computers

MDPI

Article

Minimal Complexity Support Vector Machines for Pattern Classification †

Shigeo Abe

Kobe University, Kobe 657-8501, Japan; abe@kobe-u.ac.jp

† This paper is an extended version of our paper published in Abe, S. Minimal Complexity Support Vector Machines. In Artificial Neural Networks in Pattern Recognition. ANNPR 2020. Lecture Notes in Computer Science, Vol 12294; Schilling, F.P., Stadelmann, T., Eds.; Springer: Cham, Switzerland, 2020; pp. 89–101.

Received: 15 September 2020; Accepted: 30 October 2020; Published: 4 November 2020

Abstract: Minimal complexity machines (MCMs) minimize the VC (Vapnik-Chervonenkis) dimension to obtain high generalization abilities. However, because the regularization term is not included in the objective function, the solution is not unique. In this paper, to solve this problem, we discuss fusing the MCM and the standard support vector machine (L1 SVM). This is realized by minimizing the maximum margin in the L1 SVM. We call the machine Minimum complexity L1 SVM (ML1 SVM). The associated dual problem has twice the number of dual variables and the ML1 SVM is trained by alternatingly optimizing the dual variables associated with the regularization term and with the VC dimension. We compare the ML1 SVM with other types of SVMs including the L1 SVM using several benchmark datasets and show that the ML1 SVM performs better than or comparable to the L1 SVM.

Keywords: least squares support vector machines; margin distributions; minimum complexity machines; pattern classification; support vector machines; VC dimension

1. Introduction

In the support vector machine (SVM) [1,2], training data are mapped into the high dimensional feature space, and in that space, the separating hyperplane is determined so that the nearest training data of both classes are maximally separated. Here, the distance between a data sample and the separating hyperplane is called margin. Thus, the SVM is trained so that the minimum margin is maximized.

Motivated by the success of SVMs in real world applications, many SVM-like classifiers have been developed to improve the generalization ability. The ideas of extensions lie in incorporating the data distribution (or margin distribution) to the classifiers because the SVM only considers the data that are around the separating hyperplane. If the distribution of one class is different from that of the other class, the separating hyperplane with the same minimum margin for both classes may not be optimal.

To cope with this, one approach proposes kernels based on the Mahalanobis distance [3–10]. Another approach reformulates the SVM such that the margin is measured by the Mahalanobis distance [11–15].

Yet another approach controls the overall margins instead of the minimum margin [16–22]. In [16], only the average margin is maximized. While the architecture is very simple, the generalization ability is inferior to that of the SVM [17,19]. To improve the generalization ability, the equality constraints were added in [19], but this results in the least squares SVM (LS SVM).

In [17], a large margin distribution machine (LDM) was proposed, in which the average margin is maximized and the margin variance is minimized. The LDM is formulated based on the SVM with inequality constraints. While the generalization ability is better than that of the SVM, the problem is that the LDM has one more hyperparameter than the SVM does. To cope with this problem, in [20,21],

the unconstrained LDM (ULDM) was proposed, which has the equal number of hyperparameters and which has the generalization ability comparable to that of the LDM and the SVM.

The generalization ability of the SVM can be analyzed by the VC (Vapnik-Chervonenkis) dimension [1] and the generalization ability will be improved if the radius-margin ratio is minimized, where the radius is the radius of the minimum hypersphere that encloses all the training data in the feature space. The minimum hypersphere changes if the mapping function is changed during model selection, where all the parameter values including the value for the mapping function are determined, or feature selection is performed during training a classifier. So, minimization of the radius-margin ratio can be utilized in selecting the optimal mapping function [23,24]. In [25,26], feature selection is carried out during training by minimizing the radius-margin ratio.

If the center of the hypersphere is assumed to be at the origin, the hypersphere can be minimized for a given feature space as discussed in [27,28]. The minimal complexity machine (MCM) was derived based on this assumption. In the MCM, the VC dimension is minimized by minimizing the upper bound of the soft-margin constraints for the decision function. Because the regularization term is not included, the MCM is trained by linear programming. The quadratic version of the MCM (QMCM) tries to minimizes the VC dimension directly [28]. The generalization performance of the MCM is shown to be superior to that of the SVM, but according to our analysis [29], the solution is non-unique and the generalization ability is not better than that of the SVM. The problem of non-uniqueness of the solution is solved by adding the regularization term in the objective function of the MCM, which is a fusion of the MCM and the linear programming SVM (LP SVM) called MLP SVM.

