



Fusing sequential minimal optimization and Newton's method for support vector training

Abe, Shigeo

(Citation)

International Journal of Machine Learning and Cybernetics, 7(3):345-364

(Issue Date)

2016-06

(Resource Type)

journal article

(Version)

Accepted Manuscript

(Rights)

First Online: 22 May 2014 ©Springer-Verlag 2014. The final publication is available at Springer via <http://dx.doi.org/10.1007/s13042-014-0265-x>

(URL)

<https://hdl.handle.net/20.500.14094/90003294>



Fusing Sequential Minimal Optimization and Newton's Method for Support Vector Training

Shigeo Abe

Received: date / Accepted: date

Abstract Sequential Minimal Optimization (SMO) is widely used for training support vector machines (SVMs) because of fast training. But the training slows down when a large margin parameter value is used. Training by Newton's method (NM) accelerates training in such a situation but it slows down for a small margin parameter value. To solve this problem, in this paper we fuse SMO with NM and call it SMO-NM. Because slow training is caused by repetitive corrections of the same variables, we modify the working set selection when they are detected. We call the variables that are selected by SMO, SMO variables. At the current step, if a variable selected as an SMO variable was selected in a previous step, we consider that a loop is detected. And in addition to the SMO variables, we add, to the working set, the unbounded variables that were selected as SMO variables and correct the variables by NM. If no loop is detected, the training procedure is the same as that of SMO. As a variant of this working set strategy, we further add violating variables to the working set.

We clarify that if the classification problem is not linearly separable in the feature space, the solutions of L1/L2 SVMs (with the linear sum/square sum of slack variables) are unbounded as the margin parameter value approaches infinity but that, if the mapped training data are not linearly independent in the feature space, the solution of the LS (Least Squares) SVM is unbounded as the margin parameter approaches infinity. We also clarify the condition, in which the increment of the objective function value by SMO-NM is larger than that by SMO.

We evaluate SMO-NM for several benchmark data sets and confirm the effectiveness over SMO especially for a large margin parameter value.

S. Abe
Kobe University, Rokkodai, Nada, Kobe, Japan
Tel.: +81-78-994-3432
Fax: +81-78-994-3432
E-mail: abe@kobe-u.ac.jp

Keywords Least Squares Support Vector Machines · Newton's Method · Sequential Minimal Optimization · Support Vector Machines · Working Set Selection

Symbols

α_i	Lagrange multiplier for $\mathbf{x}_i$
δ_{ij}	delta function with $\delta_{ij} = 1$ for $i = j$ and $\delta_{ij} = 0$ for $i \neq j$
τ	tolerance parameter for convergence
ξ_i	slack variable associated with $\mathbf{x}_i$
A^{-1}	inverse of matrix A
A^T	transpose of matrix A
b	bias term of the hyperplane
b_{low}	lower bound of b that satisfies the KKT conditions
b_{up}	upper bound of b that satisfies the KKT conditions
C	margin parameter
$D(\mathbf{x})$	decision function in the feature space
$\phi(\mathbf{x})$	mapping function from $\mathbf{x}$ to the feature space
γ	parameter for a radial basis function kernel
$K(\mathbf{x}, \mathbf{x}')$	kernel
l	dimension of the feature space
M	number of training data
m	number of input variables
N	indices of the fixed set
Q	objective function
r	scaling factor of corrections
$\ \mathbf{x}\ $	Euclidean norm of vector $\mathbf{x}$
$\mathbf{w}$	coefficient vector of the hyperplane
W	set of the working set indices
W'	set of the working set indices excluding index s included in W
$ W_s $	maximum working set size
$\mathbf{x}_i$	i th m -dimensional training data
y_i	class label 1 or -1 for input $\mathbf{x}_i$

1 Introduction

Since the introduction of support vector machines [1], extensive research on kernel methods including support vector machines [2–6] has been conducted and the results are successfully applied to not only electrical and electronics engineering and computer science fields such as character recognition [7] and diagnosis problems [8] but also to bio-informatics and chemistry [9,10] such as protein analysis [11] and cancer classification [12].

The support vector machine (SVM), which is formulated in a quadratic programming program, is usually trained in dual form, in which the number

of variables equals the number of training data. Therefore, its training speedup is essential for a large size problem, and a number of training methods have been developed [13]. The widely used training methods use decomposition techniques with variable-size chunking or fixed-size chunking.

Sequential minimal optimization (SMO) [14] optimizes two variables at a time and thus uses fixed-size chunking with the working set size of two. Since the introduction of SMO, much effort has been made to improve convergence of SMO. The most important progress is the exact Karush-Kuhn-Tucker (KKT) conditions [15], which exclude the bias term included in the original KKT conditions, but not in the dual form. By selecting the two most violating variables, convergence to the optimal solution is guaranteed [15].

Because the objective function of the SVM is quadratic, the pair of variables that maximize the increase in the objective function value can be selected [16]. The idea of using quadratic information works to improve convergence for a medium margin parameter value around 1000. But it does not work for the value around 1 or much larger than 1000. For a larger value the repetitive corrections of the same variables dramatically increase to reach to the optimal solution. Therefore, convergence slows down. To cope with this situation, in [17] a loop is defined in which the same variable appears in a sequence of selected violating variables. And if it is detected, corrections are made combining the descent directions of variables in the loop. This idea is extended to the introduction of the momentum term [18], which modifies the current corrections by previous-step corrections.

The least squares SVM (LS SVM) can be trained by solving a set of linear equations. But for a large size problem, i.e., a large number of training data, training becomes slow. To accelerate LS SVM training, the conjugate gradient method is applied [19]. The LS SVM can be trained also by SMO. In [19] it is shown that the conjugate gradient method and SMO have comparable training speeds for small and medium margin parameter values but for a larger value, the conjugate gradient method is faster. The second order SMO is also extended to the LS SVM [20] and so is the momentum term [21].

To improve convergence, more than two variables are optimized at a time [22–27]. In [22], q (≥ 2) modifiable variables are selected in the steepest ascent direction and are optimized. In [28], SimpleSVM [23] was compared with LIBSVM [29], which is based on the second order SMO, for the checkers problem. Although SimpleSVM is faster for a large margin parameter value, it is slower for a small one. If the same violating variables appear consecutively or after two iteration steps, in [27], the associated working sets are combined into one and the corrections are calculated by Newton's method. This is the similar idea as that by [17]. The computer experiment showed that for a large margin parameter value, the method was faster than SMO but for a small value, the training speed was comparable or slower.

Incremental and decremental training by [30], which keeps track of the status change among unbounded support vectors, bounded support vectors, and non-support vectors, is extended to batch training [31–36]. These methods

are sometimes called active set training and are considered to be a variant of variable size chunking.

Considering that SMO works well for a small margin parameter value but not for a large one, and that opposite is true for Newton's method (NM) [27], in this paper we propose SMO-NM, which fuses SMO and Newton's method. We call the variables that are selected by SMO, SMO variables. If a variable, which was selected in a previous step as an SMO variable, is selected at the current step, we consider that a loop is detected, and the SMO variables in the previous steps as well as current SMO variables are included in the working set and the corrections are calculated by Newton's method. As an alternative working set selection strategy, in addition to SMO variables we also add violating variables.

We analyze the conditions that the solution becomes unbounded as the margin parameter value approaches infinity for the L1 SVM (with the linear sum of slack variables), the L2 SVM (with the square sum of slack variables), and the LS SVM. And we clarify the difference of convergence between the L1/L2 SVMs and the LS SVM. We also clarify the condition that by SMO-NM the increment of the objective function value is larger than by SMO.

In Section 2, we briefly summarize SVMs and the KKT conditions, and in Section 3 we discuss the proposed training method. In Section 4, we compare SMO-NM with SMO using several benchmark data sets.

2 Support Vector Machines

In this section, according to [13], we briefly summarize the L1 SVM with a linear sum of slack variables, the L2 SVM with a square sum of slack variables, and the LS SVM, with a square sum of slack variables and equality constraints, instead of inequality constraints in the L1 and L2 SVMs.

2.1 L1 SVM

Using the nonlinear vector function $\phi(\mathbf{x})$ that maps the m -dimensional input vector $\mathbf{x}$ into the l -dimensional feature space, we consider determining the decision function that separates Class 1 data from Class 2 data in the feature space:

$$D(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b, \quad (1)$$

where $\mathbf{w}$ is the l -dimensional vector and b is the bias term. Then the L1 and L2 SVMs are formulated in the primal form as follows:

$$\begin{aligned} \text{minimize} \quad & Q(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2} \mathbf{w}^\top \mathbf{w} + \frac{C}{p} \sum_{i=1}^M \xi_i^p \end{aligned} \quad (2)$$

$$\text{subject to} \quad y_i D(\mathbf{x}_i) \geq 1 - \xi_i \quad \text{for } i = 1, \dots, M, \quad (3)$$

where $p = 1$ for the L1 SVM and $p = 2$ for the L2 SVM, C is the margin parameter that controls the trade-off between the training error and the generalization ability, $\mathbf{x}_i$ are M m -dimensional training inputs and belong to Class 1 or 2 and the associated labels are $y_i = 1$ for Class 1 and -1 for Class 2, and $\xi_i (\geq 0)$ are the slack variables for $\mathbf{x}_i$.

The dual form of the L1 SVM is given by

$$\text{maximize} \quad Q(\boldsymbol{\alpha}) = \sum_{i=1}^M \alpha_i - \frac{1}{2} \sum_{i,j=1}^M \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (4)$$

$$\text{subject to} \quad \sum_{i=1}^M y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C \quad \text{for } i = 1, \dots, M, \quad (5)$$

where α_i are Lagrange multipliers associated with $\mathbf{x}_i$ and $K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}^\top(\mathbf{x}) \boldsymbol{\phi}(\mathbf{x}')$ is a kernel function. Among several kernels, polynomial kernels and radial basis function (RBF) kernels are often used for pattern classification, but because in most cases RBF kernels perform better [13], we use RBF kernels:

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2 / m), \quad (6)$$

where m is the number of inputs for normalization and γ is a spread of a radius.

The KKT complementarity conditions are given by

$$\alpha_i \left(\sum_{j=1}^M y_i y_j \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + y_i b - 1 + \xi_i \right) = 0 \quad \text{for } i = 1, \dots, M, \quad (7)$$

$$(C - \alpha_i) \xi_i = 0 \quad \alpha_i \geq 0, \quad \xi_i \geq 0 \quad \text{for } i = 1, \dots, M. \quad (8)$$

For the solution of (4) and (5), if $\alpha_i > 0$, $\mathbf{x}_i$ are called support vectors; especially if $\alpha_i = C$, bounded support vectors and if $0 < \alpha_i < C$, unbounded support vectors.

