*LETTER* Communicated by John Platt

# Decomposition Methods for Linear Support Vector Machines

**Wei-Chun Kao**
*b89106@csie.ntu.edu.tw*
**Kai-Min Chung**
*b88061@csie.ntu.edu.tw*
**Chia-Liang Sun**
*b88047@csie.ntu.edu.tw*
**Chih-Jen Lin**
*cjlin@csie.ntu.edu.tw*
*Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan*

**In this letter, we show that decomposition methods with alpha seeding are extremely useful for solving a sequence of linear support vector machines (SVMs) with more data than attributes. This strategy is motivated by Keerthi and Lin (2003), who proved that for an SVM with data not linearly separable, after $C$ is large enough, the dual solutions have the same free and bounded components. We explain why a direct use of decomposition methods for linear SVMs is sometimes very slow and then analyze why alpha seeding is much more effective for linear than nonlinear SVMs. We also conduct comparisons with other methods that are efficient for linear SVMs and demonstrate the effectiveness of alpha seeding techniques in model selection.**

## 1 Introduction

Solving linear and nonlinear support vector machines (SVM) has been considered two different tasks. For linear SVM without too many attributes in data instances, people have been able to train millions of data (e.g. Mangasarian & Musicant, 2000), but for other types of problems, in particular, nonlinear SVMs, the requirement of huge memory as well as computational time has prohibited us from solving very large problems. Currently, the decomposition method, a specially designed optimization procedure, is one of the main tools for nonlinear SVMs. In this letter, we show the drawbacks of existing decomposition methods, in particular, sequential minimal optimization (SMO)–type algorithms, for linear SVMs. To remedy these drawbacks, using theorem 3 of Keerthi and Lin (2003), we develop effective strategies so that decomposition methods become efficient for solving linear SVMs.

 

First, we briefly describe linear and nonlinear SVMs. 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 standard SVM formulation (Cortes & Vapnik, 1995) is as follows:

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

If $\phi(x) = x$, usually we say equation 1.1 is the form of a linear SVM. On the other hand, if $\phi$ maps $x$ to a higher-dimensional space, equation 1.1 is a nonlinear SVM.

For a nonlinear SVM, the number of variables depends on the size of $w$ and can be very large (even infinite), so people solve the following dual form:

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

where $Q$ is an $l \times l$ positive semidefinite matrix with $Q_{ij} = y_i y_j \phi(x_i)^T \phi(x_j)$, $e$ is the vector of all ones, and $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel function. Equation 1.2 is solvable because its number of variables is the size of the training set, independent of the dimensionality of $\phi(x)$.

The primal and dual relation shows

$$w = \sum_{i=1}^{l} \alpha_i y_i \phi(x_i), \tag{1.3}$$

so

$$\operatorname{sgn}(w^T \phi(x) + b) = \operatorname{sgn}\left( \sum_{i=1}^{l} \alpha_i y_i K(x_i, x) + b \right)$$

is the decision function.

Unfortunately, for a large training set, $Q$ becomes such a huge, dense matrix that traditional optimization methods cannot be directly applied. Currently, some specially designed approaches, such as decomposition methods (Osuna, Freund, & Girosi, 1997; Joachims, 1998; Platt, 1998) and finding the nearest points of two convex hulls (Keerthi, Shevade, Bhattacharyya, & Murthy, 2000), are major ways of solving equation 1.2.

On the other hand, for linear SVMs, if $n \ll l$, $w$ is not a huge vector variable, so equation 1.1 can be solved by many regular optimization methods. As at the optimal solution $\xi_i = \max(0, 1 - y_i(w^T x_i + b))$, in a sense we mainly have to find out $w$ and $b$. Therefore, if the number of attributes $n$ is small, there are not many main variables $w$ and $b$ in equation 1.1, no matter how large the training set is. Currently, people have been able to train a linear SVM with millions of data (e.g. Mangasarian & Musicant, 2000); but for a nonlinear SVM with many fewer data, we need more computational time as well as more computer memory.

Therefore, it is natural to ask whether in SVM software, linear and nonlinear SVMs should be treated differently and solved by two methods. It is also interesting to see how capable nonlinear SVM methods (e.g., decomposition methods) are for linear SVMs. By linear SVMs, we mean those with $n < l$. If $n \geq l$, the dual form, equation 1.2, has fewer variables than $w$ of the primal, a situation similar to nonlinear SVMs. As the rank of $Q$ is less than or (usually) equal to $\min(n, l)$, the linear SVMs we are interested in here are those with low-ranked $Q$.

Recently, in many situations, linear and nonlinear SVMs have been considered together. Some approaches (Lee & Mangasarian, 2001; Fine & Scheinberg, 2001) approximate nonlinear SVMs by different problems, which are in the form of linear SVMs (Lin & Lin, in press; Lin, 2002) with $n \ll l$. In addition, for nonlinear SVM model selection with gaussian kernel, Keerthi and Lin (2003) proposed an efficient method, which has to conduct linear SVMs model selection first (i.e., linear SVMs with different $C$). Therefore, it is important to discuss optimization methods for linear and nonlinear SVMs at the same time.

