bounds or not. Especially in final iterations, putting a free variable on a bound can take thousands of iterations. On the other hand, the number of free variables of BSVM is always small (less than 50).

We also note that sometimes many free variables in the iterative process are still free in the final solution. Hence BSVM may pay too much attention on them or wrongly put them as bounded variables. Therefore, some iterations are wasted so the gap between BSVM and SVM^{light} is smaller. An example of this is adult problems.

Next we study the relation between number of iterations and q, the size of the working set. Using the RBF kernel and $C = 1000$, results are in Tables 12 and 13. We find out that by using BSVM, number of iterations is dramatically decreased as q becomes larger. On the other hand, using SVM^{light}, the number of iterations does not decrease much. Since optimization solvers costs a certain amount of computational time in each iteration, this result shows that SVM^{light} is only suitable for using small q. On the other hand, Algorithm 2.1 provides the potential of using different q in different situations.

4 Using Algorithm 2.1 for Standard SVM Formulation

We can modify Algorithm 2.1 for standard SVM formulation (1.2):

Let r be the number of free variables at α_k
Let $q = q_1 + q_2$, where $q_1 \leq r$ and q_2 is an even number
For those $0 < (\alpha_k)_i < C$, select q_1 elements with the smallest $|\nabla f(\alpha_k)_i + by_i|$
Select q_2 elements from the rest of the index set by SVM^{light}'s working set selection.

Algorithm 4.1: Selection of the working set

Here we describe Algorithm 4.1 in a more general way. If under a similar setting of Algorithm 2.1, we choose $q_1 \approx q_2 \approx q/2$. Now if $(\alpha_k)_i$ is a free variable and i is in the previous working set, we have

$$\nabla f(\alpha_k)_i + by_i = 0.$$

This is different from the bounded case where we have $\nabla f(\alpha_k)_i = 0$. Therefore, unlike Algorithm 2.1 where $q/2$ free elements with the smallest $|\nabla f(\alpha_k)_i|$ are chosen, here we select the smallest $|\nabla f(\alpha_k)_i + by_i|$. Note that b is not a constant and must be recalculated in each iteration.

As in previous sections we focus on using Algorithm 2.1 for bounded formulation (1.5), we wonder whether the concept of reducing the number of free variables could also work for the regular formulation (1.2). Note that the experience in Section 2 tells us that without experiments and analysis it is difficult to judge whether a working set selection is useful for one formulation or not. Thus in this section we will conduct some preliminary experiments.

We implement Algorithm 4.1 with $q = 10, q_1 = 6$, and $q_2 = 4$. All other settings such as stopping criteria are the same as in the previous section. Results using MATLAB for small problems are in Tables 14 and 15. The RBF kernel is used so we compare them with

Tables 5 and 6. It can be clearly seen that the number of iterations is smaller than that of SVM^{light} and is competitive with BSVM.

This experiment confirms the importance of choosing free variables into the working set. A full implementation using Algorithm 4.1 for (1.2) will be an important research topic. A good linear-constrained optimization software must be chosen in order to solve (1.3).

Table 14: BSVM: RBF kernel and $C = 1$

Problem	SV(BSV)	Mis.	Obj.	Iter.
australian	244(190)	98	-201.64	211
diabetes	447(435)	168	-413.56	102
german	600(512)	189	-502.77	330
heart	132(107)	36	-100.88	63
vehicle	438(415)	210	-413.02	131
fourclass	408(401)	167	-383.58	89

Table 15: BSVM: RBF kernel and $C = 1000$

Problem	SV(BSV)	Mis.	Obj.	Iter.
australian	222(55)	28	-81198.78	17302
diabetes	377(274)	128	-302470.21	24299
german	509(15)	3	-39909.12	26128
heart	101(1)	0	-4815.73	1742
vehicle	284(172)	47	-179597.77	13191
fourclass	89 (74)	2	-53886.61	383

5 Discussions and Conclusions

From an optimization point of view, decomposition methods are like "coordinate search" methods or "alternating variables method" (Fletcher, 1987, Chapter 2.2). They have slow convergences as the first and second order information is not used. In addition, if the working set selection is not appropriate, though the strict decrease of the objective value holds, the algorithm may not converge (see, for example, Powell (1973)). However, even with such disadvantages, the decomposition method has become one of the major methods for SVM. We think the main reason is that the decomposition method is efficient for SVM in the following situations:

1. C is small and most support vectors are at the upper bound. That is, there are not many free variables in the iterative process.