To avoid using the bias term we use exact KKT conditions proposed by [15]. We rewrite (7) as follows:

$$\alpha_i (y_i b - y_i F_i + \xi_i) = 0, \quad (9)$$

where

$$F_i = y_i - \sum_{j=1}^M y_j \alpha_j K(\mathbf{x}_i, \mathbf{x}_j). \quad (10)$$

Then the KKT conditions are simplified as follows:

$$b_{\text{up}} \geq b \geq b_{\text{low}}, \quad (11)$$

where

$$b_{\text{low}} = \max_{i=1, \dots, M} \tilde{F}_i, \quad (12)$$

$$b_{\text{up}} = \min_{i=1, \dots, M} \bar{F}_i, \quad (13)$$

$$\tilde{F}_i = F_i \quad \text{if } (y_i = 1, \alpha_i = 0), \quad 0 < \alpha_i < C \quad \text{or} \quad (y_i = -1, \alpha_i = C), \quad (14)$$

$$\bar{F}_i = F_i \quad \text{if } (y_i = -1, \alpha_i = 0), \quad 0 < \alpha_i < C \quad \text{or} \quad (y_i = 1, \alpha_i = C). \quad (15)$$

To ease convergence, we consider that the KKT conditions are satisfied when

$$b_{\text{up}} \geq b_{\text{low}} - \tau \quad (16)$$

is satisfied, where τ is a positive tolerance parameter.

The set of indices, V_{KKT} , of violating variables is given by

$$V_{\text{KKT}} = \{i \mid b_{\text{up}} < \tilde{F}_i - \tau \quad \text{or} \quad b_{\text{low}} > \bar{F}_i + \tau \quad \text{for} \quad i \in \{1, \dots, M\}\}. \quad (17)$$

2.2 L2 SVM

Setting $p = 2$ in (2), we obtain the L2 SVM. Its dual form is given by

$$\text{maximize} \quad Q(\alpha) = \sum_{i=1}^M \alpha_i - \frac{1}{2} \sum_{i,j=1}^M y_i y_j \alpha_i \alpha_j \left(K(\mathbf{x}_i, \mathbf{x}_j) + \frac{\delta_{ij}}{C} \right) \quad (18)$$

$$\text{subject to} \quad \sum_{i=1}^M y_i \alpha_i = 0, \quad \alpha_i \geq 0 \quad \text{for} \quad i = 1, \dots, M, \quad (19)$$

where $\delta_{ij} = 1$ for $i = j$ and $\delta_{ij} = 0$ for $i \neq j$.

For the L1 SVM, if we replace $K(\mathbf{x}_i, \mathbf{x}_j)$ with $K(\mathbf{x}_i, \mathbf{x}_j) + \delta_{ij}/C$ and remove the upper bound given by C for α_i , we obtain the L2 SVM.

Because $1/C$ is added to the diagonal elements of the matrix $K = \{K(\mathbf{x}_i, \mathbf{x}_j)\}$ called the *kernel matrix*, the resulting matrix becomes positive definite.

For the L2 SVM, the exact KKT conditions are as follows:

$$b_{\text{up}} \geq b_{\text{low}}, \quad (20)$$

where

$$b_{\text{low}} = \max_{i=1, \dots, M} \tilde{F}_i, \quad (21)$$

$$b_{\text{up}} = \min_{i=1, \dots, M} \bar{F}_i, \quad (22)$$

$$\tilde{F}_i = F_i - \frac{y_i \alpha_i}{C} \quad \text{if} \quad (y_i = 1, \alpha_i = 0) \quad \text{or} \quad \alpha_i > 0, \quad (23)$$

$$\bar{F}_i = F_i - \frac{y_i \alpha_i}{C} \quad \text{if} \quad (y_i = -1, \alpha_i = 0) \quad \text{or} \quad \alpha_i > 0. \quad (24)$$

2.3 LS SVM

The LS SVM is defined by replacing the inequality constraints of the L2 SVM in (3) with the equality constraints and can be trained by solving a set of linear equations. But because the matrix size is the number of training data plus 1 and because the complexity of matrix inversion is the cubic order of the matrix size, training becomes extremely slow for a large data set. To alleviate this problem, SMO is extended to training LS SVMs [20, 21].

The dual form of the LS SVM is as follows:

$$\text{maximize } Q(\boldsymbol{\alpha}) = \sum_{i=1}^M \alpha_i y_i - \frac{1}{2} \sum_{i,j=1}^M \alpha_i \alpha_j \left(K(\mathbf{x}_i, \mathbf{x}_j) + \frac{\delta_{ij}}{C} \right) \quad (25)$$

$$\text{subject to } \sum_{i=1}^M \alpha_i = 0. \quad (26)$$

The KKT conditions of the above problem is given by

$$\alpha_i \left(\sum_{j=1}^M \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + \alpha_i / C + b - y_i \right) = 0 \quad \text{for } i = 1, \dots, M. \quad (27)$$

We define

$$F_i = y_i - \sum_{j=1}^M \alpha_j K(\mathbf{x}_i, \mathbf{x}_j). \quad (28)$$

Then, (27) becomes

$$\alpha_i (b - F_i + \alpha_i / C) = 0 \quad \text{for } i = 1, \dots, M. \quad (29)$$

Because of the equality constraints in the primal form, we can assume that irrespective of α_i the following conditions are satisfied for the optimal solution:

$$b - F_i + \alpha_i / C = 0 \quad \text{for } i = 1, \dots, M. \quad (30)$$

Then the KKT conditions are satisfied when

$$b_{\text{up}} \geq b_{\text{low}}, \quad (31)$$

where

$$b_{\text{low}} = \max_{i=1, \dots, M} (F_i - \alpha_i / C), \quad (32)$$

$$b_{\text{up}} = \min_{i=1, \dots, M} (F_i - \alpha_i / C). \quad (33)$$

In training, we use the stopping condition (16).

3 Training Methods

In this section we discuss corrections of variables by Newton's method and the working set selection strategies.

3.1 Calculating Corrections by Newton's Method

First, we discuss corrections of variables for the L1 SVM and then for the L1 and LS SVMs.

3.1.1 Training L1 SVMs

We optimize the variables α_i ($i \in W$) fixing α_i ($i \in N$), where $W \cup N = \{1, \dots, M\}$ and $W \cap N = \phi$, by

$$\begin{aligned} \text{maximize} \quad Q(\alpha_W) &= \sum_{i \in W} \alpha_i - \frac{1}{2} \sum_{i,j \in W} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ &\quad - \sum_{\substack{i \in W, \\ j \in N}} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \quad (34)$$

$$\text{subject to} \quad \sum_{i \in W} y_i \alpha_i = - \sum_{i \in N} y_i \alpha_i, \quad 0 \leq \alpha_i \leq C \quad \text{for } i \in W. \quad (35)$$

Here $\alpha_W = (\dots, \alpha_i, \dots)^\top$, $i \in W$.

Solving the equality in (35) for α_s ($s \in W$), we obtain

$$\alpha_s = - \sum_{i \neq s, i=1}^M y_s y_i \alpha_i. \quad (36)$$

Substituting (36) into (34), we eliminate the equality constraint. Let $\alpha_{W'} = (\dots, \alpha_i, \dots)^\top$ ($i \neq s, i \in W$). Now because $Q(\alpha_{W'})$ is quadratic, we can express the change of $Q(\alpha_{W'})$, $\Delta Q(\alpha_{W'})$, as a function of the change of $\alpha_{W'}$, $\Delta \alpha_{W'}$, by

$$\Delta Q(\alpha_{W'}) = \frac{\partial Q(\alpha_{W'})}{\partial \alpha_{W'}} \Delta \alpha_{W'} + \frac{1}{2} \Delta \alpha_{W'}^\top \frac{\partial^2 Q(\alpha_{W'})}{\partial \alpha_{W'}^2} \Delta \alpha_{W'}. \quad (37)$$

Then, neglecting the bounds, $\Delta Q(\alpha_{W'})$ has the maximum at

$$\Delta \alpha_{W'} = - \left(\frac{\partial^2 Q(\alpha)}{\partial \alpha_{W'}^2} \right)^{-1} \frac{\partial Q(\alpha_{W'})}{\partial \alpha_{W'}}, \quad (38)$$

where

$$\frac{\partial Q(\alpha_{W'})}{\partial \alpha_i} = y_i (F_i - F_s) \quad \text{for } i \in W', \quad (39)$$

$$\begin{aligned} \frac{\partial^2 Q(\alpha_{W'})}{\partial \alpha_i \partial \alpha_j} &= y_i y_j (-K(\mathbf{x}_i, \mathbf{x}_j) + K(\mathbf{x}_i, \mathbf{x}_s) + K(\mathbf{x}_s, \mathbf{x}_j) - K(\mathbf{x}_s, \mathbf{x}_s)) \\ &\quad \text{for } i, j \in W'. \end{aligned} \quad (40)$$

Here, we assume that $-\partial^2 Q(\boldsymbol{\alpha})/\partial \boldsymbol{\alpha}_{W'}^2$ is positive definite. The procedure when the matrix is positive semi-definite is discussed in Section 3.3.

Then from (35) and (38), we obtain the correction of α_s :

$$\Delta \alpha_s = - \sum_{i \in W'} y_s y_i \Delta \alpha_i. \quad (41)$$

For α_i ($i \in W$), if

$$\alpha_i = 0, \quad \Delta \alpha_i < 0 \quad \text{or} \quad \alpha_i = C, \quad \Delta \alpha_i > 0, \quad (42)$$

we delete these variables from the working set and repeat the procedure for the reduced working set. Let $\Delta \alpha'_i$ be the maximum or minimum correction of α_i that is within the bounds. Then if $\alpha_i + \Delta \alpha_i < 0$, $\Delta \alpha'_i = -\alpha_i$. And if $\alpha_i + \Delta \alpha_i > C$, $\Delta \alpha'_i = C - \alpha_i$. Otherwise $\Delta \alpha'_i = \Delta \alpha_i$. Then we calculate

$$r = \min_{i \in W} \left| \frac{\Delta \alpha'_i}{\Delta \alpha_i} \right|, \quad (43)$$

where r ($0 < r \leq 1$) is the scaling factor.

The corrections of the variables in the working set are given by

$$\boldsymbol{\alpha}_W^{\text{new}} = \boldsymbol{\alpha}_W^{\text{old}} + r \Delta \boldsymbol{\alpha}_W. \quad (44)$$

3.1.2 Training L2 SVMs

The training method for the L2 SVM is similar to that for L1 SVM. In the previous discussion, we replace (39) and (40), respectively, with

$$\frac{\partial Q(\boldsymbol{\alpha}_{W'})}{\partial \alpha_i} = y_i (F_i - F_s) - \alpha_i y_i / C + \alpha_s y_s / C \quad \text{for } i \in W', \quad (45)$$

$$\frac{\partial^2 Q(\boldsymbol{\alpha}_{W'})}{\partial \alpha_i \partial \alpha_j} = y_i y_j (-K(\mathbf{x}_i, \mathbf{x}_j) + K(\mathbf{x}_i, \mathbf{x}_s) + K(\mathbf{x}_s, \mathbf{x}_j) - K(\mathbf{x}_s, \mathbf{x}_s)) - 2 \delta_{ij} / C$$

for $i, j \in W'$. (46)

And we replace (42) with

$$\alpha_i = 0, \quad \Delta \alpha_i < 0. \quad (47)$$

The change $\Delta \alpha'_i$ in (43) is given as follows: If $\alpha_i + \Delta \alpha_i < 0$, $\Delta \alpha'_i = -\alpha_i$. Otherwise $\Delta \alpha'_i = \Delta \alpha_i$.