In this letter, we focus on decomposition methods. In section 2, we show that existing decomposition methods are inefficient for training linear SVMs. Section 3 demonstrates theoretically and experimentally that the alpha seeding technique is particularly useful for linear SVMs. Some implementation issues are discussed in section 4. The decomposition method with alpha seeding is compared with existing linear SVM methods in section 5. In section 6, we apply the new implementation to solve a sequence of linear SVMs required for the model selection method in Keerthi and Lin (2003). The discussion and concluding remarks are in section 7.

## 2 Drawbacks of Decomposition Methods for Linear SVMs with $n \ll l$

The decomposition method is an iterative procedure. In each iteration, the index set of variables is separated into two sets $B$ and $N$, where $B$ is the working set. Then in that iteration, variables corresponding to $N$ are fixed while a subproblem on variables corresponding to $B$ is minimized. If $q$ is the size of the working set $B$, in each iteration, only $q$ columns of the Hessian matrix $Q$ are required. They can be calculated and stored in computer memory when needed. Thus, unlike regular optimization methods, which usually require

access of the whole $Q$, here, the memory problem is solved. Clearly, decomposition methods are specially designed for nonlinear SVMs. Throughout this article, we use *DSVM* to refer to the solver of SVM that adopts the decomposition method, such as LIBSVM (Chang & Lin, 2001b) and $SVM^{light}$ (Joachims, 1998). When the size of its working set is two, we say it is of the SMO type (Platt, 1998).

Unlike popular optimization methods such as Newton or quasi-Newton, which enjoy fast convergence, decomposition methods converge slowly, as in each iteration only very few variables are updated. We will show that the situation is even worse when solving linear SVMs.

It has been demonstrated (Hsu & Lin, 2002b) by experiments that if $C$ is large and the Hessian matrix $Q$ is not well conditioned, decomposition methods converge very slowly. For linear SVMs, if $n \ll l$, then $Q$ is a low-rank and hence ill-conditioned matrix. In Figure 1, we demonstrate a simple example by using the problem heart from the statlog database (Michie, Spiegelhalter, & Taylor, 1994). Each attribute is scaled to $[-1, 1]$. We use LIBSVM (Chang & Lin, 2001b) to solve linear and nonlinear (RBF kernel, $e^{-\|x_i - x_j\|^2/(2\sigma^2)}$ with $1/(2\sigma^2) = 1/n$) SVMs with $C = 2^{-8}, 2^{-7.5}, \ldots, 2^8$ and present the number of iterations. Though two different problems are solved (in particular, their $Q_{ij}$'s are in different ranges), Figure 1 clearly indicates the huge number of iterations for solving the linear SVMs. Note that for linear SVMs, the slope is greater than that for nonlinear SVMs and is very close to one, especially when $C$ is large. This means that a doubled C leads to a doubled number of iterations.

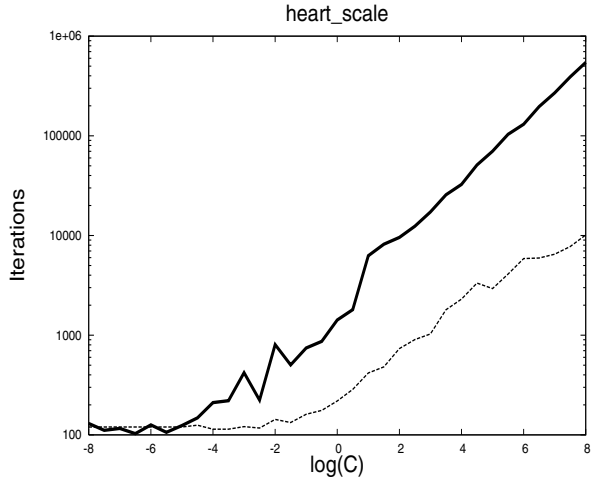


Figure 1: Number of decomposition iterations for solving SVMs with linear (the thick line) and RBF (the thin line) kernel.

The following theorems that hold only for linear SVMs help us to realize the difficulty the decomposition methods suffer from. Theorem 2 can further explain why the number of iterations is nearly doubled when $C$ is doubled.

**Theorem 1** (Keerthi & Lin, 2003). *The dual linear SVM has the following properties:*

- *There is $C^*$ such that for all $C \geq C^*$, there are optimal solutions at the same face.*
- *In addition, for all $C \geq C^*$, the primal solution $w$ is the same.*

By the "face" of $\alpha$, we mean three types of value of each $\alpha_i$: (1) lower-bounded, that is, $\alpha_i = 0$, (2) upper-bounded, that is, $\alpha_i = C$, and (3) free, that is, $0 < \alpha_i < C$. More precisely, the face of $\alpha$ can be represented by a length $l$ vector whose components are in {lower-bounded, upper-bounded, free}.

This theorem indicates that after $C \geq C^*$, exponentially increased numbers of iterations are wasted in order to obtain the same primal solution $w$. Even if we could detect $C^*$ and stop training SVMs, for $C$ not far below $C^*$, the number of iterations may already be huge. Therefore, it is important to have an efficient linear SVM solver that could handle both large and small $C$.

Next, we try to explain the nearly doubled iterations by the difficulty of locating faces of the dual solution $\alpha$.

