Fast Training of Linear Programming Support Vector Machines Using Decomposition Techniques

Yusuke Torii[1] and Shigeo Abe[2]

[1] Kobe University, Rokkodai, Nada, Kobe, Japan
057t238n@stu.kobe-u.ac.jp
[2] Kobe University, Rokkodai, Nada, Kobe, Japan
abe@kobe-u.ac.jp

Abstract. Decomposition techniques are used to speed up training support vector machines but for linear programming support vector machines (LP-SVMs) direct implementation of decomposition techniques leads to infinite loops. To solve this problem and to further speed up training, in this paper, we propose an improved decomposition techniques for training LP-SVMs. If an infinite loop is detected, we include in the next working set all the data in the working sets that form the infinite loop. To further accelerate training, we improve a working set selection strategy: at each iteration step, we check the number of violations of complementarity conditions and constraints. If the number of violations increases, we conclude that the important data are removed from the working set and restore the data into the working set. The computer experiments demonstrate that training by the proposed decomposition technique with improved working set selection is drastically faster than that without using the decomposition technique. Furthermore, it is always faster than that without improving the working set selection for all the cases tested.

1 Introduction

Decomposition techniques [1] are widely used to speed up training of support vector machines (SVMs) [2,3] for large size problems. Stable convergence to solutions by decomposition techniques is verified both by computer experiments and theoretical analysis [4,5]. In [5], the sequential minimum optimization technique, which uses a decomposition technique with a working set size of two, is shown to converge asymptotically if the most violating variables are selected.

But for a linear programming support vector machine (LP-SVM), in which the quadratic objective function in an SVM is replaced with a linear function [6], direct implementation of decomposition techniques sometimes leads to infinite loops. But this phenomenon has not been discussed so far.

In this paper, we propose decomposition techniques for training an LP-SVM that resolve infinite loops and speed up training by improved working set selection. In training an LP-SVM by decomposition techniques, first, we select the

F. Schwenker and S. Marinai (Eds.): ANNPR 2006, LNAI 4087, pp. 165–176, 2006.

initial working set randomly, and optimize the subproblem. Using the primal and dual solutions, we check if each of the training data satisfies the complementarity conditions and the constraints. And if all the training data satisfy the complementarity conditions and the constraints, we finish training.

But if there exist training data that do not satisfy the complementarity conditions or the constraints, we select the working set again. In selecting the working set, we detect the variables, in the fixed set, that do not satisfy complementarity conditions, and move them to the working set. And we detect the data, in the working set, that is not support vectors of the subproblem, and move them to the fixed set. Then, we optimize the new subproblem and iterate the algorithm until all the training data satisfy the complementarity conditions and the constraints.

The above training method sometimes leads to infinite loops, in which the same sequence of working sets repeatedly appears. To resolve infinite loops, if an infinite loop is detected, we include in the new working set all the data that are in the working sets that form the infinite loop. This working set strategy works to resolve infinite loops but according to our experiments, many iteration steps are spent before the solution goes into an infinite loop. Thus, to further speed up training, we propose an improved working set selection strategy. Namely, at each iteration step, we check the number of violations of the complementarity conditions and constraints. If the number of violations increases at some step, we conclude that important data were removed at the previous step of working set selection and add those data to the next working set.

The structure of this paper is as follows. In Section 2, we summarize the architecture of LP-SVMs, and in Section 3, we discuss the proposed method. In Section 4, we show the simulation results using benchmark data sets and in Section 5, we describe the conclusions.

2 Linear Programming Support Vector Machines

Let m-dimensional training inputs $\mathbf{x}_i\,(i = 1, ..., M)$ belong to Class 1 or 2 and the associated labels be $y_i = 1$ for Class 1 and -1 for Class 2, where M is the number of training inputs. In the normal SVMs [7], we determine the decision function by

$$D(\mathbf{x}) = \mathbf{w}^T \mathbf{g}(\mathbf{x}) + b, \tag{1}$$

where $\mathbf{w}$ is an l-dimensional vector, b is a scalar, and $\mathbf{g}(\mathbf{x})$ is the mapping function that maps m-dimensional vector $\mathbf{x}$ into the l-dimensional feature space. The optimal separating hyperplane can be obtained by solving the following quadratic programming problem:

$$\text{Minimize} \quad Q(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{M} \xi_i \tag{2}$$

$$\text{subject to} \quad y_i(\mathbf{w}^t \mathbf{g}(\mathbf{x}_i) + b) \geq 1 - \xi_i \quad \text{for} \;\; i = 1, ..., M, \tag{3}$$

where C is the margin parameter that determines the tradeoff between the maximization of the margin and minimization of the classification error, and ξ_i is the nonnegative slack variable for $\mathbf{x}_i$.