In [30], to improve the generalization ability of the standard SVM (L1 SVM), we proposed fusing the MCM and the L1 SVM, which is the minimal complexity L1 SVM (ML1 SVM). We also proposed $ML1_v$ SVM, whose component is more similar to the L1 SVM. By the computer experiment using RBF (radius basis function) kernels, the proposed classifiers generalize better than or comparable to the L1 SVM.

In this paper we discuss the ML1 SVM and $ML1_v$ SVM more in detail, propose their training methods, prove their convergence, and demonstrate the effectiveness of the proposed classifiers using polynomial kernels in addition to RBF kernels.

The ML1 SVM is obtained by adding the upper bound on the decision function and the upper bound minimization term in the objective function of the L1 SVM. We show that this corresponds to minimizing the maximum margin. We derive the dual form of the ML1 SVM with one set of variables associated with the soft-margin constraints and the other set, upper-bound constraints. We then decompose the dual ML1 SVM into two subproblems: one for the soft-margin constraints, which is similar to the dual L1 SVM, and the other for the upper-bound constraints. These subproblems include neither the bias term nor the upper bound. Thus, for a convergence check, we derive the exact KKT (Karush-Kuhn-Tucker) conditions that do not include the bias term and the upper bound. The second subproblem is different from the first subproblem in that it includes the inequality constraint on the sum of dual variables. To remove this, we change the inequality constraint into two equality constraints and obtain $ML1_v$ SVM.

We consider training the ML1 SVM and $ML1_v$ SVM optimizing the first and the second subprograms, alternatingly. Because the ML1 SVM and $ML1_v$ SVM are very similar to the L1 SVM, we discuss the training method based on Sequential Minimum Optimization (SMO) fused with Newton's method [31]. We also show the convergence proof of the proposed training methods. By computer experiments using polynomial kernels, in addition to RBF kernels, we compare the ML1 SVM and $ML1_v$ SVM with other classifiers including the L1 SVM.

In Section 2, we summarize the architectures of L1 SVM and the MCM. In Section 3, we discuss the architecture of the ML1 SVM and derive the dual form and the optimality conditions of the solution. Then, we discuss the architecture of the $ML1_v$ SVM. In Section 4, we discuss the training method that is an extension of SMO fused with Newton's method [31] and the working set selection method. We also

show the convergence proof of the proposed method. In Section 5, we evaluate the generalization ability of the ML1 SVM and other SVM-like classifiers using two-class and multiclass problems.

2. L1 Support Vector Machines and Minimal Complexity Machines

In this section, we briefly explain the architectures of the L1 SVM and the MCM [27]. Then we discuss the problem of non-unique solutions of the MCM and one approach to solving the problem [29].

2.1. L1 Support Vector Machines

Let M training data and their labels be $\{\mathbf{x}_i, y_i\}$ $(i = 1, \ldots, M)$, where $\mathbf{x}_i$ is an n-dimensional input vector and $y_i = 1$ for Class 1 and -1 for Class 2. The input space is mapped into the l-dimensional feature space by the mapping function $\boldsymbol{\phi}(\mathbf{x})$ and in the feature space the separating hyperplane is constructed. The decision boundary given by the decision function $f(\mathbf{x})$ is given by

$$f(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) + b = 0, \tag{1}$$

where $\mathbf{w}$ is the l-dimensional coefficient vector and b is the bias term.

The primal form of the L1 SVM is given by

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

$$\text{subject to} \qquad y_i\left(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) + b\right) + \xi_i \geq 1, \quad \xi_i \geq 0 \quad \text{for} \quad i = 1, \ldots, M, \tag{3}$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_M)^\top$, ξ_i is the slack variable for $\mathbf{x}_i$, and C is the margin parameter that determines the trade-off between the maximization of the margin and minimization of the classification error. Inequalities (3) are called soft-margin constraints.

The dual problem of (2) and (3) is given by

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

$$\text{subject to} \qquad \sum_{i=1}^{M} y_i\alpha_i = 0, \quad C \geq \alpha_i \geq 0 \quad \text{for} \quad i = 1, \ldots, M, \tag{5}$$

where $K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}^\top(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x}_j)$, $K(\mathbf{x}_i, \mathbf{x}_j)$ is the kernel function, $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M)^\top$, α_i is the Lagrange multiplier for the ith inequality constraint, and $\mathbf{x}_i$ associated with positive α_i (> 0) is called a support vector.