**Theorem 2.** *Assume that any two parallel hyperplanes in the feature space do not contain more than $n + 1$ points of $\{x_i\}$ on them. We have:*[1]

1. *For any optimal solution of equation 1.2, it has no more than $n + 1$ free components.*
2. *There is $C^*$ such that after $C \geq C^*$, all optimal solutions of equation 1.2 share at least the same $l - n - 1$ bounded $\alpha$ variables.*

The proof is available in Kao, Chung, Sun, and Lin (2002). This result indicates that when $n \ll l$, most components of optimal solutions are at bounds. Furthermore, dual solutions at $C$ and $2C$ share at least the same $l - 2(n - 1)$ upper- and lower-bounded components. If upper-bounded $\alpha_i$ at $C$ remains upper-bounded at $2C$, a direct use of decomposition methods means that $\alpha_i$ is updated from 0 to $C$ and from 0 to $2C$, respectively. Thus, we anticipate that the efforts are roughly doubled. We confirm this explanation

[1] Note that a pair of parallel hyperplanes is decided by $n + 1$ numbers (the $n$ number decides one hyperplane in the feature space $R^n$, and another one decides the other hyperplane parallel to it). So the assumption of theorem 2 would be violated if $m$ linear equations in $n + 1$ variables, where $m > n + 1$, have solutions. The occurrence of this scenario is of measure zero. This explains that the assumption of theorem 2 is generic.

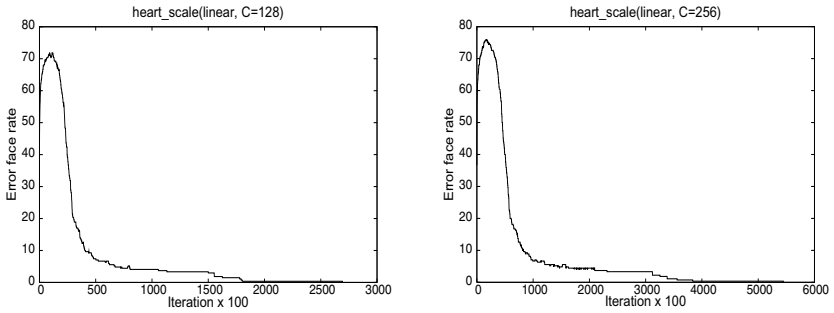


Figure 2: The error face rate (i.e., the difference between the current face and the one at the final solution) for solving linear SVM with $C = 128$ and $C = 256$.

by comparing the error face rate (i.e., the difference between the current face and the one at the final solution) with $C = 2^7$ and $C = 2^8$. As shown in Figure 2, two curves are quite similar except that the scale of $x$-axis differs by twice. This indicates that $\alpha$ travels similar faces for $C = 2^7$ and $C = 2^8$, and the number of iterations spent on each face with $C = 2^8$ is roughly doubled.

## 3 Alpha Seeding for Linear SVMs

Theorem 2 implies that for linear SVMs, dual solutions may share many upper- and lower-bounded variables. Therefore, we conjecture that if $\alpha^1$ is an optimal solution at $C = C_1$, then $\alpha^1 C_2/C_1$ can be a very good initial point for solving equation 1.2 with $C = C_2$. The reason is that $\alpha^1 C_2/C_1$ is at the same face as $\alpha^1$, and it is likely to be at a similar face of one optimal solution of $C = C_2$. This technique, called alpha seeding, was originally proposed for SVM model selection (DeCoste & Wagstaff, 2000) where several problems (see equation 1.2) with different $C$ have to be solved. Earlier work that focuses on nonlinear SVMs mainly uses alpha seeding as a heuristic. For linear SVMs, the speed could be significantly boosted due to the above analysis.

The following theorem further supports the use of alpha seeding:

**Theorem 3.** *There are two vectors $A$, $B$, and a number $C^*$ such that for any $C \geq C^*$, $AC + B$ is an optimal solution of equation 1.2.*

The proof is in Kao et al. (2002). If $A_i > 1$, $A_i C + B > C$ after $C$ is large enough, and this violates the bounded constraints in equation 1.2. Similarly, $A_i$ cannot be less than zero, so $0 \leq A_i \leq 1$. Therefore, we can consider the following three situations of vectors $A$ and $B$:

1. $0 < A_i \leq 1$

Table 1: Comparison of Iterations (Linear Kernel), With and Without Alpha Seeding.

| | $\alpha$-Seeding | | | Without $\alpha$-Seeding | | |
|---|---|---|---|---|---|---|
| Problem | Number of Iterations | $C^*$ | $w^T w$ | Total Iterations | Number of Iterations ($C = 2^{7.5}$) | Number of Iterations ($C = 2^8$) |
| heart | 27,231 | $2^{3.5}$ | 5.712 | 2,449,067 | 507,122 | 737,734 |
| australian | 79,162 | $2^{2.5}$ | 2.071 | 20,353,966 | 3,981,265 | 5,469,092 |
| diabetes | 33,264 | $2^{6.5}$ | 16.69 | 1,217,926 | 274,155 | 279,062 |
| german | 277,932 | $2^{10}$ | 3.783 | 42,673,649 | 6,778,373 | 14,641,135 |
| web | 24,044,242 | Unstable | Unstable | $\geq 10^8$ | 74,717,242 | $\geq 10^8$ |
| adult | 3,212,093 | Unstable | Unstable | $\geq 10^8$ | 56214289 | 84,111,627 |
| ijcnn | 590,645 | $2^6$ | 108.6 | 41,440,735 | 8,860,930 | 13,927,522 |