In the L2 SVM, because $1/C$ is added to the diagonal elements of the kernel matrix, $-\partial^2 Q(\boldsymbol{\alpha})/\partial \boldsymbol{\alpha}_{W'}^2$ is positive definite.

3.1.3 Training LS SVMs

Because the dual form of the LS SVM is defined by a quadratic problem with one equality constraint, the training method is further simplified.

In the previous discussions, we replace the partial derivatives of $Q(\alpha_{W'})$ by

$$\frac{\partial Q(\alpha_{W'})}{\partial \alpha_i} = F_i - F_s - \alpha_i/C + \alpha_s/C \quad \text{for } i \in W', \quad (48)$$

$$\frac{\partial^2 Q(\alpha_{W'})}{\partial \alpha_i \partial \alpha_j} = -K(\mathbf{x}_i, \mathbf{x}_j) + K(\mathbf{x}_i, \mathbf{x}_s) + K(\mathbf{x}_s, \mathbf{x}_j) - K(\mathbf{x}_s, \mathbf{x}_s) - 2\delta_{ij}/C$$

for $i, j \in W'$. (49)

Because $-\partial^2 Q(\alpha)/\partial \alpha_{W'}^2$ is positive definite and there are no inequality constraints, $r = 1$ and the corrections are always possible. In the extreme case where $|W| = M - 1$, the solution is obtained in just one step without iterations.

3.2 Working Set Selection

Let the variable pair associated with $\min \bar{F}_i$ and $\max \tilde{F}_i$ be $\alpha_{i_{\min}}$ and $\alpha_{i_{\max}}$, respectively.¹ In the first order SMO, at each iteration step, $\alpha_{i_{\min}}$ and $\alpha_{i_{\max}}$ are selected for optimization. This guarantees the convergence of the first order SMO [15]. In the second order SMO, to reduce computational burden, in searching the optimal pair of variables, fixing $\alpha_{i_{\min}}$, the variable that maximizes the objective function is searched [16]:

$$i_{2\text{nd}} = \arg \max_{i \in V_{\text{KKT}}} \Delta Q(\alpha_i, \alpha_{i_{\min}}). \quad (50)$$

We call the pair of variables that are determined either by the first or the second order SMO, *SMO variables*.

For the small value of C around 1, there is not much gain using the second order SMO but for the medium value of C around 1000, the second order SMO is faster than the first order SMO [17]. Therefore, in the following we use the second order SMO. For a large C value, training speed of the second order SMO slows down because variables need to be updated many times to reach to the optimal values. To speed up convergence in such a situation, we consider adding variables into the working set in addition to the SMO variables. These variables are variables selected in the previous steps as SMO variables.

We consider that a *loop* is detected when at least one of the current SMO variables has already appeared as an SMO variable at a previous step. When a loop is detected, we pick up the *loop variables* that are the SMO variables in the one step to l_c steps prior to the current step, where l_c is a user-defined parameter and we call the detected loop, *l_c -cycle loop*. To avoid obtaining an infeasible solution by adding loop variables to the working set, we restrict loop

¹ In the LS SVM, F_i are used instead of $\bar{F}_i$ and $\tilde{F}_i$.

variables to be unbounded support vectors for the L1 SVM and support vectors for the L2 SVM. But for the LS SVM, any variables are selected because they are unconstrained.

Let $|W_s|$ denote the maximum working set size. Then we set $|W_s| = 2l_c + 2$.

We then propose the following three working set selection strategies:

1. Loop variable (LV) selection: If no loop is detected, the working set includes only the SMO variables. If a loop is detected, the working set consists of the SMO variables and the loop variables.
2. Loop and violating variable (L&V) selection: In the loop variable selection, if no loop is detected, or if the number of loop variables is smaller than $2l_c$, we add the violating variables to the working set so that the number of the loop variables and the violating variables is $2l_c$. (There may be cases where violating variables are not sufficient to fill the working set.)
3. Violating variable (VV) selection: We select the violating variables from the most violating to the least in order. In this case also the SMO variables are included in the working set.

Violating variable selection does not include loop variables. Thus it is a variant of the previous method and is used to evaluate the effectiveness of loop detection by LV and L&V strategies in improving the convergence of training. The improvement by these strategies are shown in Section 5.

In the following we show the procedure of LV selection for the L1 SVM more in detail.

At the start of training, we initialize $status(i) = 0$ for $i = 1, \dots, M$, where $status(i) = 0$ for α_i not being selected as an SMO variable, and $status(i) = 1$, already being selected, and $ptr = 0$, where ptr is the pointer of the first-in last-out stack *filo* with the stack size of $|W_s|$. At each iteration step, after $i_{\min}$ and i_{2nd} are calculated, we do the following.

Loop detection and working set selection

1. (Loop detection) If $status(i_{2nd}) = 1$ or $status(i_{\min}) = 1$, then a loop is detected and go to 2. Else, $status(i_{2nd}) = 1$, $status(i_{\min}) = 1$, $filo \leftarrow \{i_{\min}, i_{2nd}\}$, $ptr \leftarrow ptr + 2$, $W_1 = i_{\min}$, and $W_2 = i_{2nd}$, and exit.
2. (Working set setting) $W_1 = i_{\min}$, $W_2 = i_{2nd}$, and $k = 1$.
do $j = ptr, 1, -1$
if $filo(j) \notin W$ and $0 < \alpha_{filo(j)} < C$, then $k \leftarrow k + 1$, $W_k = filo(j)$
end do
Set $status(i_{2nd}) = 1$ and $status(i_{\min}) = 1$, $ptr = 0$, and $filo \leftarrow \{i_{\min}, i_{2nd}\}$ and exit.

In Step 2, the condition of $filo(j) \notin W$ is to avoid duplicate indices in the working set and the condition $0 < \alpha_{filo(j)} < C$ is to avoid obtaining an infeasible solution. For the L2 SVM, the condition is changed to $C > 0$ and for the LS SVM, no condition is imposed on $\alpha_{filo(j)}$.

If the procedure is exited at Step 1, $|W| = 2$ and the corrections of variables are calculated according to the SMO procedure. And if it is exited at Step 2 the corrections are calculated by Newton's method.

For L&V selection, the violating variables are added in Steps 1 and 2 in the above procedure to fill the working set W . For VV selection, without loop detection, after $i_{\min}$ and $i_{2\text{nd}}$ are calculated, $W_1 = i_{\min}$, $W_2 = i_{2\text{nd}}$, and violating variables are added to W .

The advantage of the LV selection over L&V or VV selection is that the working set size $|W|$ is determined automatically according to whether loop variables exist. Thus, the overhead caused by matrix inversion is reduced.

3.3 Calculating Corrections by Cholesky Factorization

We use the Cholesky factorization in calculating (38). If the infeasible solution is obtained or the matrix becomes singular during factorization, we can obtain a feasible solution using the factorized matrices as explained in this section.

We set $\alpha_s = \alpha_{i_{\min}}$, which is the first element of W and $W' = W - \{i_{\min}\}$. Let $K = \{K_{ij}\} = -\partial^2 Q(\alpha) / \partial \alpha_{W'}^2$ ($i, j = 1, \dots, |W'|$). Here, the set $\{1, \dots, |W'|\}$ is a subset of V_{KKT} and the elements are renumbered from 1 to $|W'|$ and 1 corresponds to $i_{2\text{nd}}$. If K is positive definite, it is decomposed by the Cholesky factorization into

$$K = L L^\top, \quad (51)$$

where L is the regular lower triangular matrix and each element L_{ij} is given by

$$L_{op} = \frac{K_{op} - \sum_{n=1}^{p-1} L_{pn} L_{on}}{L_{pp}} \quad \text{for } o = 1, \dots, |W'|, \quad p = 1, \dots, o-1, \quad (52)$$

$$L_{aa} = \sqrt{K_{aa} - \sum_{n=1}^{a-1} L_{an}^2} \quad \text{for } a = 1, 2, \dots, |W'|. \quad (53)$$

Then during the Cholesky factorization, if the argument of the square root associated with the diagonal element is smaller than the prescribed value $\eta (> 0)$:

$$H_{aa} - \sum_{n=1}^{a-1} L_{an}^2 \leq \eta, \quad (54)$$

we stop factorizing the matrix and use the already-factorized matrices to obtain the corrections. This happens for the L1 SVM and for the L2, LS SVMs with extremely large C value. Otherwise, we use the full L to obtain the corrections.

For the L1 and L2 SVMs, we check whether the corrections satisfy the inequality constraints. If some of the variables do not satisfy the constraints, we recalculate the corrections, deleting the rows in the L after the rows associated with the variables that violate the inequality constraints. We repeat this

procedure until the feasible corrections are obtained (i.e., $r > 0$). The above procedure is done using the matrices factorized so far. For the LS SVM, $r = 1$.

Because we select the SMO variables as α_s and the first variable in W' , the first diagonal element of $-\partial^2 Q(\boldsymbol{\alpha})/\partial \boldsymbol{\alpha}_{W'}^2$ is non-zero and the SMO variables give the feasible solution. Therefore, for the L1/L2 SVMs, SMO-NM reduces to SMO, at worst.

The Cholesky factorization requires $|W'|^3/3$ floating operations [37] compared to one division for SMO. Therefore, to speed up training using Cholesky factorization over SMO, enough reduction of the number of iterations is necessary.

3.4 Training Procedure of SMO-NM

In the following we show the training procedure of SMO-NM for the L1 SVM using the LV selection strategy.

1. (Initialization) Set an appropriate value to l_c . Select α_i and α_j in the opposite classes and set $\alpha_i = \alpha_j = C$ and $\alpha_k = 0, k \neq i, j, k = 1, \dots, M$.
2. (Corrections) Calculate partial derivatives (39) and (40) and calculate corrections by (38). Then, modify the variables by (44).
3. (Convergence Check) Update F_i and calculate b_{up} and b_{low} . If (16) is satisfied, stop training. Otherwise calculate $i_{2\text{nd}}$ by (50).
4. (Loop detection and working set selection) Do loop detection and working set selection shown in the previous section and go to Step 2.

In Step 2, if SMO-NM reduces to SMO, (38) reduces to scalar division.

In Step 3, we calculate F_i incrementally if the current working set size is smaller than the number of support vectors. Otherwise, they are calculated from scratch.

4 Characteristics of Solutions

In this section, we clarify the dependence of solutions of the L1, L2, and LS SVMs on the C value. Then we discuss convergence of SMO-NM.

As the value of C approaches infinity, the solution of the L1 SVM approaches that of the L2 SVM [13]. And we can prove the following theorem:

Theorem 1 *If the classification problem is not linearly separable in the feature space, for $C = \infty$, the dual solutions of the L1 and L2 SVMs are unbounded.*

Proof. If the classification problem is not linearly separable in the feature space, bounded support vectors exist in the solution of the L1 SVM, i.e., $\alpha_i = C$ for some $i \in \{1, \dots, M\}$. Therefore, as the value of C approaches ∞ , the dual variables associated with the bounded support vectors also approach ∞ . Therefore, for $C = \infty$, the dual solution of the L1 SVM is unbounded. For $C = \infty$, L1 and L2 SVMs are equivalent. Thus the dual solution of the L2 SVM is also unbounded.