The decision function given by (1) is rewritten as follows:

$$f(\mathbf{x}) = \sum_{i\in S}\alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b, \tag{6}$$

where S is the set of support vector indices.

2.2. Minimal Complexity Machines

The VC (Vapnik-Chervonenkis) dimension is a measure for estimating the generalization ability of a classifier and lowering the VC dimension leads to realizing a higher generalization ability. For an SVM-like classifier with the minimum margin δ_{min}, the VC dimension D is bounded by [1]

$$D \leq 1 + \min\left(\frac{R^2}{\delta_{\text{min}}^2}, l\right), \tag{7}$$

where R is the radius of the smallest hypersphere that encloses all the training data.

In training the L1 SVM, both R and l are not changed. In the LS SVM, where ξ_i are replaced with ξ_i^2 in (2) and the inequality constraints, with equality constraints in (3), although both R and l are not changed by training, the second term in the objective function works to minimize the square sums of $y_i f(\mathbf{x}_i) - 1$. Therefore, like the LDM and ULDM, this term works to condense the margin distribution in the direction orthogonal to the separating hyperplane.

In [27], first the linear MCM, in which the classification problem is linearly separable, is derived by minimizing the VC-dimension, i.e., $R/\delta_{\min}$ in (7). In the input space, R and $\delta_{\min}$ are calculated, respectively, by

$$R = \min_{i=1\ldots,M} \|(\mathbf{x}_i^\top, 1)\|, \tag{8}$$

$$\delta_{\min} = \min_{i=1\ldots,M} \frac{\|(\mathbf{w}^\top, b)(\mathbf{x}_i^\top, 1)^\top\|}{\|(\mathbf{x}_i^\top, 1)\|}, \tag{9}$$

where 1 is added to $\mathbf{x}_i$ to make the separating hyperplane passes through the origin in the augmented space. Equation (8) shows that the minimum hypersphere is also assumed to pass through the origin in the augmented space. After some derivations, the linear MCM is derived. The nonlinear MCM is derived as follows:

$$\text{minimize} \qquad Q(\boldsymbol{\alpha}, h, \boldsymbol{\xi}, b) = h + C \sum_{i=1}^{M} \xi_i \tag{10}$$

$$\text{subject to} \qquad h \geq y_i \left(\sum_{j=1}^{M} \alpha_j K_{ij} + b \right) + \xi_i \geq 1 \qquad \text{for } i = 1, \ldots M, \tag{11}$$

where h is the upper bound of the soft-margin constraints and $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. Here, the mapping function $\boldsymbol{\phi}(\mathbf{x})$ in (1) is [32]

$$\boldsymbol{\phi}(\mathbf{x}) = (K_{11}, \ldots, K_{1M})^\top, \tag{12}$$

and $\mathbf{w} = \boldsymbol{\alpha}$. The MCM can be solved by linear programming.

Because the upper bound h in (11) is minimized in (10), the separating hyperplane is determined so that the maximum distance between the training data and the separating hyperplane is minimized. This is a similar idea to that of the LS SVM, LDM, and ULDM.

The MCM is derived based on the minimization of the VC dimension, and thus considers maximizing the minimum margin. However, the MCM does not explicitly include the term related to the margin maximization. This makes the solution non-unique and unbounded.

To make the solution unique under the condition that the extended MCM still is a linear programming problem, in [29] we proposed the minimal complexity LP SVM (MLP SVM), which fuses the MCM an LP SVM:

$$\text{minimize} \qquad Q(\boldsymbol{\alpha}, h, \boldsymbol{\xi}, b) = C_h h + \sum_{i=1}^{M} (C_\alpha |\alpha_i| + C \xi_i) \tag{13}$$

$$\text{subject to} \qquad h \geq y_i \left(\sum_{j=1}^{M} \alpha_j K_{ij} + b \right) + \xi_i \geq 1 \quad \text{for } i = 1, \ldots M, \tag{14}$$

where C_h is the positive parameter and $C_\alpha = 1$. Deleting $C_h h$ in (13) and upper bound h in (14), we obtain the LP SVM. Setting $C_h = 1$ and $C_\alpha = 0$ in (13), we obtain the MCM. Compared to the MCM and the LP SVM, the number of hyperparameters increases by 1.