2. $A_i = 0, B_i = 0$
3. $A_i = 0, B_i > 0$

For the second case, $\alpha_i^1 \frac{C_2}{C_1} = A_i C_2 + B_i = 0$, and for the first case, $A_i C \gg B_i$ after $C$ is large enough. Therefore, $\alpha_i^1 \frac{C_2}{C_1} = A_i C_2 + B_i \frac{C_2}{C_1} \approx A_i C_2 + B_i$. For both cases, alpha seeding is very useful. On the other hand, using theorem 2, there are few ($\leq n + 1$) components satisfying the third case.

Next, we conduct some comparisons between DSVM with and without alpha seeding. Here, we consider two-class problems only. Some statistics of the data sets used are in Tables 1 and 2. The four small problems are from the statlog collection (Michie et al., 1994). The problem adult is compiled by Platt (1998) from the UCI "adult" data set (Blake & Merz, 1998). Problem web is also from Platt. Problem ijcnn is from the first problem of IJCNN challenge 2001 (Prokhorov, 2001). Note that we use the winner's transformation of the raw data (Chang & Lin, 2001a).

We train linear SVMs with $C \in \{2^{-8}, 2^{-7.5}, \ldots, 2^8\}$. That is, $[2^{-8}, 2^8]$ is discretized to 33 points with equal ratio. Table 1 presents the total number

Table 2: Comparison of Iterations (RBF Kernel), With and Without Alpha Seeding.

| Problem | $l$ | $n$ | $\alpha$-Seeding | Without $\alpha$-Seeding |
|---|---|---|---|---|
| heart | 270 | 13 | 43,663 | 56,792 |
| australian | 690 | 14 | 230,983 | 323,288 |
| diabetes | 768 | 8 | 101,378 | 190,047 |
| german | 1000 | 24 | 191,509 | 260,774 |
| web | 49,749 | 300 | 633,788 | 883,319 |
| adult | 32,561 | 123 | 2,380,265 | 4,110,663 |
| ijcnn | 49,990 | 22 | 891,563 | 1,968,396 |

of iterations of training 33 linear SVMs using the alpha seeding approach. We also individually solve them by LIBSVM and list the number of iterations (total, $C = 2^{7.5}$, and $C = 2^8$). The alpha seeding implementation will be described in detail in section 4. We also list the approximate $C^*$ for which linear SVMs with $C \geq C^*$ have the same decision function. In addition, the constant $w^T w$ after $C \geq C^*$ is also given. For some problems (e.g., web and adult), $w^T w$ has not reached a constant until $C$ is very large, so we indicate them as "unstable" in Table 1.

To demonstrate that alpha seeding is much more effective for linear than nonlinear SVMs, Table 2 presents the number of iterations using the radial basis function (RBF) kernel $K(x_i, x_j) = e^{-\|x_i - x_j\|^2/(2\sigma^2)}$ with $1/2\sigma^2 = 1/n$. It is clear that the number of iterations saved by using alpha seeding is marginal. In addition, comparing to the "Total Iterations" column in Table 1, we confirm again the slow convergence for linear SVMs without alpha seeding.

The alpha seeding approach performs so well that its total number of iterations is much fewer than solving one single linear SVM with the original decomposition implementation. Therefore, if we intend to solve one linear SVM with a particular $C$, it may be more efficient to solve one with small initial $C_0$ and then use the proposed alpha seeding method by gradually increasing $C$.

Furthermore, since we have solved linear SVMs with different $C$, model selection by cross validation is already done. From the discussion in section 2, solving several linear SVMs without alpha seeding is time-consuming and model selection is not easy.

Note that in Table 1, web is the most difficult problem and requires the largest number of iterations. Theorem 2 helps to explain this: since web's large number of attributes might lead to more free variables during iterations or at the final solution, alpha seeding is less effective.

## 4 Implementation

Though the concept is so simple, an efficient and elegant implementation requires considerable thought.

Most DSVM implementations maintain the gradient vector of the dual objective function during iterations. The gradient is used for selecting the working set or checking the stopping condition. In the nonlinear SVM, calculation of the gradient $Q\alpha - e$ requires $O(l^2 n)$ operations ($O(n)$ for each kernel evaluation), which are expensive. Therefore, many forms of DSVM software use $\alpha = 0$ as the initial solution, which makes the initial gradient $-e$ immediately available. However, in DSVM with alpha seeding, the initial solution is obtained from the last problem, so the initial gradient is not a constant vector. Fortunately, for linear SVMs, the situation is not as bad as that in the nonlinear SVM. In this case, the kernel matrix is of the form $Q = X^T X$, where $X = [y_1 x_1, \ldots, y_l x_l]$ is an $n$ by $l$ matrix. We can calculate

the gradient by $Q\alpha - e = X^T(X\alpha) - e$, which requires only $O(ln)$ operations. The first decomposition software that used this for linear SVMs is $SVM^{light}$ (Joachims, 1998).