By replacing the L2-norm $\|\mathbf{w}\|_2^2 = w_1^2 + w_2^2 + \cdots + w_l^2$ in the objective function (2) with an L1-norm $\|\mathbf{w}\|_1 = |w_1| + |w_2| + \cdots + |w_l|$, the SVM becomes as follows:

$$\text{Minimize} \quad Q(\mathbf{w}, \boldsymbol{\xi}) = \sum_{i=1}^{l} |w_i| + C \sum_{i=1}^{M} \xi_i \tag{4}$$

$$\text{subject to} \quad y_i(\mathbf{w}^t \mathbf{g}(\mathbf{x}_i) + b) \geq 1 - \xi_i \quad \text{for} \;\; i = 1, ..., M. \tag{5}$$

By this formulation, for the linear kernel, i.e., $\mathbf{g}(\mathbf{x}) = \mathbf{x}$, we can solve the problem by linear programming. However, for the kernels other than linear kernels, we need to treat the feature space explicitly.

To apply linear programming to the feature space, we define the decision function in the dual form as follows [8]:

$$D(\mathbf{x}) = \sum_{i=1}^{M} \alpha_i H(\mathbf{x}, \mathbf{x}_i) + b, \tag{6}$$

where α_i and b take on real values. Thus, we need not use label numbers. And $H(\mathbf{x}, \mathbf{x}')$ is a kernel function that is given by

$$H(\mathbf{x}, \mathbf{x}') = \mathbf{g}(\mathbf{x})^T \mathbf{g}(\mathbf{x}'). \tag{7}$$

The kernels that are used in our study are as follows:

- polynomial kernels: $H(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + 1)^d$, where d is a positive integer,
- RBF kernels: $H(\mathbf{x}, \mathbf{x}') = \exp(-\gamma ||\mathbf{x} - \mathbf{x}'||^2)$, where γ is a positive parameter.

Then we consider solving the following linear programming problem:

$$\text{Minimize} \quad Q(\boldsymbol{\alpha}, \boldsymbol{\xi}) = \sum_{i=1}^{M} (|\alpha_i| + C\xi_i) \tag{8}$$

$$\text{subject to} \quad y_j \left(\sum_{i=1}^{M} \alpha_i H(\mathbf{x}_j, \mathbf{x}_i) + b \right) \geq 1 - \xi_j \quad \text{for} \;\; j = 1, \ldots, M, \tag{9}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M)^T$ and $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_M)^T$. Letting $\alpha_i = \alpha_i^+ - \alpha_i^-$ and $b = b^+ - b^-$, where $\alpha_i^+ \geq 0$, $\alpha_i^- \geq 0$, $b^+ \geq 0$, $b^- \geq 0$, we can solve (8) and (9) for $\boldsymbol{\alpha}$, b, and $\boldsymbol{\xi}$ by linear programming. Furthermore, we introduce the slack variables u_i $(i = 1, \ldots, M)$ into (9). Then (8) and (9) become as follows:

$$\text{Minimize} \quad Q(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-, \boldsymbol{\xi}) = \sum_{i=1}^{M} (\alpha_i^+ + \alpha_i^- + C\xi_i) \tag{10}$$

subject to

$$y_j \left(\sum_{i=1}^{M} (\alpha_i^+ - \alpha_i^-) H(\mathbf{x}_j, \mathbf{x}_i) + b^+ - b^- \right) + \xi_j = 1 + u_j \quad \text{for} \;\; j = 1, \ldots, M. \tag{11}$$

And the decision function (6) becomes

$$D(\mathbf{x}) = \sum_{i=1}^{M} (\alpha_i^+ - \alpha_i^-) H(\mathbf{x}, \mathbf{x}_i) + b^+ - b^-. \tag{12}$$

But we must notice that since $\mathbf{w} = \sum_{i=1}^{M} \alpha_i \, \mathbf{g}(\mathbf{x}_i)$, minimization of the sum of $|\alpha_i|$ does not lead to maximization of the margin measured in the L1 norm.