When the classification problem is linearly separable in the feature space, for the L1 SVM, there exists a C value, $C_{\max}$, in which the support vectors are all unbounded for $C > C_{\max}$. Therefore the solution does not change for $C > C_{\max}$. Although the solution of the L2 SVM changes as the value of C is increased, it approaches that of the L1 SVM as C approaches ∞ .

For the LS SVM, linear separability is not related to the unbounded dual solution as the following theorem shows:

Theorem 2 *For $C = \infty$, the dual solution of the LS SVM is unbounded if the kernel matrix is singular, namely if the mapped training data are not linearly independent.*

Proof. If $C = \infty$, $1/C = 0$. Then, if the kernel matrix is singular, the solution is unbounded. The rank of the kernel matrix is equivalent to the number of linearly independent mapped training data in the feature space [38]. Thus the theorem holds.

For the linear kernel, if the number of training data exceeds that of the input variables, the kernel matrix is singular. Therefore, the dual solution of the LS SVM is unbounded for $C = \infty$, irrespective of linear separability. Even for a non-linear kernel, except for the small sample problems, the kernel matrix is usually positive semi-definite. Therefore, unlike the L1, L2 SVMs, in most cases the dual solution of the LS SVM goes $\pm\infty$ as C approaches ∞ . This makes the iterative methods for training the LS SVM more inefficient than those for the L1 and L2 SVMs.

Now we investigate the effect of SMO-NM over SMO.

For the L1 and L2 SVMs, the following Theorem holds.

Theorem 3 *For the working set W , the increase of the objective function value, $\Delta Q(\alpha_{W'})$, is given by*

$$\Delta Q(\alpha_{W'}) = -\frac{1}{2}r_W(2 - r_W) \frac{\partial Q(\alpha_{W'})}{\partial \alpha_{W'}}^\top \left(\frac{\partial^2 Q(\alpha_{W'})}{\partial \alpha_{W'}^2} \right)^{-1} \frac{\partial Q(\alpha_{W'})}{\partial \alpha_{W'}} \geq 0, \quad (55)$$

where r_W is the scaling factor for W . Then if $r_W \geq r_{W_S}$,

$$\Delta Q(\alpha_{W'}) \geq \Delta Q(\alpha_{W'_S}) \quad (56)$$

is satisfied, where $W \supset W_S$. The strict inequality holds when some values of $\alpha_i \in W - W_S$ are not equal to those of the optimal solution for the working set W .

Proof. By the corrections $r \Delta \alpha_W$, the increment of the objective function value, $\Delta \alpha_{W'}$ is given by substituting $r \Delta \alpha_W$ and (38) into (37). Thus, we obtain (55).

When W_S is used, $\alpha_i \in W - W_S$ are not optimized. Therefore, (56) holds. The strict inequality holds if some values of $\alpha_i \in W - W_S$ are not equal to those of the optimal solution for the working set W . Thus the theorem holds.

For the LS SVM, $r = 1$, irrespective of the working set size. Therefore, (56) holds for any working set size. This means that the number of iteration by SMO-NM is smaller than or equal to that by SMO.

For the L1, L2 SVMs, $r_W \geq r_{W_S}$ is not always guaranteed. Therefore, there may be cases where the number of iterations by SMO is smaller than by SMO-NM. In such cases, training time by SMO-NM may be longer than by SMO due to the overhead of multiple-variable processing.

In any case, because by SMO the SMO variables, which improve the objective function value most, are selected, by variable corrections the objective function value increases monotonically. By SMO-NM, if the working set with the size larger than two does not give a solution, it reduces to two. Therefore, irrespective of the working set size, the objective function value increases monotonically. Thus, by SMO-NM, convergence to the optimal solution is guaranteed.

5 Performance Evaluation

In this section we show by computer experiments that the theoretical results obtained in Section 4 are valid; the large C value is sometimes necessary to realize high generalization ability; and the proposed working set selection strategies can improve convergence of Newton’s method for a small C values and that of SMO for a large C value.

For these purposes we use the following 8 benchmark data sets: blood-cell [13], thyroid [39], hiragana-50 [13], hiragana-13 [13], hiragana-105 [13], satimage [39], USPS (United States Postal Service) [40], and letter [39] data sets. We combined the training and test data sets, if divided, into one data set to test the training time for large data sets. The benchmark data sets were all multiclass classification problems and we used fuzzy pairwise SVMs [13] and measured the training time using a personal computer (3GHz, 2GB memory, Windows XP operating system). If training time was shorter than 100 s, we measured training time five times and took the average. We prepared a cache memory with the size equal to the kernel matrix, if possible.

Each input of the training data was linearly scaled into $[0, 1]$. We set $\eta = 10^{-9}$ so that matrix singularity does not happen frequently and $\tau = 0.001$, which is widely used as the valid parameter value for convergence [16].

5.1 Convergence Comparison between L1, L2, and LS SVMs

In this section, we evaluate the theoretical results discussed in Section 4 using the classes 2 and 3 data in the original blood cell training data set, which includes 398 data samples with 13 inputs. As a case where the blood cell data set is not linearly separable, we used linear kernels: $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$, and as a linearly separable case, the RBF kernel given by (6) with $\gamma = 1$. Table 1 shows the results for large C values. In the table, “SVs,” “ α_{199} ,” and “Rec”

Table 1 Convergence comparison of SVMs for the blood cell data

Kernel	C	SVs			α_{199}			Rec (%)		
		L1	L2	LS	L1	L2	LS	L1	L2	LS
Linear	10^6	102 (88)	185	398	1000000	660046	-666650	90.20	90.20	90.20
	10^9	102 (88)	185	398	1000000000	660046328	-666653213	90.20	90.20	90.20
$\gamma = 1$	10^6	103 (8)	109	398	1000000	396626	-116513	99.25	99.50	96.48
	10^9	102	102	398	786675	785806	-156254721	100.00	100.00	100.00
	10^{12}	102	102	398	786675	786674	-260707322	100.00	100.00	100.00
	10^{15}	102	102	398	786675	786675	-260901917	100.00	100.00	100.00

denote the number of support vectors (the numerals in the L1 SVM column are the numbers of bounded support vectors), the value of α_{199} , which is the 199th Lagrange multiplier, and recognition rate of the training data in %, respectively.

For the linear kernel, the recognition rates of the three types of SVMs were the same for $C = 10^6, 10^9$. This means that as the value of C approaches ∞ , although the values of α_i increase, the coefficient vectors of the hyperplane converge to constant vectors [13]. The number of support vectors for the LS SVM is 398, which is the number of training data.

For the RBF kernel, the recognition rates reached 100% for values of C larger than 10^6 , and the solutions of the L1 SVM were the same. The solution of the L2 SVM approached that of the L1 SVM as the value of C was increased and the values of α_{199} were the same for $C = 10^{15}$. But the absolute values of α_{199} increased for the LS SVM.

The above results confirmed the validity of the results obtained in Section 4.

5.2 Necessity of Large C Values

According to the footnote of [19], it is discussed that the C value greater than 10^3 seldom occurs in practice. To confirm this, we performed fivefold cross-validation for the benchmark data sets used. We also did for the original training data sets and evaluated the recognition rates of the test data sets. We broke the tie by selecting the smaller C value for the same recognition rates for the validation data set. Table 2 shows the results. In the table, “ C ” and “CV Rec” in the second and third columns denote the margin parameter value determined by cross-validation and the associated recognition rate, respectively. The last three columns denote the results for the original training and test data sets, and “ C ,” “CV Rec” are those using the training data set and “Test Rec” denotes the recognition rate for the test data set. Using the combined training and test data set, except for the hiragana-105 and USPS data sets, which are relatively easily separated, the C values exceeded 10^3 . Except for the thyroid and USPS data sets, the C values for the divided training and test data sets were smaller than for the combined data sets, but the C values for four data sets were still larger than 1000.

Table 2 Cross-validation results using L1 SVM

Data	C	CV Rec (%)	Divided train & test data		
			C	CV Rec (%)	Test Rec (%)
Blood	7000	95.03	500	94.51	92.77
Thyroid	1000000	98.07	1000000	97.99	97.87
Hiragana-50	2000	99.79	500	99.39	98.83
Hiragana-13	100000	99.80	2000	99.46	99.64
Hiragana-105	100	99.98	500	99.67	100.00
Satimage	10000	90.61	3000	89.76	88.65
USPS	100	96.76	100	97.22	94.12
Letter	100000	95.78	100000	95.00	95.35

Of course, if we chose a value from several choices for the γ value, the selected C value might be lowered. But at least we need to know that the C value larger than 10^3 is not so rare.

5.3 Parameter Selection

Usually model selection of the SVM, i.e., determination of the values of C and γ is done by cross-validation. And according to the evaluation in Section 5.2, for a given classification problem, it is better to carry out cross-validation including a large C value. Therefore, in our experiment, we measured the training time for $C = 1, 1000, 10^6, 10^9$ for L1 and L2 SVMs but for LS SVM we excluded $C = 10^9$, because training was very slow.

Training time is also affected by the selection of the γ value. But here we set $\gamma = 1$ in (6), which is a default value in LIBSVM.

The optimal values for C and γ may be different from the above values, but at least we need to carry out cross-validation including $\gamma = 1$ and large C values. Therefore, it is better to compare the training time of different methods for the above grid points than to compare time only at the cross-validated C and γ values.

To use the proposed working set selection strategies, we need to set the maximum working set size $|W_s|$. Because the training behaviors for a small C value and a large C value are different, for the L1 and L2 SVMs, we determined the maximum working set sizes for $C = 1$ and $C = 10^9$ and used the size for $C = 1$ to those for $C = 1, 1000$ and the size for $C = 10^9$ to those for $C = 10^6, 10^9$. But because training of the LS SVM is very slow, we measured time for $C = 1, 1000, 10^6$, determined the maximum working set sizes for $C = 1000, 10^6$, and used the size for $C = 1000$ also to that for $C = 1$. As the benchmark data set, we used classes 2 and 3 data in the original blood cell training data set and the determined sizes were used for all the benchmark data sets.

In the following we show the results of the working set sizes for the L1, L2, and LS SVMs.