Similarly, if $\alpha$ is changed by $\Delta\alpha$ between two consecutive iterations, then the change of gradient is $Q(\Delta\alpha) = X^T(X\Delta\alpha)$. Since there are only $q$ nonzero elements in $\Delta\alpha$, the gradient can be updated with $O(nq) + O(ln) = O(ln)$ operations, where $q$ is the size of working set. Note that because $l \gg q$, increasing $q$ from 2 to some other small constant will not affect the time of the updating gradient. In contrast, for nonlinear SVMs, if $Q$ is not in the cache, the cost for updating the gradient is by computing $Q(\Delta\alpha)$, which requires $q$ columns of $Q$ and takes $O(lnq)$-time. Thus, while the implementation of nonlinear SVMs may choose SMO-type implementation (i.e., $q = 2$) for a lower cost per iteration, we should use a larger $q$ for linear SVMs because the gradient update is independent of $q$ and the number of total iterations may be reduced.

In the nonlinear SVM, constructing the kernel matrix $Q$ is expensive, so a cache for storing recently used elements of $Q$ is necessary. However, in linear SVM, either the kernel matrix $Q$ or the cache is no longer needed.

## 5 Comparison with Other Approaches

It is interesting to compare the proposed alpha seeding approach with efficient methods for linear SVMs. In this section, we consider active SVM (ASVM) (Mangasarian & Musicant, 2000) and Lagrangian SVM (LSVM) (Mangasarian & Musicant, 2001).

DSVM, ASVM, and LSVM solve slightly different formulations, so a fair comparison is difficult. However, our goal here is only to demonstrate that with alpha seeding, decomposition methods can be several times faster and competitive with other linear SVM methods. In the following, we briefly describe the three implementations.

DSVM is the standard SVM, which uses the dual formulation, equation 1.2. ASVM and LSVM both consider a square error term in the objective function:

$$\min_{w,b,\xi} \ \frac{1}{2}(w^T w + b^2) + C\sum_{i=1}^{l} \xi_i^2. \tag{5.1}$$

Then, the dual problem of equation 5.1 is

$$\begin{aligned} \min_{\alpha} \ & \frac{1}{2}\alpha^T \left(Q + yy^T + \frac{I}{2C}\right)\alpha - e^T\alpha \\ \text{subject to } & 0 \le \alpha_i, \quad i = 1, \ldots, l, \end{aligned} \tag{5.2}$$

where $I$ is the identity matrix. The solution of equation 5.2 has far more free components than that of equation 1.2 as upper-bounded variables of

Table 3: Comparison of Different Approaches for Linear SVMs.

| | Decomposition Methods (LIBSVM) | | | Methods for Linear SVMs | | |
|---|---|---|---|---|---|---|
| | | With $\alpha$ Seeding | Without $\alpha$ Seeding | | ASVM | LSVM |
| Problem | Acc. | Time: $C \le 2^8 (C \le 2^3)$ | Time: $C \le 2^3$ | Acc. | Time | Time |
| australian | 85.51 | 4.1 (3.6) | 7.0 | 88.70 | 8.1 | 2.3 |
| heart | 85.56 | 1.4 (0.9) | 1.0 | 85.56 | 5.8 | 1.8 |
| diabetes | 79.69 | 2.4 (2.3) | 1.6 | 81.64 | 6.2 | 2.0 |
| german | 73.65 | 10.2 (6.2) | 18.2 | 73.95 | 16.4 | 9.0 |
| ijcnn | 92.65 | 981.9 (746.0) | 3,708.6 | 92.51 | 725.2 | 17,496.4 |
| adult | 85.02 | 1065.9 (724.8) | 12,026.8 | 84.90 | 3,130.7 | 13,445.4 |
| web | 98.67 | 18,035.3 (1738.2) | 7,035.6 | 98.63 | 10,315.9 | 43,060.1 |

Notes: Acc.: test accuracy using the parameter obtained from cross validation. Time (in seconds): total training time of five-fold cross validation by trying $C = 2^{-10}, 2^{-9.5}, \ldots, 2^8$ (or $2^3$ if specified).

this equation are likely to be free now. With different formulations, their stopping conditions are not exactly the same. We use conditions from similar derivations; details are discussed in Kao et al. (2002).

In this experiment, we consider LIBSVM for DSVM (with and without alpha seeding). For ASVM, we use the authors' C++ implementation available on-line at http://www.cs.wisc.edu/dmi/asvm. The authors of LSVM provide only MATLAB programs, so we implement it by modifying LIBSVM. The experiments were done on an Intel Xeon 2.8 GHz machine with 1024 MB RAM using the gcc compiler.

Using the same benchmark problems as in section 3, we perform comparisons in Table 3 as follows: for each problem, we randomly select two-thirds of the data for training and leave the remaining for testing. For algorithms except DSVM without alpha seeding, five-fold cross validation with $C = 2^{-10}, 2^{-9.5}, \ldots, 2^8$ on the training set is conducted. For DSVM without alpha seeding, as the training time is huge, only $C$ up to $2^3$ is tried. Then using the $C$ that gives the best cross-validation rate, we train a model and predict the test data. Both testing accuracy and the total computational time are reported.