Let (10) and (11) be a primal problem. Then the dual problem is given as follows:

$$\text{Maximize} \quad \sum_{i=1}^{M} z_i, \tag{13}$$

$$\text{subject to} \quad \sum_{i=1}^{M} y_i H(\mathbf{x}_i, \mathbf{x}_j) z_i + v_j^+ = 1 \quad \text{for} \;\; j = 1, \ldots, M, \tag{14}$$

$$\sum_{i=1}^{M} y_i H(\mathbf{x}_i, \mathbf{x}_j) z_i = v_j^- - 1 \quad \text{for} \;\; j = 1, \ldots, M, \tag{15}$$

$$\sum_{i=1}^{M} y_i z_i = 0, \tag{16}$$

$$z_j \; + w_j = C \quad \text{for} \;\; j = 1, \ldots, M, \tag{17}$$

where $z_i \geq 0$ $(i = 1, \ldots, M)$ are dual variables, and $v_i^+ \geq 0$, $v_i^- \geq 0$, $w_i \geq 0$ $(i = 1, \ldots, M)$ are slack variables.

By this formulation, in the primal problem, the number of variables is $4M + 2$ and the number of equality constrains is M. In the dual problem, the number of variables is $4M$ and the number of equality constrains is $3M + 1$. Thus for a large number of training data, training becomes slow even by linear programming. Therefore, we need to use decomposition techniques.

We can solve above primal problem (10) and (11) or dual problem (13)–(17) by linear programming. If we optimize a linear programming problem by the simplex method, we need only to solve the primal or dual problem. If we solve one, the other is also solved [10,11]. Therefore, in this paper, we solve only the primal problem.

By solving (10) and (11), we obtain the primal and dual solutions. If these solutions are optimal, they satisfy the following complementarity conditions:

$$\alpha_i^+ v_i^+ = 0 \quad \text{for} \;\; i = 1, \ldots, M, \tag{18}$$

$$\alpha_i^- v_i^- = 0 \quad \text{for} \;\; i = 1, \ldots, M, \tag{19}$$

$$\xi_i w_i = 0 \quad \text{for} \;\; i = 1, \ldots, M, \tag{20}$$

$$u_i z_i = 0 \quad \text{for} \;\; i = 1, \ldots, M. \tag{21}$$

The training data $\mathbf{x}_i$ that satisfy

$$\alpha_i = \alpha_i^+ - \alpha_i^- = 0, \tag{22}$$

$$\xi_i = 0, \tag{23}$$

$$z_i = 0, \tag{24}$$

do not affect the solution even if they are removed. Namely, the training data $\mathbf{x}_i$ that do not satisfy (22)–(24) are support vectors.

3 Proposed Method

In this section, we discuss the decomposition technique for the LP-SVM. If we directly implement the strategy for working set selection developed for normal SVMs [1], the solution often goes into an infinite loop. Even if it does not, the convergence is usually slow. To overcome these, in Subsection 3.1, we discuss a strategy for working set selection to avoid infinite loops, and in Subsection 3.2, we further refine the working set selection to speed up training.

3.1 Decomposition Techniques for LP-SVMs

In decomposition techniques for an LP-SVM, we iterate optimizing subproblems that are smaller than the original optimization problem (10) and (11). Namely, we decompose the index set $T = \{1, \ldots, M\}$ into two sets W and F, where W is a working set and F is a fixed set. Here, $W \cup F = \{1, \ldots, M\}$ and $W \cap F = \phi$. Then we decompose $\boldsymbol{\alpha}^+ = \{\alpha_i^+ | i = 1, \ldots, M\}$ into $\boldsymbol{\alpha}_W^+ = \{\alpha_i^+ | i \in W\}$ and $\boldsymbol{\alpha}_F^+ = \{\alpha_i^+ | i \in F\}$. Likewise, we decompose the remaining variables, i.e., decompose $\boldsymbol{\alpha}^-$ into $\boldsymbol{\alpha}_W^-$ and $\boldsymbol{\alpha}_F^-$, $\boldsymbol{\xi}$ into $\boldsymbol{\xi}_W$ and $\boldsymbol{\xi}_F$, $\mathbf{u}$ into $\mathbf{u}_W$ and $\mathbf{u}_F$, $\mathbf{v}^+$ into $\mathbf{v}_W^+$ and $\mathbf{v}_F^+$, $\mathbf{v}^-$ into $\mathbf{v}_W^-$ and $\mathbf{v}_F^-$, $\mathbf{w}$ into $\mathbf{w}_W$ and $\mathbf{w}_F$, and $\mathbf{z}$ into $\mathbf{z}_W$ and $\mathbf{z}_F$.