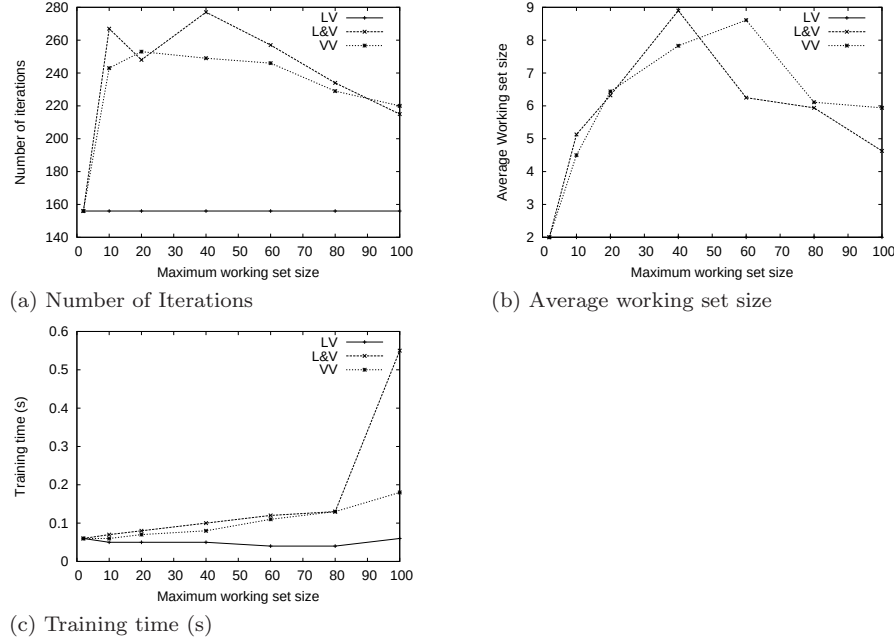


Fig. 1 Comparison of selection strategies using the L1 SVM for the blood cell data with $C = 1$

5.3.1 L1 SVM

We changed the $|W_s|$ value from 2 to 100 for $C = 1$ and examined, for the three working set strategies, the number of iterations for convergence, the average working set size, and training time.

Figure 1 shows the result. For $|W_s| = 2$, the three strategies reduce to the second order SMO. In Fig. 1 (a) the numbers of iterations for the LV selection are the same for different working set sizes. This is because the loop was not detected and the average working set size was two, irrespective of the $|W_s|$ values as seen from Fig. 1 (b). Therefore, from Fig. 1 (c), the training time was almost constant. But for L&V and VV selection strategies, the numbers of iterations were larger than that for the SMO and training was slower than that by the LV selection.

Figure 2 shows the change of objective function values during iterations for the three selection strategies. Although the LV selection converged fastest, the behaviors of the three strategies are similar.

From the above results, for $C = 1$, the LV selection strategy worked best and the large $|W_s|$ value did not deteriorate the convergence speed.

Figure 3 shows the behaviors of the selection strategies for $C = 10^9$. In Fig. 3 (a), by the second order SMO (i.e., $|W_s| = 2$), the number of iterations increased drastically and the training time was much longer than that for $C = 1$ (Fig. 3 (c)). As we increased the $|W_s|$ value, the average working set

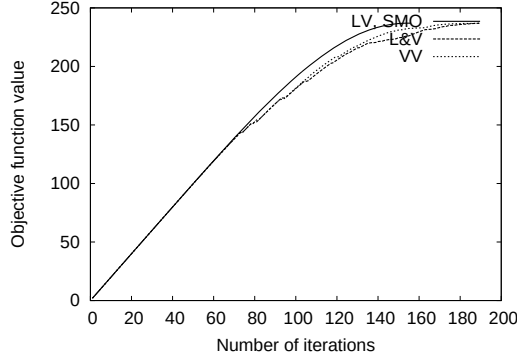


Fig. 2 Objective function values for $C = 1$

size also increased (Fig. 3 (b)), and the numbers of iterations by the LV and L&V strategies decreased drastically, but by the VV strategy, the number of iterations remained large (Fig. 3 (a)). This shows that the introduction of loop variables remarkably decreased the number of iterations and thus sped up training. Especially, the LV strategy showed the best performance.

Figure 4 shows the change of the objective function values during training for $W_s = 600$. Before 200 iterations, training using the LV and L&V strategies converged but the objection function values for the SMO and the VV strategy were very far away from that of the optimal solution.

According to the results for $C = 1$ and $C = 10^9$, we used the LV selection strategy and set $|W_s| = 600$ (the number of cycles = 298).

5.3.2 L2 SVM

Figure 5 shows the behaviors of the selection strategies for $C = 1$. The differences of the results for the L1 and L2 SVMs are clear by comparing Fig. 5 with Fig. 1. For the L2 SVM, the numbers of iterations and the average working set sizes increased (Figs. 5 (a), (b)). This led to slower training (Fig. 5 (c)). The main reason is the lack of the upper bound C for the variables. This made it possible to solve the solution with larger working set sizes.

The shortest training time was achieved by the VV selection strategy with $|W_s| = 10$ but the time difference between VV and LV selection strategies for $|W_s| = 10$ is small. In addition, the training time did not change much for the LV selection strategy. Thus, we selected the LV selection strategy with $|W_s| = 10$.

Figure 6 shows the behaviors of the selection strategies for $C = 10^9$, which are very similar to those shown in Figure 3. This is consistent with the results in Section 5.1. For $C = 10^9$, the LV selection strategy with $|W_s| = 600$ is the best.

According to the above discussions, we set $|W| = 10$ for $C = 1, 1000$ and $|W| = 600$ for $C = 10^6, 10^9$, and trained the L2 SVM.

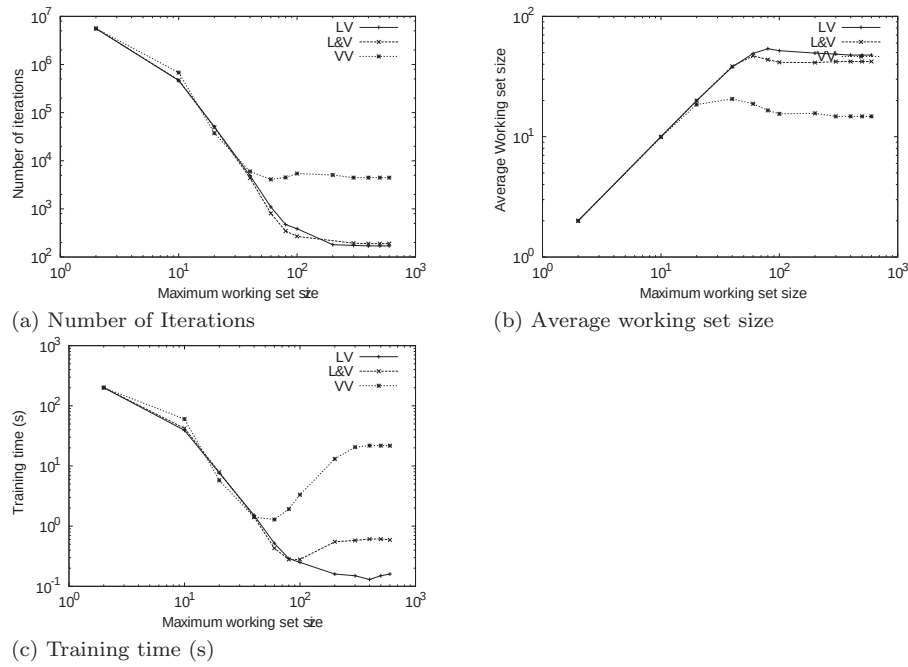


Fig. 3 Comparison of selection strategies using the L1 SVM for the blood cell data with $C = 10^9$

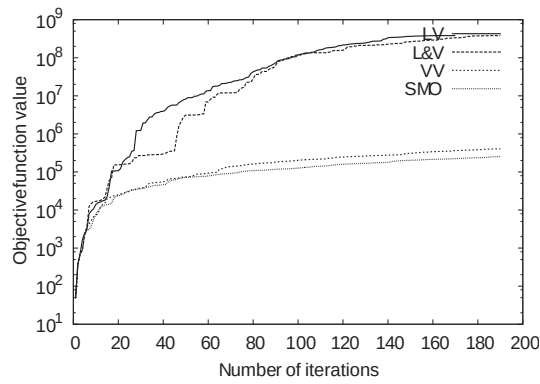


Fig. 4 Objective function values for $C = 10^9$

5.3.3 LS SVM

Figure 7 shows the behaviors of the selection strategies for $C = 1000$. For each selection strategy, training time was shortest for $|W_s| = 10$. Thus we set $|W_s| = 10$ for $C = 1, 1000$.

Figure 8 shows the behaviors of selection strategies for $C = 10^6$. The LV selection strategy was not the best. Thus, for the LS SVM, we selected the

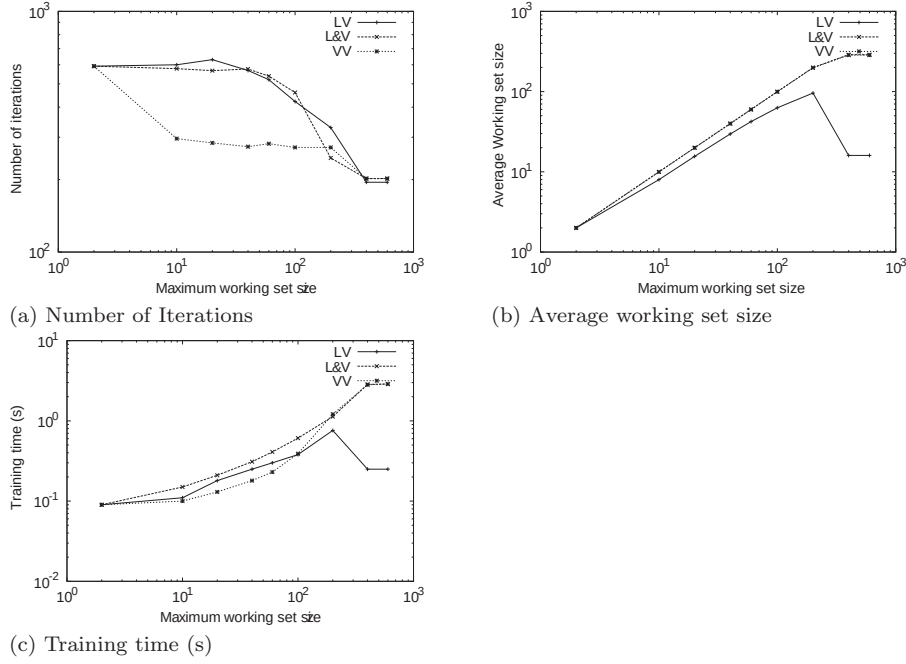


Fig. 5 Comparison of selection strategies using the L2 SVM for the blood cell data with $C = 1$

maximum working set size for each selection strategy for $C = 10^6$: $|W_s| = 100$ for the LV selection strategy, $|W_s| = 400$ both for L&V and VV selection strategies.

5.4 Performance Comparison

To compare the training time of the L1 SVM, we used two most popular training tools: LIBSVM [29], which is based on the second order SMO and SVMlight [22], which is based on fixed-size chunking.

In [36], the revised simplex method for SVM training (SVM-RSQP) was compared with SVM-QP [32], SVMlight [22] and LIBSVM [29]. According to the computer experiments, for the 17 cases tested, SVM-RSQP showed the shortest training time 8 times and LIBSVM did 7 times, and SVMlight twice. Comparing SVM-RSQP and LIBSVM, for small C values both methods are comparable, but for large C values SVM-RSQP is faster. Therefore, for SMO-NM to be comparable to SVM-RSQP for small C values, SMO-NM needs to be comparable to LIBSVM for these values.

The version we used was LIBSVM-3.17. By this version training is terminated if training does not finish within 10^6 iterations. If this happened we used LIBSVM-3.0, which does not have this restriction.