In Table 3, alpha seeding with $C$ up to $2^8$ is competitive with solving $C$ up to only $2^3$ without alpha seeding. For these problems, considering $C \le 2^3$ is enough, and if alpha seeding stops at $2^3$ as well, it is several times faster than without alpha seeding.

Since alpha seeding is not applied to ASVM and LSVM, we admit that their computational time can be further improved. Results here also serve as the first comparison between ASVM and LSVM. Clearly ASVM is faster. Moreover, due to the huge computational time, we set the maximal iterations of LSVM at 1000. For problems adult and web, after $C$ is large, the iteration limit is reached before stopping conditions are satisfied.

In addition to comparing DSVM with ASVM and LSVM, we compare the performance of SMO type ($q = 2$) and that with a larger working set

Table 4: Comparison of Different Subproblem Size in Decomposition Methods for Linear SVMs.

| | Decomposition Methods with Alpha Seeding | | | |
|---|---|---|---|---|
| | $q = 2$ ($SVM^{light}$) | | $q = 30$ ($SVM^{light}$) | |
| Problem | Total Iterations | Time | Total Iterations | Time |
| australian | 50,145 | 0.71 | 6533 | 1.19 |
| heart | 25,163 | 0.21 | 1317 | 0.33 |
| diabetes | 30,265 | 0.4 | 5378 | 0.31 |
| german | 182,051 | 2.9 | 6006 | 3.68 |
| ijcnn | 345,630 | 185.85 | 79847 | 115.03 |
| adult | 1,666,607 | 1455.2 | 414,798 | 516.71 |
| web | NA[a] | NA[a] | 2,673,578 | 1885.1 |

Notes: Time in seconds; $q$: size of the working set. The time here is shorter than that in Table 3 because we do not perform cross validation. [a] $SVM^{light}$ faced numerical difficulties.

($q = 30$) for DSVM with alpha seeding in Table 4 by modifying the software $SVM^{light}$, which allows adjustable $q$. All default settings of $SVM^{light}$ are used. In this experiment, we solve linear SVMs with $C = 2^{-8}, 2^{-7.5}, \ldots, 2^{8}$ and report their computational time and total number of iterations. Note that when $q$ is two, $SVM^{light}$ and LIBSVM use the same algorithm and differ only in some implementation details.

The results in Table 4 show that the implementation with a larger working set takes less time than that with a smaller one. This is consistent with our earlier statement that for linear SVMs, SMO-type decomposition methods are less favorable.

Regarding the computational time reported in this section, we must caution that quite a few implementation details may affect it. For example, each iteration of ASVM and LSVM involves several matrix vector multiplications. Hence, it is possible to use finely tuned dense linear algebra subroutines. For the LSVM implementation here, by using ATLAS (Whaley, Petitet, & Dongarra, 2000), for large problems, the time is reduced by two-thirds. Thus, it is possible to further reduce the time of ASVM in Table 3, though we find it too complicated to modify the authors' program. Using such tools also means $X$ is considered as a dense matrix. In contrast, $X$ is currently treated as a sparse matrix in both LIBSVM and $SVM^{light}$, where each iteration requires two matrix vector multiplications $X(X^T(\alpha^{k+1} - \alpha^k))$. This sparse format creates some overhead when data are dense.

## 6 Experiments on Model Selection

If the RBF kernel

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

is used, Keerthi and Lin (2003) propose the following model selection procedure for finding good $C$ and $\sigma^2$:

**Algorithm 1.** *Two-line model selection*

1. *Search for the best $C$ of linear SVMs and call it $\tilde{C}$.*
2. *Fix $\tilde{C}$ from step 1 and search for the best $(C, \sigma^2)$ satisfying $\log \sigma^2 = \log C - \log \tilde{C}$ using the RBF kernel.*

That is, we solve a sequence of linear SVMs first and then a sequence of nonlinear SVMs with the RBF kernel. The advantage of algorithm 1 over an exhaustive search of the parameter space is that only parameters on two lines are considered. If decomposition methods are directly used for both linear and nonlinear SVMs here, due to the huge number of iterations, solving the linear SVMs becomes a bottleneck. Our goal is to show that by applying the alpha seeding technique to linear SVMs, the computational time spent on the linear part becomes similar to that on the nonlinear SVMs.

Earlier, in Keerthi and Lin (2003), due to the difficulty of solving linear SVMs, algorithm 1 was tested on only small two-class problems. Here, we would like to evaluate this algorithm on large multiclass data sets. We consider the problems dna, satimage, letter, and shuttle, which were originally from the statlog collection (Michie et al., 1994) and were used in Hsu and Lin (2002a). Except dna, which takes two possible values 0 and 1, each attribute of all training data is scaled to $[-1, 1]$. Then test data are adjusted using the same linear transformation.