And we define the following subproblem. Fixing $\boldsymbol{\alpha}_F^+$ and $\boldsymbol{\alpha}_F^-$,

$$\text{minimize} \quad Q(\boldsymbol{\alpha}_W^+, \boldsymbol{\alpha}_W^-, \boldsymbol{\xi}_W) = \sum_{i \in W} (\alpha_i^+ + \alpha_i^- + C\xi_i) \tag{25}$$

subject to

$$y_j \left(\sum_{i \in W} (\alpha_i^+ - \alpha_i^-) H(\mathbf{x}_i, \mathbf{x}_j) + b^+ - b^- + \sum_{i \in F} (\alpha_i^+ - \alpha_i^-) H(\mathbf{x}_i, \mathbf{x}_j) \right) + \xi_j = 1 + u_j \quad \text{for} \quad j \in W. \tag{26}$$

Solving (25) and (26), we obtain $\boldsymbol{\alpha}_W^+$, $\boldsymbol{\alpha}_W^-$, $\boldsymbol{\xi}_W$, $\mathbf{u}_W$, $\mathbf{v}_W^+$, $\mathbf{v}_W^-$, $\mathbf{w}_W$, and $\mathbf{z}_W$. These vectors constitute the optimal solution for the subproblem associated with the working set W. But the optimal solution for the subproblem may not be optimal for the entire problem (10) and (11). To check if the obtained solution is optimal for the entire problem, we need to determine the values for the variables in the fixed set.

First, we obtain primal variables $\boldsymbol{\alpha}_F^+$, $\boldsymbol{\alpha}_F^-$, $\boldsymbol{\xi}_F$, and $\mathbf{u}_F$. Fixing $\boldsymbol{\alpha}_F^+ = \mathbf{0}$ and $\boldsymbol{\alpha}_F^+ = \mathbf{0}$, we obtain $\boldsymbol{\xi}_F$ and $\mathbf{u}_F$ by the following constraints

$$y_j \left(\sum_{i \in W, F} (\alpha_i^+ - \alpha_i^-) H(\mathbf{x}_i, \mathbf{x}_j) + b^+ - b^- \right) + \xi_j = 1 + u_j \quad \text{for} \quad j \in F. \tag{27}$$

Here, (27) is a subset of (11). Using the decision function (12), the constraints (27) become

$$y_j D(\mathbf{x}_j) + \xi_j = 1 + u_j \quad \text{for} \quad j \in F. \tag{28}$$

Then we obtain $\boldsymbol{\xi}_F$ and $\mathbf{u}_F$ as follows:

1. If $y_j D(\mathbf{x}_j) > 1$, $\xi_j = 0$. Therefore, from (28), $u_j = y_j D(\mathbf{x}_j) - 1$.
2. If $y_j D(\mathbf{x}_j) \leq 1$, $\xi_j = 1 - y_j D(\mathbf{x}_j)$. Therefore from (28), $u_j = 0$

Secondly, we obtain the dual variables $\mathbf{v}_F^+$, $\mathbf{v}_F^-$, $\mathbf{w}_F$, and $\mathbf{z}_F$. Fixing $\mathbf{z}_F = \mathbf{0}$, we obtain $\mathbf{v}_F^+$, $\mathbf{v}_F^-$, and $\mathbf{w}_F$ by the following constraints:

$$v_j^+ = 1 - \sum_{i=1}^{M} y_i H(\mathbf{x}_i, \mathbf{x}_j) z_i \quad \text{for} \quad j \in F, \tag{29}$$

$$v_j^- = 1 + \sum_{i=1}^{M} y_i H(\mathbf{x}_i, \mathbf{x}_j) z_i \quad \text{for} \quad j \in F, \tag{30}$$

$$w_j = C \qquad \text{for} \quad j \in F. \tag{31}$$

Here, (29)–(31) are obtained from (14), (15), and (17). We must notice that when we obtain v_j^+, v_j^- $(j \in F)$ from (29) and (30), they may take negative values.