2. The problem is well-conditioned even though there are many free variables.

For example, we do not think that adult problems with $C = 1000$ belong to the above cases. They are difficult problems for decomposition methods.

If for most applications we only need solutions of problems which belong to the above situations, current decomposition methods may be good enough. Especially a SMO type (Platt, 1998) algorithm has the advantage of not requiring any optimization solver. However, if in many cases we need to solve difficult problems (for example, C is large), more optimization knowledge and techniques should be considered. We hope that practical applications will provide a better understanding on this issue.

Regarding the SVM formulation, we think (1.5) is simpler than (1.1) but with similar quality for our test problems. In addition, in this paper we experiment with different implementation of the working set selection. The cost is always the same: at most $O(ql)$ by selecting some of the largest and smallest $\nabla f(\alpha_k)$. This may not be the case for regular SVM formulation (1.1) due to the linear constraint $y^T\alpha = 0$. In SVM^{light}, the implementation is simple because (2.1) is very special. If we change constraints of (2.1) to $0 \leq \alpha_k + d \leq C$, the solution procedure may be more complicated. Currently we add $b^2/2$ into the objective function. This is the same as finding a hyperplane passing through the origin for separating data $[\phi(x_i), 1], i = 1, \dots, l$. It was pointed out by Cristianini and Shawe-Taylor (2000) that the number 1 added may not be the best choice. Experimenting with different numbers can be a future issue for improving the performance of BSVM.

From a numerical point of view, there are also possible advantages of using (1.5). When solving (1.2), a numerical difficulty is on deciding whether a variable is at the bound or not because it is generally not recommended to compare a floating-point number with another one. For example, to calculate b of (1.2), we use the following KKT condition

$$y_i(Q\alpha)_i + b - 1 = 0 \text{ if } 0 < \alpha_i < C. \tag{5.1}$$

Therefore, we can calculate b by (5.1) where i is any element in B. However, when implementing (5.1), we cannot directly compare α_i to 0 or C. In SVM^{light}, a small $\epsilon_a = 10^{-12} > 0$ is introduced. They consider α_i to be free if $\alpha_i \geq \epsilon_a$ and $\alpha_i \leq C - \epsilon_a$. Otherwise, if a wrong α_i is considered, the obtained b can be erroneous. On the other hand, if the bounded formulation is used and appropriate solvers for sub-problem (2.7) are used, it is possible to directly compare α_i with 0 or C without needing an ϵ_a. For example, in Lin and Moré (1999), they used a method called "project gradient" and in their implementation all values at bounds are done by direct assignments. Hence it is safe to compare α_i with 0 or C. To be more precise, for floating-point computation, if $\alpha_i \leftarrow C$ is assigned somewhere, a future floating-point comparison between C and C returns true as they both have the same internal representation.

In (Platt, 1998), a situation was considered where the bias b of the standard SVM formulation (1.2) is a constant instead of a variable. Then the dual problem is a bound-constrained formulation so BSVM can be easily modified for it.

In Section 2, we demonstrate that finding a good working set is not an easy task. Sometimes a natural method turns out to be a bad choice. It is also interesting to note that for different formulations (1.2) and (1.5), similar selection strategies (2.1) and (2.2) give totally different performance. On the other hand, both Algorithms 2.1 and 4.1 are very useful for (1.2) and (1.5), respectively. Therefore, for any new SVM formulations, we should be careful when applying existing selections to them.