Since LIBSVM contains a well-developed cross-validation procedure, we use it as the DSVM solver in this experiment. We search for $\tilde{C}$ by five-fold cross validation on linear SVMs using uniformly spaced $\log_2 \tilde{C}$ value in $[-10, 10]$ (with grid space 1). As LIBSVM considers $\gamma = 1/2\sigma^2$ as the kernel parameter, the second step is to search for good $(C, \gamma)$ satisfying

$$-1 - \log_2 \gamma = \log_2 C - \log_2 \tilde{C}. \tag{6.1}$$

We discretize $[-10, 4]$ as values of $\log_2 \gamma$ and calculate $\log_2 C$ from equation 6.1. To avoid $\log_2 C$ locating in an abnormal region, we consider only points with $-2 \le \log_2 C \le 12$, so the second step may solve fewer SVMs than the first step. The same computational environment as that for section 3 is used.

Since this model selection method is based on the analysis of binary SVMs, a multiclass problem has to be decomposed to several binary SVMs. We employ the one-against-one approach: if there are $k$ classes of data, all $k(k-1)/2$ two-class combinations are considered. For any two classes of data, the model selection is conducted to have the best $(C, \sigma^2)$. With

Table 5: Comparison of Different Model Selection Methods.

| | Complete Grid Search | | | Algorithm 1 | | | |
|---|---|---|---|---|---|---|---|
| Problem | 1 $(C, \sigma^2)$ Accuracy | $k(k-1)/2$ Time | $(C, \sigma^2)$ Accuracy | Time | Time (linear) | Time (nonlinear) | Accuracy |
| dna | 95.62 | 4945 | 95.11 | 202 | 123 | 79 | 94.86 (94.77) |
| satimage | 91.9 | 7860 | 92.2 | 1014 | 743 | 271 | 91.55 (90.55) |
| letter | 97.9 | 56,753 | 97.72 | 5365 | 3423 | 1942 | 96.54 (95.9) |
| shuttle | 99.92 | 104,904 | 99.94 | 4196 | 2802 | 1394 | 99.81 (99.7) |

Notes: Accuracies of algorithm 1 enclosed in parentheses are the accuracies if we search $\log_2 \tilde{C} \in [-10, 3]$ in step 1 of algorithm 1. Time is in seconds.

the $k(k-1)/2$ best $(C, \sigma^2)$ and corresponding decision functions, a voting strategy is used for the final prediction. In Table 5, we compare this approach with two versions of complete grid searches. First, for any two classes of data, five-fold cross validation is conducted on 225 points, a discretization of the $(\log_2 C, \log_2 \gamma) = [-2, 12] \times [-10, 4]$ space. The second way is from the cross-validation procedure adopted by LIBSVM for multiclass data, where a list of $(C, \sigma^2)$ is selected first, and then for each $(C, \sigma^2)$, the one-against-one method is used for estimating the cross-validation accuracy of the multiclass data. Therefore, for the final optimal model, $k(k-1)/2$ decision functions share the same $C$ and $\sigma^2$. Since the same number of nonlinear SVMs is trained, the time for the two complete grid searches is exactly the same, but the performance (test accuracy) may be different. There is no comparison so far, so we present a preliminary investigation here.

Table 5 presents experimental results. For each problem, we compare test accuracy by two complete grid searches and by algorithm 1. The two grid searches are represented as "1 $(C, \sigma^2)$" and "$k(k-1)/2$ $(C, \sigma^2)$," respectively, depending on how many $(C, \sigma^2)$ used by the decision functions. The performances of the three approaches are very similar. However, the total model selection time of algorithm 1 is much shorter. In addition, we also list the accuracy of algorithm 1 in parentheses if we search only $\log_2 \tilde{C}$ value in $[-10, 3]$ in step 1. We find that the accuracy is consistently lower if we search only $\tilde{C}$ in this smaller region. In fact, if we search $\log_2 \tilde{C}$ in $[-10, 3]$, there are many $\log_2 \tilde{C}$ equals to 3 in this experiment. This means that $[-10, 3]$ is too small to cover good parameter regions. We make algorithm 1 practical with using alpha seeding. Otherwise, the time for solving linear SVMs increases greatly, so the proposed model selection does not possess any advantage.

We then investigate the stability of the new model selection approach. Due to timing restrictions, we consider two smaller problems, banana and adult_small, tested in Keerthi and Lin (2003). adult_small, a subset of adult used in section 3, is a binary problem with 1605 examples. Table 6 shows

Table 6: Mean and Standard Deviation of Two Model Selection Methods.

| | Complete Grid Search | | | | | | Algorithm 1 | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\log_2 C$ | | $\log_2 \gamma$ | | Accuracy | | $\log_2 \tilde{C}$ | | Accuracy | |
| Problem | Mean | SD | Mean | SD | Mean | SD | Mean | SD | Mean | SD |
| banana | 7 | 4.45 | -0.4 | 1.51 | 87.91 | 0.47 | -1.9 | 2.18 | 76.36 | 12.21 |
| adult_small | 5.4 | 2.37 | -7.6 | 1.71 | 83.82 | 0.27 | 0.3 | 4.08 | 83.20 | 1.28 |
| dna | 5.4 | 3.34 | -5 | 0 | 95.56 | 0.19 | - | - | 94.85 | 0.20 |
| satimage | 2.5 | 0.71 | 0.1 | 0.57 | 91.74 | 0.24 | - | - | 91.19 | 0.28 |

Note: Each method was applied 10 times.

the means and standard deviations of parameters and accuracy using the $k(k-1)/2$ $(C, \sigma^2)$ grid search and algorithm 1 10 times. For algorithm 1, we list only $\tilde{C}$'s variances because the variances of parameters $C$ and $\sigma^2$, which are computed from equation 6.1, are less meaningful. By applying the same method 10 times, note that different parameters as well as accuracy are due to the randomness of cross validation.