In this way, we obtain $\boldsymbol{\alpha}_F^+$, $\boldsymbol{\alpha}_F^-$, $\boldsymbol{\xi}_F$, $\mathbf{u}_F$, $\mathbf{v}_F^+$, $\mathbf{v}_F^-$, $\mathbf{w}_F$, and $\mathbf{z}_F$. Next, we check if each of the variables satisfies the complementarity conditions (18)–(21) and the constraints. But after solving the subproblem whose variables constitute the working set, the variables satisfy both the complementarity conditions and the constraints because they are the optimal solution of the subproblem. Thus we need to check only the variables in the fixed set. As is apparent from the foregoing discussions, the variables in the fixed set satisfy the constraints (27), (29)–(31), but v_j^+, v_j^- $(j \in F)$ may take negative values. This is the violation of the constraints $v_j^+ \geq 0$ and $v_j^- \geq 0$. Furthermore, variables in the fixed set may violate the complementarity conditions (18)–(21). Therefore, we detect the variables that do not satisfy the complementarity conditions or the constraints in the fixed set. If they exist, we add them to the working set. Meanwhile, in the working set, we detect the data that are not support vectors of the subproblem, i.e., the data that satisfy (22)–(24). And we move the data that are not support vectors of the subproblem from the working set to the fixed set. And we iterate training until all the training data satisfy the complementarity conditions and the constraints.

We also finish training in the case where the value of the objective function changes little from the previous iteration, i.e., $|Q_k - Q_{k-1}| < \epsilon$, where Q_k is the value of the objective function at the kth iteration and ϵ is a small positive parameter.

The above mentioned working set selection strategy, however, often leads to an infinite loop. In the infinite loop, the same working set is selected repeatedly.

The detail of the infinite loop is as follows. Let the working set sequence be

$$\cdots,\ W_k,\ W_{k+1},\cdots,\ W_{k+t},\ W_{k+t+1},\ W_{k+t+2},\cdots,W_{k+2t+1},\cdots,$$

where W_k is the working set at the kth iteration. If

$$W_k = W_{k+t+1},\quad W_{k+1} = W_{k+t+2},\quad \cdots\quad ,W_{k+t} = W_{k+2t+1},$$

the same sequence of working sets is repeated infinitely. Namely, an infinite loop occurs. Infinite loops occur because some data that are removed at some iteration step violate the complementarity conditions or constraints in the subsequent step and move back to the working set. Thus the simplest way to avoid an infinite loop is as follows. If we find an infinite loop at the $(k + 2t + 1)$th iteration, we set $W_{k+2t+2} = W_k \cup W_{k+1} \cup \cdots \cup W_{k+t}$.

According to the above discussion, a procedure for training an LP-SVM using the decomposition techniques that avoids infinite loops is as follows.

Step 1

We initialize $\boldsymbol{\alpha}^+ = \mathbf{0},\ \boldsymbol{\alpha}^- = \mathbf{0},\ \mathbf{z} = \mathbf{0}$, and $k = 1$, where k is the iteration number.

Step 2

We set q points from the training data set to W_1, where q is a positive integer and W_1 is an initial working set.

Step 3

We optimize the subproblem for the working set W_k.

Step 4

We obtain variables in the fixed set by (28)–(31).

Step 5

We check if each of the training data in the fixed set satisfies the complementarity conditions (18)–(21) and the constraints $v_i^+ \geq 0,\ v_i^- \geq 0\ \ (i = 1, \ldots, M)$. If there exist training data that violate the complementarity conditions or the constraints, we go to Step 6. If all the training data satisfy the complementarity conditions and the constraints or $|Q_k - Q_{k-1}| < \epsilon$, we finish training.

Step 6

We check if the infinite loop exists. If it exists, we add all the data in the working sets that form the infinite loop to the next working set W_{k+1}. And we add 1 to k and go to Step 3. Otherwise we go to Step 7.

Step 7

In the fixed set, we detect the variables that violate the complementarity conditions or the constraints, and move at most q points to the working set W_{k+1}. In the working set, we detect the data that are not support vectors of the subproblem, and move them to the fixed set. And we add 1 to k and go to Step 3.

3.2 Improving Working Set Selection

The decomposition technique for an LP-SVM discussed in Subsection 3.1 can resolve infinite loops. But according to our experiments, usually an infinite loop appears after long iteration steps. Therefore, if an infinite loop appears, training is usually very slow. In this subsection, we discuss how to accelerate training by improving the working set selection strategy.