Finally we summarize some possible advantages of Algorithm 2.1:

1. It keeps the number of free variables as low as possible. This in general leads to faster convergences for difficult problems.

2. It tends to consider free variables in the current iteration again in subsequent iterations. Therefore, corresponding columns of these elements are naturally cached.

Acknowledgments

This work was supported in part by the National Science Council of Taiwan via the grant NSC 89-2213-E-002-013. The authors thank Chih-Chung Chang for many helpful discussions and comments. Part of the software implementation benefited from his help. They also thank Thorsten Joachims, Pavel Laskov, John Platt, and anonymous referees for helpful comments.

References

Chang, C.-C., Hsu, C.-W., & Lin, C.-J. (2000). The analysis of decomposition methods for support vector machines. *IEEE Trans. Neural Networks*, *11*(4), 1003-1008.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines.* Cambridge, UK: Cambridge University Press.

Fletcher, R. (1987). *Practical methods of optimization.* John Wiley and Sons.

Friess, T.-T., Cristianini, N., & Campbell, C. (1998). The kernel adatron algorithm: a fast and simple learning procedure for support vector machines. In *Proceeding of 15th Intl. Conf. Machine Learning.* Morgan Kaufman Publishers.

Ho, T. K., & Kleinberg, E. M. (1996). Building projectable classifiers of arbitrary complexity. In *Proceedings of the 13th International Conference on Pattern Recognition* (p. 880-885). Vienna, Austria.

Joachims, T. (1998). Making large-scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, & A. J. Smola (Eds.), *Advances in kernel methods - support vector learning.* Cambridge, MA: MIT Press.

Joachims, T. (2000). (Private communication)

Laskov, P. (1999). An improved decomposition algorithm for regression support vector machines. In *Workshop on Support Vector Machines, NIPS99.*

Lin, C.-J. (2000). *On the convergence of the decomposition method for support vector machines* (Tech. Rep.). Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan.

Lin, C.-J., & Moré, J. J. (1999). Newton's method for large-scale bound constrained problems. *SIAM J. Optim.*, *9*, 1100-1127. (Software available at `http://www.mcs.anl.gov/~more/tron`)

Mangasarian, O. L., & Musicant, D. R. (1999). Successive overrelaxation for support vector machines. *IEEE Trans. Neural Networks*, *10*(5), 1032-1037.

Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). *Machine learning, neural and statistical classification.* Englewood Cliffs, N.J.: Prentice Hall. (Data available at anonymous ftp: `ftp.ncc.up.pt/pub/statlog/`)

Murphy, P. M., & Aha, D. W. (1994). *UCI repository of machine learning databases.* Irvine, CA. (Available at `http://www.ics.uci.edu/~mlearn/MLRepository.html`)

Osuna, E., Freund, R., & Girosi, F. (1997). Training support vector machines: An application to face detection. In *Proceedings of CVPR'97.*

Platt, J. C. (1998). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, & A. J. Smola (Eds.), *Advances in kernel methods - support vector learning.* Cambridge, MA: MIT Press.

Powell, M. J. D. (1973). On search directions for minimization. *Math. Programming*, *4*, 193-201.

Saunders, C., Stitson, M. O., Weston, J., Bottou, L., Schölkopf, B., & Smola, A. (1998). *Support vector machine reference manual* (Tech. Rep. No. CSD-TR-98-03). Egham, UK: Royal Holloway, University of London.

Schölkopf, B., Burges, C. J. C., & Smola, A. J. (Eds.). (1998). *Advances in kernel methods - support vector learning.* Cambridge, MA: MIT Press.

Vanderbei, R. (1994). *LOQO: An interior point code for quadratic programming* (Tech. Rep. No. SOR 94-15). Statistics and Operations Research, Princeton University. (revised November, 1998)

Vapnik, V. (1995). *The nature of statistical learning theory.* New York, NY: Springer-Verlag.

Vapnik, V. (1998). *Statistical learning theory.* New York, NY: John Wiley.