From Table 6, we can see that although the performance (testing accuracy and consumed time) of the model selection algorithm 1 is good, it might be less stable. That is, the variance of accuracy is significantly larger than that of the complete grid search method, while the variances of both parameters are large. We think that in the complete grid search method, the cross-validation estimation bounds the overall error. Thus, the variances of gained parameters do not affect the testing performance. However, in the two-line search method (algorithm 1), two-stage cross validations are utilized. Thus, the variance in the first stage may affect the best performance of the second stage.

## 7 Discussion and Conclusion

It is arguable that we may have used a too strict a stopping condition in DSVM when $C$ is large. One possibility is to use the stopping tolerance that is proportional to $C$. This will reduce the number of iterations so that directly solving linear SVMs with large $C$ may be possible. However, in the appendix of Chung et al. (2002), we show that even in these settings, DSVM with alpha seeding still makes the computational time several times faster than the original DSVM, especially for large data sets. Moreover, a stopping tolerance that is too large will cause DSVM to stop with wrong solutions.

In conclusion, we hope that based on this work, SVM software using decomposition methods can be suitable for all types of problems, both $n \ll l$ and $n \gg l$.

**Acknowledgments**

This work was supported in part by the National Science Council of Taiwan, by grant NSC 90-2213-E-002-111. We thank Thorsten Joachims for help on modifying $SVM^{light}$ for experiments in section 5.

**References**

Blake, C. L., & Merz, C. J. (1998). *UCI repository of machine learning databases* (Tech. Rep.). Irvine, CA: University of California, Department of Information and Computer Science. Available on-line at: http://www.ics.uci.edu/~mlearn/MLRepository.html.

Chang, C.-C., & Lin, C.-J. (2001a). IJCNN 2001 challenge: Generalization ability and text decoding. In *Proceedings of IJCNN*. New York: IEEE.

Chang, C.-C., & Lin, C.-J. (2001b). *LIBSVM: A library for support vector machines*. Available on-line at: http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Cortes, C., & Vapnik, V. (1995). Support-vector network. *Machine Learning, 20*, 273–297.

DeCoste, D., & Wagstaff, K. (2000). Alpha seeding for support vector machines. In *Proceedings of International Conference on Knowledge Discovery and Data Mining*. New York: ACM Press.

Fine, S., & Scheinberg, K. (2001). Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research, 2*, 243–264.

Hsu, C.-W., & Lin, C.-J. (2002a). A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks, 13*(2), 415–425.

Hsu, C.-W., & Lin, C.-J. (2002b). A simple decomposition method for support vector machines. *Machine Learning, 46*, 291–314.

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.

Kao, W.-C., Chung, K.-M., Sun, T., & Lin, C.-J. (2002). *Decomposition methods for linear support vector machines* (Tech. Rep.). Taipei: Department of Computer Science and Information Engineering, National Taiwan University. Available on-line at: http://www.csie.ntu.edu.tw/~cjlin/papers/linear.pdf.

Keerthi, S. S., & Lin, C.-J. (2003). Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Computation, 15*(7), 1667–1689.

Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., & Murthy, K. R. K. (2000). A fast iterative nearest point algorithm for support vector machine classifier design. *IEEE Transactions on Neural Networks, 11*(1), 124–136.

Lee, Y.-J., & Mangasarian, O. L. (2001). RSVM: Reduced support vector machines. In *Proceedings of the First SIAM International Conference on Data Mining*. Philadelphia: SIAM.

Lin, K.-M. (2002). *Reduction techniques for training support vector machines*. Unpublished master's thesis, National Taiwan University.

Lin, K.-M., & Lin, C.-J. (2003). A study on reduced support vector machines. *IEEE Transactions on Neural Networks, 14*(6), 1449–1559.

Mangasarian, O. L., & Musicant, D. R. (2000). Active set support vector machine classification. In T. K. Leen, T. G. Dietterich, & V. Tresp (Eds.), *Advances in neural information processing systems, 13* (pp. 577–583). Cambridge, MA: MIT Press.

Mangasarian, O. L., & Musicant, D. R. (2001). Lagrangian support vector machines. *Journal of Machine Learning Research, 1*, 161–177.

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

Osuna, E., Freund, R., & Girosi, F. (1997). Training support vector machines: An application to face detection. In *Proceedings of CVPR'97* (pp. 130–136). New York: IEEE.

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.

Prokhorov, D. (2001). IJCNN 2001 neural network competition. Slide presentation in IJCNN'01, Ford Research Laboratory. Available on-line at: http://www.geocities.com/ijcnn/nnc_ijcnn01.pdf.

Whaley, R. C., Petitet, A., & Dongarra, J. J. (2000). *Automatically tuned linear algebra software and the ATLAS project* (Tech. Rep.). Chattanooga: Department of Computer Sciences, University of Tennessee.

Received March 7, 2003; accepted January 7, 2004.