The data that are support vectors of the entire problem are very important because if all of these data are in the working set, the optimal solution for the subproblem is the optimal solution for all the training data. In selecting a working set, it often occurs that these important data, i.e., support vectors of the entire problem, go out of the working set. But in many cases, these important data return back to the working set in the subsequent iteration step. In particular, when an infinite loop occurs, these important data repeatedly go out of and return back to the working set.

Let V_k be the number of data that violate the complementarity conditions or the constraints at the kth iteration. In general, V_k is large in early stage of training. And as training proceeds while selecting the working set, V_k gets smaller and smaller until it reaches 0, at which step we finish training. But if we observe the value of V_k during training, it sometimes increases. This is attributed to the fact that important data for training go out of the working set.

Therefore, when V_k increases, i.e., $V_k \geq V_{k-1}$, we can conclude that the data that were in W_{k-1} went out of W_{k-1} at the $(k-1)$th iteration and violate the complementarity conditions or the constraints after training using W_k. That is, these data try to go back to the working set soon after they go away since these data are important for training. Therefore, our new strategy is to return back these data to the working set W_{k+1}. Here, we must notice that for $V_k \geq V_{k-1}$ we do not remove any data from W_k. We only return back data into the working set. This is because if we remove the data from W_k, important data may be removed.

By implementing this process, we can stop important data from going in and out of the working set. Thus, it leads to accelerating training. A procedure for training an LP-SVM using the improved decomposition technique is as follows.

Step 1
We initialize $\boldsymbol{\alpha}^+ = \mathbf{0},\ \boldsymbol{\alpha}^- = \mathbf{0},\ \mathbf{z} = \mathbf{0}$, and $k = 1$, where k is the iteration number.

Step 2
We set q points from the training data set to W_1, where q is a positive integer and W_1 is an initial working set.

Step 3
We optimize the subproblem for the working set W_k.

Step 4
We obtain variables in the fixed set by (28)–(31).

Step 5
We check if each of the training data in the fixed set satisfies the complementarity conditions (18)–(21) and the constraints $v_i^+ \geq 0$, $v_i^- \geq 0$ $(i =$

$1, \ldots, M$). If there exist training data that violate the complementarity conditions or the constraints, we go to Step 6. If all the training data satisfy the complementarity conditions and the constraints or $|Q_k - Q_{k-1}| < \epsilon$, we finish training.

Step 6

We check if we are in the infinite loop. If so, we add all the data in the working sets that form the infinite loop to the next working set W_{k+1}. And we add 1 to k and go to Step 3. Otherwise we go to Step 7.

Step 7

If $V_k < V_{k-1}$, we go to Step 8. Otherwise, we detect the data that were in W_{k-1} but went out of W_{k-1} at the $(k-1)$th iteration and violate the complementarity conditions or the constraints at the kth iteration. If there exist such data, we add these data to the working set W_{k+1}. And we add 1 to k and go to Step 3. But if no such data exist, we go to Step 8.

Step 8

In the fixed set, we detect the variables that violate the complementarity conditions or the constraints, and move at most q points to the working set W_{k+1}. From the working set, we detect the data that are not support vectors of the subproblem and move them to the fixed set. And we add 1 to k and go to Step 3.

4 Simulation Experiments

In this section, we show two experimental results. In the first experiment, we show the effectiveness of the improved decomposition techniques discussed in Subsection 3.2 over training without decomposition. In the second experiment, we compare the improved decomposition technique with the original decomposition technique discussed in Subsection 3.1.

Linear programming can be solved either by the simplex method or the primal-dual interior-point method. But in our experiments, we use the "lp.c" [9], which is a program for the simplex method.

The data sets used to evaluate the performance are multiclass data sets: the numeral data for license plate recognition [12], the blood cell data [13], the thyroid data [14], and hiragana data [15,16]. Table 1 shows the numbers of inputs, classes, training data, and test data of the benchmark data sets. We use one-against-all support vector machines [7]. Therefore, all the training data are used for training.

Table 1. Benchmark data specification

Data	Inputs	Classes	Trn.	Test
Numeral	12	10	810	820
Blood cell	13	12	3097	3100
Thyroid	21	3	3772	3428
Hiragana-50	50	39	4610	4610
Hiragana-13	13	38	8375	8356

4.1 Effect of the Decomposition Techniques

We evaluate the speedup by using the improved decomposition technique.