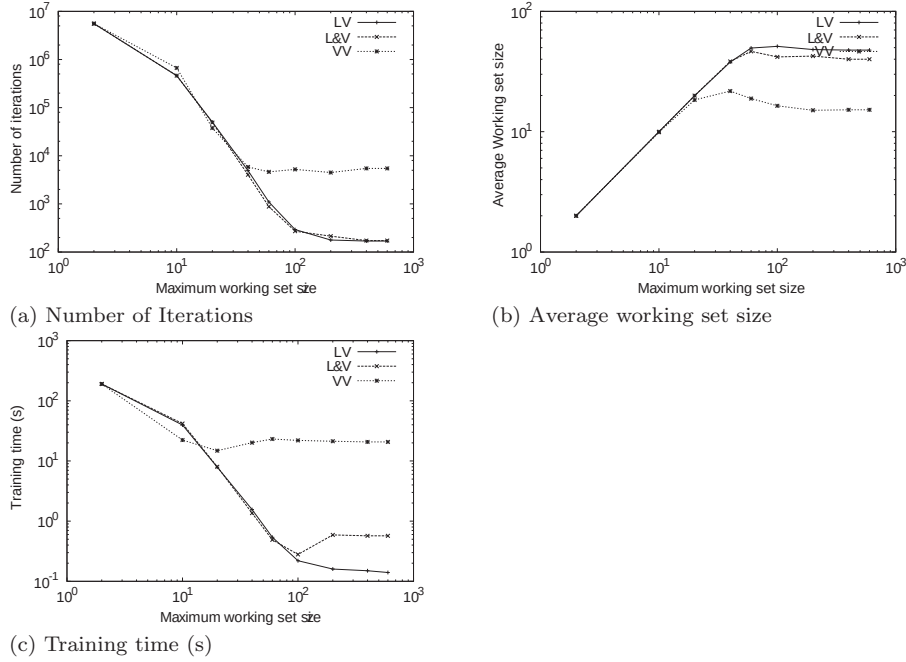


Fig. 6 Comparison of selection strategies using the L2 SVM for the blood cell data with $C = 10^9$

For comparison of training time of the LS SVM, we used MLS-LIBSVM [21], which implements training of the LS SVM with momentum terms on LIBSVM.

In the following first we compare SMO-NM, SMO, LIBSVM, and SVMlight for two-class problems because SVMlight is a binary classifier. Then we compare SMO-NM, SMO, and LIBSVM for multi-class problems. Here, SMO is the SMO-NM with $|W_s| = 2$. Therefore, the algorithm of SMO is based on the second order SMO and the basic algorithm is equivalent to that of LIBSVM. But sophisticated techniques like shrinking are not implemented.

5.4.1 Performance Comparison for Two-class Problems

We compared performance of SMO-NM, SMO, LIBSVM, and SVMlight using blood cell and thyroid training data sets. For each training data set, we extracted training data for classes 2 and 3, which are the most difficult class pairs to separate. Setting $\gamma = 1$, we changed the C value from 1 to 10^9 and measured the training time, the recognition rates of the training data sets and the test data sets, and the numbers of iterations.

Figure 9 shows the results. In Fig. 9 (a), training time of the four methods were comparable for C values smaller than or equal to 1000. For the C values larger than 1000, training time of SMO-NM was almost constant, but for the other three methods training time increased rapidly. As the C value was

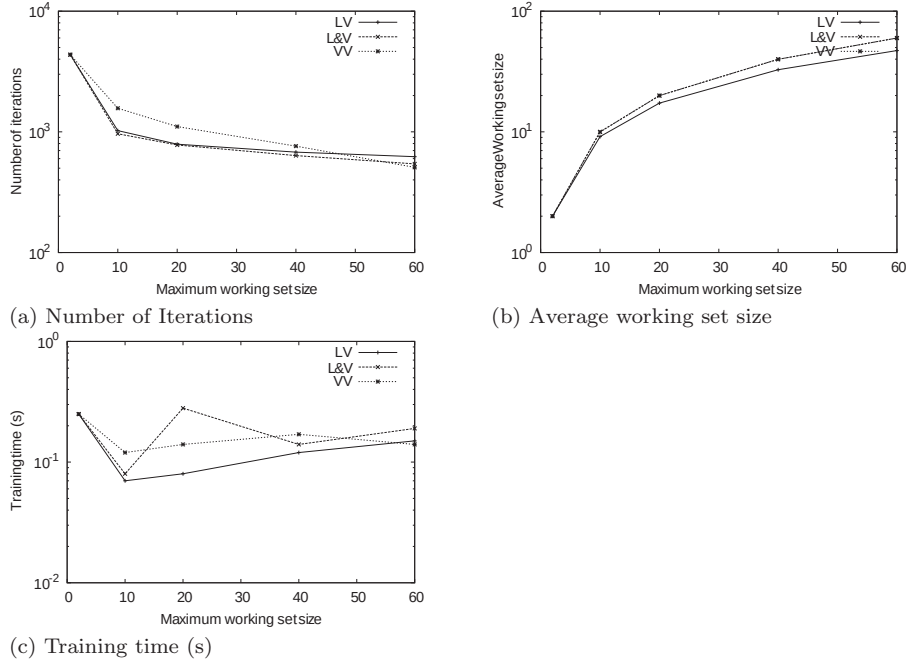


Fig. 7 Comparison of selection strategies using the LS SVM for the blood cell data with $C = 1000$

increased, the recognition rates of SMO-NM and SMO increased monotonically for the training data set (Fig. 9 (b)). But for LIBSVM, the recognition rates were lower than those by SMO-NM and SMO for $C = 10^7, 10^8$. This means that the optimal solutions were not obtained for these values. The recognition rates of SVMlight behaved strangely, namely, they were almost constant. The recognition rates of the test data sets were almost the same for $C = 1$ to 10^6 by SMO-NM, SMO, and LIBSVM because almost the same solutions were obtained for the three methods (Fig. 9 (c)). Although the recognition rates by SVMlight were different from those by the other three, those of the test data were almost identical for $C = 1$ to 10^4 for the four methods. The numbers of iterations for SMO-NM were almost constant but for the other three methods, the numbers of iterations increased rapidly as the C values were increased (Fig. 9 (d)). This means that by SMO-NM with the LV strategy, the convergence was drastically improved, while the other three training methods required large numbers of iterations and still LIBSVM and SVMlight did not converge to the optimal solutions for large C values.

Figure 10 shows the behaviors of the four training methods for the thyroid data set. As seen from Fig. 10 (a), training by LIBSVM was fastest for the C value smaller than or equal to 10^5 but for the C values larger than 10^5 , SMO-NM was the fastest. In addition training time by SMO-NM was almost constant. The recognition rates of the training data set monotonically

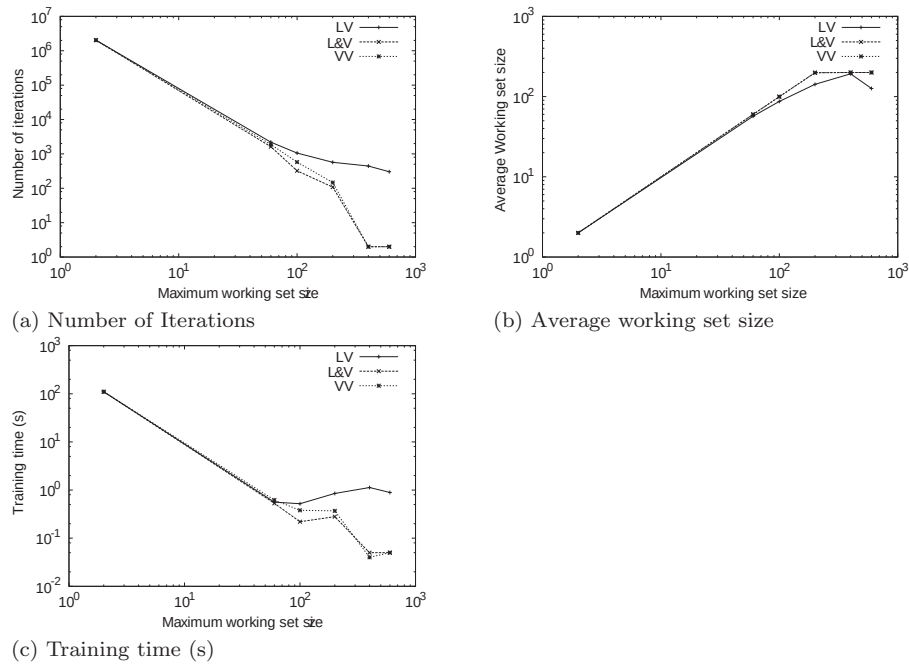


Fig. 8 Comparison of selection strategies using the LS SVM for the blood cell data with $C = 10^6$