Figure 1 shows the training time for the change of q, namely the number of data added to the working set, using the blood cell data. We use polynomial kernels with $d = 3$ and fix the margin parameter $C = 1000$. From the figure, it is seen that training is accelerated most when q is around 100.

Table 2 shows the optimum value of q and the speedup by the improved decomposition techniques. "Dec.," "No-Dec.," and "Speedup" denote that the decomposition technique is used, not used, and the speedup obtained by the improved decomposition technique. From the table, it is seen that we can speed up training drastically by the improved decomposition technique for all the data sets.

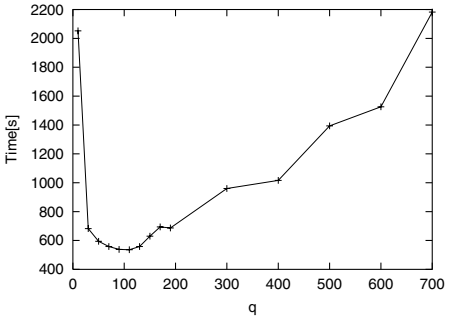

Fig. 1. Training time for the change of q

Table 2. Optimum value of q and the speedup by the improved decomposition techniques

Data	Term	Dec.	No-Dec.
Numeral	Optimum q	30	–
	Rate[%]	99.51	99.51
	Time[s]	1.2	361
	Speedup	301	1
Blood cell	Optimum q	110	–
	Rate[%]	92.58	92.58
	Time[s]	536	45840
	Speedup	86	1
Thyroid	Optimum q	90	–
	Rate[%]	97.17	97.17
	Time[s]	840	58887
	Speedup	70	1

4.2 Comparison Between Original and Improved Decomposition Techniques

In Table 3, we list performance comparison of the original and improved decomposition techniques. We use polynomial kernels with $d = 3$ and RBF kernels with $\gamma = 1$. The value of C is 10 and 10000. We set q as 80 for all the cases. From the table, it is seen that the improved decomposition techniques is faster than the original decomposition techniques for all the cases.

Figures 2 and 3 show the numbers of violations of complementarity conditions and constraints for the original and improved methods, respectively. The results are obtained for Class 1 against others using the numeral data with polynomial kernels with $d = 3$ and $C = 1000$. In Fig. 2, the numbers of violations fluctuate very much and the convergence is very slow. As explained previously, this is because the important data are removed from the working set. But in Fig. 3, the fluctuation is quickly subdued because important data are restored when the number of violations increases.

Table 3. Comparison between original and improved decomposition techniques

		Original		Improved		
Data	Parameter	Rate[%]	Time[s]	Rate[%]	Time[s]	Speedup
Numeral	$d3, C10$	89.39(92.47)	15.5	99.51(100)	1.8	8.6
	$d3, C10000$	99.51(100)	13.9	99.51(100)	1.8	7.6
	$\gamma 1, C10$	99.63(100)	6.2	99.63(100)	1.8	3.4
	$\gamma 1, C10000$	99.63(100)	2.1	99.63(100)	2.0	1.1
Blood cell	$d3, C10$	91.77(95.12)	344	91.77(95.12)	268	1.3
	$d3, C10000$	92.19(99.19)	6481	92.23(99.19)	1180	5.5
	$\gamma 1, C10$	91.00(93.00)	300	91.00(93.00)	219	1.4
	$\gamma 1, C10000$	91.90(99.13)	8760	91.87(99.16)	1136	7.7
Thyroid	$d3, C10$	95.74(96.77)	5498	95.74(96.77)	4294	1.3
	$d3, C10000$	97.20(99.47)	3204	97.20(99.47)	749	4.3
	$\gamma 1, C10$	95.01(95.55)	3202	95.01(95.55)	2365	1.4
	$\gamma 1, C10000$	97.43(99.50)	8891	97.43(99.50)	1323	6.7
Hiragana-50	$d3, C10$	98.61(100)	21031	98.59(100)	748	28.1
	$d3, C10000$	92.75(94.36)	20467	98.59(100)	764	26.8
	$\gamma 1, C10$	98.24(100)	537	98.24(100)	261	2.1
	$\gamma 1, C10000$	98.24(100)	463	98.24(100)	249	1.9
Hiragana-13	$d3, C10$	99.10(99.87)	10612	99.88(99.15)	927	11.4
	$d3, C10000$	95.75(96.73)	26179	98.96(100)	976	26.8
	$\gamma 1, C10$	98.86(99.16)	2713	98.84(99.15)	743	3.7
	$\gamma 1, C10000$	92.29(100)	22555	99.28(100)	917	24.6

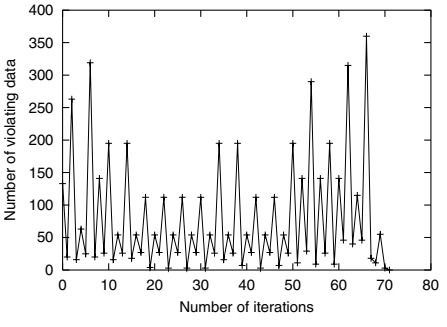

Fig. 2. The number of violations during training for the original method

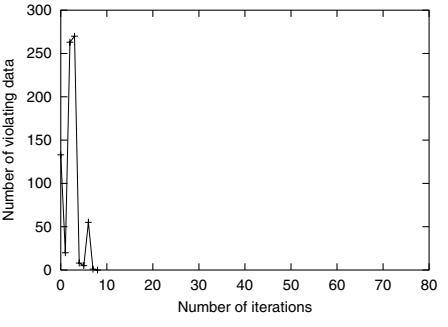

Fig. 3. The number of violations during training for the improved method

5 Conclusions

In this paper, we formulated the decomposition technique for the LP-SVM and proposed resolving the infinite loop that occurs during training. Furthermore, we proposed an improved working set selection strategy to speed up training.

In the decomposition techniques for LP-SVMs, we select the working set using the complementarity conditions, but unlike the original SVMs this often leads to an infinite loop. When an infinite loop is detected, we resolve the infinite loop by adding all the data in the infinite loop to the working set. And to speed up training, we check if the number of the violating data is increased. If so, we prohibit the important data from going out of the working set.

Using the benchmark data sets, we showed that we can speed up training by the decomposition techniques and that the improved decomposition technique can train an LP-SVM faster than the original decomposition technique.

References

1. E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," *Proc. NNSP 97*, pp. 276–285, 1997.
2. V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 1995.
3. V. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, 1998.
4. C.-J. Lin, "On the convergence of the decomposition method for support vector machines," *IEEE Trans. Neural Networks*, Vol. 12, No. 6, pp. 1288–1298, 2001.
5. S. S. Keerthi and E. G. Gilbert, "Convergence of a generalized SMO algorithm for SVM classifier design," *Machine Learning*, Vol. 46, pp. 351–360, 2002.
6. K. P. Bennett, "Combining support vector and mathematical programming methods for classification," In B. Schölkopf et al., Eds., *Advances in Kernel Methods: Support Vector Learning*, pp. 307–326, MIT Press, 1999.
7. S. Abe, *Support Vector Machines for Pattern Classification*, Springer, 2005.
8. B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, 2002.
9. T. Yamada, "lp.c,", http://www.nda.ac.jp/˜yamada/programs/lp.c.
10. R. J. Vanderbei, *Linear Programming*, Kluwer Academic Publishers, 2nd Ed., 2001.
11. V. Chavátal, *Linear Programming*, W. H. Freeman and Company, 1983.
12. H. Takenaga et al., "Input layer optimization of neural networks by sensitivity analysis and its application to recognition of numerals," *Electrical Engineering in Japan*, Vol. 111, No. 4, pp. 130–138, 1991.
13. A. Hashizume, J. Motoike, and R. Yabe, "Fully automated blood cell differential system and its application," *Proc. IUPAC 3rd International Congress on Automation and New Technology in the Clinical Laboratory*, pp. 297–302, 1988.
14. S. M. Weiss and I. Kapouleas, "An empirical comparison of pattern recognition, neural nets, and machine learning classification methods," *Proc. IJCAI*, pp. 781–787, 1989.
15. M.-S. Lan, H. Takenaga, and S. Abe, "Character recognition using fuzzy rules extracted from data," *Proc. 3rd IEEE International Conference on Fuzzy Systems*, Vol. 1, pp. 415–420, 1994.
16. S. Abe, *Pattern Classification: Neuro-Fuzzy Methods and Their Comparison*, Springer, 2001.