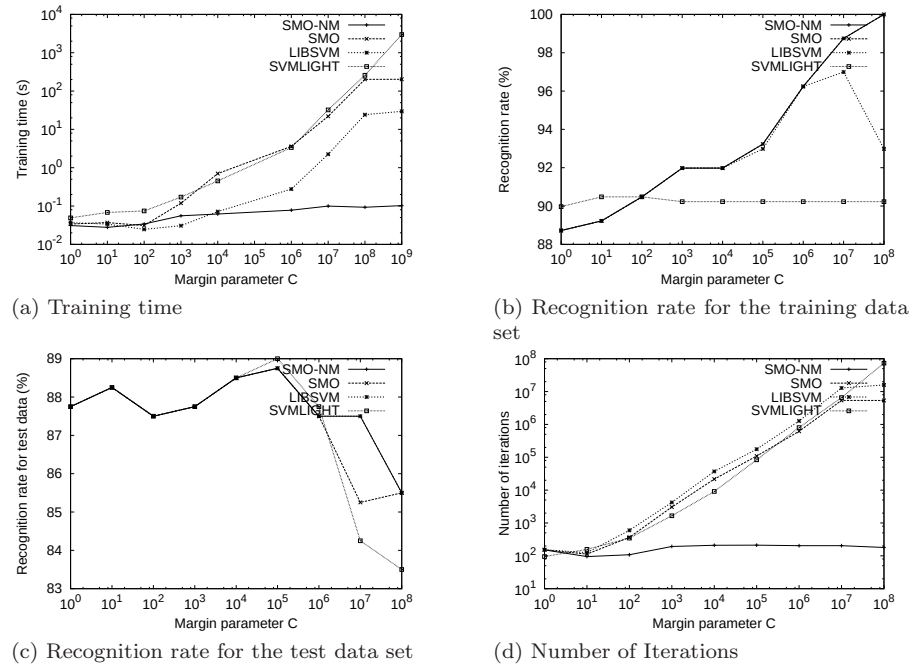


Fig. 9 Performance for blood cell data

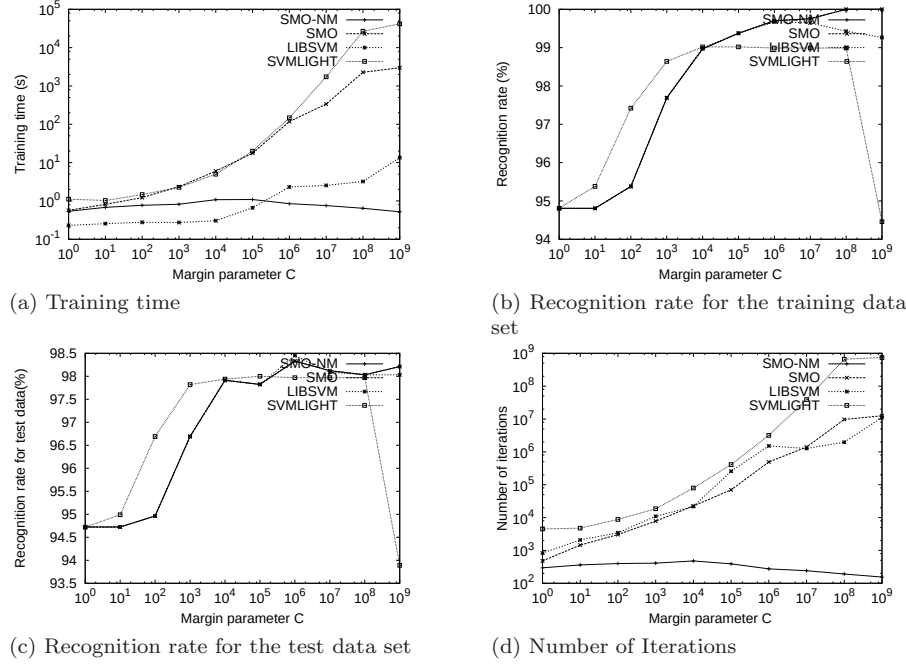


Fig. 10 Performance for thyroid data

increased for SMO-NM and SMO but for LIBSVM and SVMlight, they decreased for large C values (Fig. 10 (b)). The recognition rates of the test data set by SMO-NM, SMO, and LIBSVM were almost the same for $C = 1$ to 10^8 but those by SVMlight were different because the solutions were different. (Fig. 10 (c)) The difference of training time was reflected to the numbers of iterations (Fig. 10 (d)); as the C value was increased, the numbers of iterations for SMO, LIBSVM, and SVMlight increased drastically but for SMO-NM, the number of iterations decreased. SVMlight shows the largest increase in iterations, but this did not contribute in converging to the optimal solution as Fig. 10 (b) shows.

5.5 Performance Evaluation for Multi-class Problems

5.5.1 Performance Evaluation for L1 SVM

Table 3 shows the results for the number of iterations (Iterations), the training time (Time), and the recognition rate (Rec) of the training data set for the L1 SVM. In the table, SMO and SMO-NM denote the second order SMO and the proposed method using the LV selection strategy with $|W_s| = 600$. “Inp/class/data” denotes (the number of features)/(the number of classes)/(the number of data). For “Iterations” and “Time” columns, the

Table 3 Performance comparison for the L1 SVM

Data Inp/class/data	C	Iterations			Time (s)			Rec (%)		
		SMO	SMO-NM	LIBSVM	SMO	SMO-NM	LIBSVM	SMO	SMO-NM	LIBSVM
Blood 13/12/6197	1	129	126	136	1.97	1.97	1.43	89.54	89.56	89.54
	1000	693	52	816	3.20	1.09	0.62	96.10	96.10	96.16
	10^6	39865	38	143046	160.55	1.19	17.42	99.16	99.16	99.21
	10^9	1609847	37	1910825	7091.81	1.63	242.80	99.98	99.98	<i>98.66</i>
Thyroid 21/3/7200	1	454	310	583	1.80	1.95	0.93	92.76	92.76	92.76
	1000	4855	438	6219	6.42	4.14	0.92	97.13	97.13	97.11
	10^6	562420	432	2750222	644.54	9.61	22.25	99.38	99.38	99.35
	10^9	128333695	558	14205786	148478.31	22.73	86.02	99.85	99.85	<i>98.82</i>
Hiragana-50 50/39/9220	1	66	54	77	5.50	5.44	5.98	92.43	92.43	92.51
	1000	139	17	160	4.47	2.23	1.99	100	100	100
Hiragana-13 13/38/16731	1	102	97	111	14.71	14.63	10.02	90.39	90.39	90.40
	1000	181	17	204	9.37	2.94	1.97	99.86	99.86	99.88
	10^6	313	16	371	15.01	2.85	2.04	100	100	100
Hiragana-105 105/38/16731	1	70	57	83	14.09	13.96	18.78	98.03	98.04	98.09
	1000	1289	18	150	9.28	5.66	6.00	100	100	100
Satimage 36/6/6435	1	224	219	238	2.37	2.35	1.84	84.74	84.76	84.79
	1000	2839	264	3135	7.14	4.82	1.34	91.98	91.98	91.85
	10^6	414196	183	966057	1178.28	23.29	103.03	98.63	98.63	98.69
	10^9	1145373	157	9433628	3391.34	30.57	1034.77	100	100	<i>97.00</i>
USPS 256/10/9298	1	197	132	228	10.14	10.64	14.61	96.23	96.21	96.19
	1000	775	68	866	10.80	12.32	7.76	99.99	99.99	99.99
Letter 16/26/20000	1	284	270	313	40.09	40.00	22.67	75.95	75.95	76.08
	1000	2333	118	2779	86.64	22.06	8.71	95.32	95.32	95.10
	10^6	21074	62	26566	733.36	15.11	27.91	99.99	99.99	99.98
	10^9	21476	62	27545	750.52	15.06	29.03	100	100	100

smallest and shortest values are shown in boldface. And in the “Rec” column, the lowest recognition rate caused by early stopping is shown in italic.

Comparing SMO and LIBSVM, the number of iterations of SMO is usually smaller but training time is usually longer. In SMO, sophisticated optimization techniques such as shrinking are not implemented. This might make training time longer.

Comparing SMO and SMO-NM, the number of iterations by SMO-NM was always smaller and training time by SMO-NM was in most cases shorter and comparable even if longer. The recognition rates were almost identical and if different, the difference is within 0.02%.

Comparing SMO-NM and LIBSVM, SMO-NM was faster for $C = 10^6$ and $C = 10^9$ except for hiragana-13 data set but for $C = 1$ and $C = 1000$, LIBSVM was faster in most cases. The recognition rates by LIBSVM for the blood cell, thyroid, and satimage data sets with $C = 10^9$ were by 1% to 2% lower than those by SMO and SMO-NM.

5.5.2 Performance Evaluation for L2 SVM

Table 4 shows the results for the L2 SVM. Although the number of iterations for SMO-NM for $C = 1$ is smaller than for SMO except for the satimage data set, the training time by SMO-NM was longer. This was because the reduction of the number of iterations was not enough to shorten the training time.

For $C = 1000, 10^6, 10^9$, SMO-NM was faster except for the USPS data set. The difference of the recognition rates was 0.01% at most.

Table 4 Performance comparison for the L2 SVM

Data Inp/class/data	C	Iterations		Time (s)		Rec (%)	
		SMO	SMO-NM	SMO	SMO-NM	SMO	SMO-NM
Blood 13/12/6197	1	831	827	8.69	14.60	90.29	90.29
	1000	476	119	2.70	1.93	96.48	96.48
	10^6	31745	35	115.98	1.88	99.18	99.18
	10^9	1356992	37	5794.23	1.61	99.98	99.98
Thyroid 21/3/7200	1	4423	4385	13.76	22.69	93.13	93.13
	1000	6310	2376	13.72	11.78	96.50	96.50
	10^6	448365	354	587.52	77.66	99.31	99.32
	10^9	79959266	358	94007.58	19.42	99.82	99.82
Hiragana-50 50/39/9220	1	402	398	24.61	41.08	93.74	93.73
	1000	122	33	4.09	3.28	100.00	100.00
Hiragana-13 13/38/16731	1	693	690	82.70	144.86	92.58	92.58
	1000	123	29	6.97	4.77	99.90	99.90
	10^6	278	16	13.13	2.79	100.00	100.00
Hiragana-105 105/38/16731	1	461	454	60.47	146.72	98.36	98.36
	1000	118	36	8.77	4.72	100.00	100.00
Satimage 36/6/6435	1	1237	1238	7.04	11.37	85.27	85.27
	1000	2796	802	9.97	6.72	92.48	92.48
	10^6	27991	164	884.92	61.73	99.15	99.15
	10^9	1144538	158	3161.63	31.56	100.00	100.00
USPS 256/10/9298	1	786	777	22.72	30.75	96.60	96.61
	1000	731	375	10.03	11.69	99.99	99.99
Letter 16/26/20000	1	1494	1489	167.69	240.13	79.05	79.06
	1000	1439	355	66.78	40.01	96.24	96.24
	10^6	17660	62	612.00	15.85	100.00	100.00

5.5.3 Performance Evaluation for LS SVM

Table 5 shows the results for SMO, SMO-NM with the LL, L&V, and VV, and MLS-LIBSVM. The numerals in the first column just below the data set names are the training time calculated by solving the set of linear equations using the Cholesky factorization.

For $C = 10^6$, the solutions of MLS-SVM included 1#QNANO and the recognition rates were not calculated. In addition, the average number of iterations for the hiragana-13 data set was not calculated because the numbers of iterations for some class pairs were negative.

Comparing the results with those for the L1 and L2 SVMs the following observation is obtained. For the L1 and L2 SVMs, as the C value was increased from 1 to 1000, usually the number of iterations was smaller and the training time shorter. While for the LS SVM, the situation is quite the opposite. The reason may be as follows: For the L1 and L2 SVMs, because the numbers of support vectors are reduced for the larger C value, the total calculation burden will also be reduced although the α_i values may need to be corrected more times. But for the LS SVM, because all the training data become support vectors, larger the C value is, larger the calculation burden is. And for the $C = 10^6$, computational burden becomes much larger.

Comparing the results of SMO-NM with those of SMO, the number of iterations was smaller, irrespective of selection strategies as Theorem 3 indicated. But the training time was longer for $C = 1$ and so was $C = 1000$ for the VV

Table 5 Performance comparison for the LS SVM

Data	C	Iterations					Time (s)				
		SMO	LV	L&V	VV	MLS	SMO	LV	L&V	VV	MLS
Blood	1	1279	1279	1276	566	1399	14.39	22.28	28.47	32.38	6.39
	1000	8313	2112	2025	3800	3573	80.63	37.82	43.80	83.10	8.96
	10^6	3557477	2528	115	1780	39552357	33642	317.59	334.76	5447	50183
Thyroid	1	6699	6672	6425	4016	7206	21.99	34.98	39.55	27.03	23.82
	1000	32450	11758	11522	20827	20933	95.27	60.14	69.04	129.53	34.43
	10^6	8286792	140319	1158	3825	2774995	23472	6024	411.11	1336	863
Hiragana-50	1	590	586	539	275	646	37.43	58.22	67.41	39.43	28.31
	1000	4317	1258	1228	2081	2001	231.31	129.00	147.54	256.67	38.53
Hiragana-13	1	1095	1093	1049	496	1205	161.12	241.25	316.29	175.92	53.26
	1000	5974	1553	1491	2811	2825	798.67	357.06	434.79	907.31	72.86
	10^6	2033763	1617	234	4182	—	282919	3277	2308	12765	1745580
Hiragana-105	1	995	989	950	470	1104	192.26	237.16	336.47	191.37	152.65
	1000	8513	2606	2545	3631	3397	1141	637.14	756.33	1227	184.85
Satimage	1	2735	2725	2671	1255	3013	16.09	24.01	29.86	16.92	9.37
	1000	23862	5884	5742	12228	8165	117.38	51.59	59.36	137.14	12.25
	10^6	9088606	37797	731	3265	58196547	47450	2410	727.08	3249	34397
USPS	1	2206	2178	2130	1095	2427	55.77	71.29	84.94	60.55	113.09
	1000	40406	19593	19425	19199	17582	609.38	485.71	542.50	617.04	134.57
Letter	1	2033	2026	1978	943	2235	269.11	333.01	401.27	213.56	75.91
	1000	15906	3823	3737	7628	5859	1801.22	611.68	707.31	1406.03	105.62
	10^6	6478122	11857	185	348	26581216	733296	9776	2883	5323	227127

selection strategy. For $C = 10^6$, SMO-NM was faster. Among three strategies, the L&V strategy worked best, and the LV the second best.

Comparing SMO-NM, SMO, and MLS-LIBSVM, MLS-LIBSVM worked best for $C = 1, 1000$, except for the thyroid data set with $C = 1$ and the USPS data set with $C = 1$. Especially, the gap between SMO and MLS-LIBSVM was much larger than that for L1 SVM. One reason may be the introduction of the momentum term for MLS-LIBSVM. But the MLS-LIBSVM slowed down extremely for $C = 10^6$.

To clarify the effect of the momentum term in MLS-LIBSVM, we ran MLS-LIBSVM without the momentum term. For $C = 1$, training without the momentum term was a litter faster for all the data sets. For $C = 1000$ the training time was 1.2 to 1.7 times longer than that with the momentum term but still it was shorter than that by SMO or SMO-NM. Therefore, slow training by SMO or SMO-NM was due to LIBSVM's acceleration methods, not implemented in SMO and SMO-NM.

Comparing the iterative methods with the training method by solving the set of linear equations, except for the thyroid data set, any iterative method failed to be faster for $C = 10^6$. For large C values, still further speedup is necessary for the iterative methods.

Acknowledgment

We are grateful to Á. Barbero for providing us the MLS-LIBSVM software and helped us use the software. This work was supported by JSPS KAKENHI Grant Number 25420438.

6 Conclusions

We proposed fusing the SMO with Newton's method and called it SMO-NM. To speed up training for a large margin parameter value, we detect a loop, in which a previously selected SMO variable is selected as the current SMO variable, where the SMO variables are the variables that are selected by SMO. In addition to the SMO variables, we add loop variables that are selected in the previous steps to the working set. As a variant of this strategy, we add violating variables to the working set.

We show that the solutions of the L1 and L2 SVMs are unbounded if the margin parameter value approaches infinity under the condition that the classification problem is not linearly separable in the feature space. And for the LS SVM, the solution is unbounded if the margin parameter value approaches infinity under the condition that the mapped training data are not linearly independent in the feature space. This shows that training of the LS SVM is more difficult than that of L1/L2 SVMs.

By computer experiment using eight benchmark data sets, we demonstrated that training by SMO-NM was faster than, or comparable to, by SMO especially for the L1 SVM. For the L2 and LS SVMs, training by SMO-NM slowed down for a small margin parameter value. This was because enough reduction of iterations was not achieved to compensate for the increase in time caused by matrix inversion. But for a larger margin parameter value, SMO-NM was always faster than SMO.

We compared SMO-NM and SMO with LIBSVM for the L1 SVM and with MLS-LIBSVM for the LS SVM. For a large margin parameter value, training by SMO-NM was always faster than LIBSVM or MLS-LIBSVM. But for a small or medium margin parameter value, SMO-NM and SMO was sometimes slower than LIBSVM or much slower than MLS-LIBSVM. This was mainly caused by not implementing the sophisticated optimization techniques used in LIBSVM and MLS-LIBSVM. Because the speedup of SMO contributes in speeding up SMO-NM, speedup is expected by implementing the sophisticated techniques.

References

1. V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, NY, 1995.
2. J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK, 2004.
3. R. Khemchandani, A. Karpatne, and S. Chandra. Proximal support tensor machines. *International Journal of Machine Learning and Cybernetics*, 4(6):703–712, 2013.
4. Q. He and C. Wu. Separating theorem of samples in Banach space for support vector machine learning. *International Journal of Machine Learning and Cybernetics*, 2(1):49–54, 2011.
5. X. Wang, S. Lu, and J. Zhai. Fast fuzzy multi-category SVM based on support vector domain description. *International Journal of Pattern Recognition and Artificial Intelligences*, 22(1):109–120, 2008.

6. X. Wang, Q. He, D. Chen, and D. Yeung. A genetic algorithm for solving the inverse problem of support vector machines. *Neurocomputing*, 68:225–238, 2005.
7. H. Byun and S.-W. Lee. A survey on pattern recognition applications of support vector machines. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(3):459–486, 2003.
8. A. Widodo and B.-S. Yang. Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical Systems and Signal Processing*, 21(6):2560–2574, 2007.
9. B. Schölkopf, K. Tsuda, and J.-P. Vert. *Kernel Methods in Computational Biology*. MIT Press, Cambridge, MA, 2004.
10. N. Chen, W. Lu, J. Yang, and G. Li. *Support Vector Machine In Chemistry*. World Scientific Publishing Company, Singapore, 2004.
11. J. R. Bradford and D. R. Westhead. Improved prediction of protein-protein binding sites using a support vector machines approach. *Bioinformatics*, 21(8):1487–1494, 2005.
12. C. Ding and H. Peng. Minimum redundancy feature selection from microarray gene expression data. *Journal of Bioinformatics and Computational Biology*, 3(2):185–205, 2005.
13. S. Abe. *Support Vector Machines for Pattern Classification*. Springer-Verlag, London, UK, second edition, 2010.
14. J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 185–208. MIT Press, Cambridge, MA, 1999.
15. S. S. Keerthi and E. G. Gilbert. Convergence of a generalized SMO algorithm for SVM classifier design. *Machine Learning*, 46(1–3):351–360, 2002.
16. R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.
17. Á. Barbero and J. R. Dorronsoro. Faster directions for second order SMO. In K. Diamantaras, W. Duch, and L. S. Iliadis, editors, *Artificial Neural Networks – ICANN 2010*, volume 6353 of *Lecture Notes in Computer Science*, pages 30–39. Springer, 2010.
18. Á. Barbero and J. R. Dorronsoro. Momentum sequential minimal optimization: An accelerated method for support vector machine training. In *Proceedings of the 2011 International Joint Conference on Neural Networks (IJCNN 2011)*, pages 370–377, San Jose, CA, 2011.
19. W. Chu, C. J. Ong, and S. S. Keerthi. An improved conjugate gradient scheme to the solution of least squares SVM. *IEEE Transactions on Neural Networks*, 16(2):498–501, 2005.
20. J. López and J. A. K. Suykens. First and second order SMO algorithms for LS-SVM classifiers. *Neural Processing Letters*, 33(1):33–44, 2011.
21. J. López, Á. Barbero, and J. R. Dorronsoro. Momentum acceleration of least-squares support vector machines. In T. Honkela, W. Duch, M. Girolami, and S. Kaski, editors, *Artificial Neural Networks and Machine Learning – ICANN 2011*, volume 6792 of *Lecture Notes in Computer Science*, pages 135–142. Springer, 2011.
22. T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 169–184. MIT Press, Cambridge, MA, 1999.
23. S.V.N. Vishwanathan, A. J. Smola, and M. N. Murty. SimpleSVM. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, volume 2, pages 760–767, Washington DC, 2003.
24. W. Kienzle and B. Schölkopf. Training support vector machines with multiple equality constraints. In *Proceedings of Sixteenth European Conference on Machine Learning (ECML 2005)*, volume LNAI 3720, pages 182–193, Porto, Portugal, 2005.
25. C. Sentelle, M. Georgiopoulos, G. C. Anagnostopoulos, and C. Young. On extending the SMO algorithm sub-problem. In *Proceedings of the 2007 International Joint Conference on Neural Networks (IJCNN 2007)*, pages 886–891, Orlando, FL, 2007.
26. R. A. Hernandez, M. Strum, J. C. Wang, and J. A. Q. Gonzalez. The multiple pairs SMO: A modified SMO algorithm for the acceleration of the SVM training. In *Proceedings of the 2009 International Joint Conference on Neural Networks (IJCNN 2009)*, pages 1221–1228, Atlanta, GA, 2009.

27. S. Abe. Fast support vector training by Newton's method. In T. Honkela, W. Duch, M. Girolami, and S. Kaski, editors, *Artificial Neural Networks and Machine Learning – ICANN 2011*, volume 6792 of *Lecture Notes in Computer Science*, pages 143–150. Springer, 2011.
28. G. Loosli and S. Canu. Comments on the “Core vector machines: Fast SVM training on very large data sets”. *Journal of Machine Learning Research*, 8:291–301, 2007.
29. C.-C. Chang and C.-J. Lin. LIBSVM—A library for support vector machines: <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>.
30. G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 409–415. MIT Press, Cambridge, MA, 2001.
31. A. Shilton, M. Palaniswami, D. Ralph, and A. C. Tsoi. Incremental training of support vector machines. *IEEE Transactions on Neural Networks*, 16(1):114–131, 2005.
32. K. Scheinberg. An efficient implementation of an active set method for SVMs. *Journal of Machine Learning Research*, 7:2237–2257, 2006.
33. S. Abe. Batch support vector training based on exact incremental training. In V. Kůrková, R. Neruda, and J. Koutník, editors, *Artificial Neural Networks (ICANN 2008)—Proceedings of the Eighteenth International Conference, Prague, Czech Republic, Part I*, pages 527–536. Springer-Verlag, Berlin, Germany, 2008.
34. H. Gálmeanu and R. Andonie. Implementation issues of an incremental and decremental SVM. In V. Kůrková, R. Neruda, and J. Koutník, editors, *Artificial Neural Networks (ICANN 2008)—Proceedings of the Eighteenth International Conference, Prague, Czech Republic, Part I*, pages 325–335. Springer-Verlag, Berlin, Germany, 2008.
35. C. Sentelle, G. C. Anagnostopoulos, and M. Georgiopoulos. An efficient active set method for SVM training without singular inner problems. In *Proceedings of the 2009 International Joint Conference on Neural Networks (IJCNN 2009)*, pages 2875–2882, Atlanta, GA, 2009.
36. C. Sentelle, G. C. Anagnostopoulos, and M. Georgiopoulos. Efficient revised simplex method for SVM training. *IEEE Transactions on Neural Networks*, 22(10):1650–1661, 2011.
37. G. H. Golub and C. F. Van Loan. *Matrix Computations, Third Edition*. The Johns Hopkins University Press, Baltimore, MD, 1996.
38. S. Abe. Sparse least squares support vector training in the reduced empirical feature space. *Pattern Analysis and Applications*, 10(3):203–214, 2007.
39. A. Asuncion and D. J. Newman. UCI machine learning repository, <http://www.ics.uci.edu/~mllearn/MLRepository.html>. 2007.
40. USPS Dataset. <http://www-i6.informatik.rwth-aachen.de/~keyzers/usps.html>.