According to the computer experiment, in general, the MLP SVM is better than the MCM and LP SVM in generalization abilities. However, it is inferior to the L1 SVM [29].

3. Minimal Complexity L1 Support Vector Machines

In this section, we discuss the ML1 SVM, which consists of two optimization subprograms and derive the KKT conditions that the optimal solution of the ML1 SVM must satisfy. Then, we discuss a variant of the ML1 SVM, whose two subprograms are more similar.

3.1. Architecture

The LDM and ULDM maximize the average margin and minimize the average margin variance and LS SVM makes data condense around the minimum margin. The idea of the MCM to minimize the VC dimension results in minimizing the maximum distance of the data from the separating hyperplane. Therefore, these classifiers have the idea in common: condense data as near as possible to the separating hyperplane.

In [29], we proposed fusing the MCM and the LP SVM. Similar to this idea, here we fuse the MCM given by (10) and (11) and the L1 SVM given by (2) and (3):

$$\begin{aligned}
\text{minimize} \quad & Q(\mathbf{w}, b, h, \boldsymbol{\xi}) = C_h\, h + \frac{1}{2}\,\|\mathbf{w}\|^2 + C \sum_{i=1}^{M} \xi_i && (15)\\
\text{subject to} \quad & y_i\,(\mathbf{w}^\top\, \boldsymbol{\phi}(\mathbf{x}_i) + b) + \xi_i \geq 1, \quad \xi_i \geq 0 \quad \text{for} \quad i = 1, \ldots, M, && (16)\\
& h \geq y_i\,(\mathbf{w}^\top\, \boldsymbol{\phi}(\mathbf{x}_i) + b) \quad \text{for} \quad i = 1, \ldots, M, && (17)\\
& h \geq 1, && (18)
\end{aligned}$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_M)^\top$, C_h is the parameter to control the volume that the training data occupy, and h is the upper bound of the constraints. The upper bound defined by (11) is redefined by (17) and (18), which exclude ξ_i. This makes the KKT conditions for the upper bound simpler. We call (17) the upper bound constraints and the above classifier minimum complexity L1 SVM (ML1 SVM). The right-hand side of (17) shows the margin with the minimum margin normalized to 1 if the solution is obtained. Therefore, because h is the upper bound of the margins and h is minimized in (15), the ML1 SVM maximizes the minimum margin and minimizes the maximum margin simultaneously.

If we use (12), we can directly solve (15) to (18). However, sparsity of the solution will be lost. Therefore, in a way similar to solving the L1 SVM, we solve the dual problem of (15) to (18).

Introducing the nonnegative Lagrange multipliers α_i, β_i, and η, we obtain

$$\begin{aligned}
Q(\mathbf{w}, b, h, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \eta) \quad = \quad & C_h\, h + \frac{1}{2}\,\|\mathbf{w}\|^2 + C \sum_{i=1}^{M} \xi_i \\
& - \sum_{i=1}^{M} \alpha_i \left(y_i\,(\mathbf{w}^\top\, \boldsymbol{\phi}(\mathbf{x}_i) + b) - 1 + \xi_i\right) - \sum_{i=1}^{M} \beta_i\, \xi_i \\
& - \sum_{i=1}^{M} \alpha_{M+i} \left(h - y_i\,(\mathbf{w}^\top\, \boldsymbol{\phi}(\mathbf{x}_i) + b)\right) - (h-1)\,\eta, \qquad (19)
\end{aligned}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M, \alpha_{M+1}, \ldots, \alpha_{2M})^\top$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_M)^\top$.

For the optimal solution, the following KKT conditions are satisfied:

$$\frac{\partial Q(\mathbf{w}, b, h, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \eta)}{\partial \mathbf{w}} = \mathbf{0}, \qquad (20)$$

$$\frac{\partial Q(\mathbf{w}, b, h, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \eta)}{\partial h} = 0, \qquad (21)$$

$$\frac{\partial Q(\mathbf{w}, b, h, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \eta)}{\partial b} = 0, \qquad (22)$$

$$\frac{\partial Q(\mathbf{w}, b, h, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \eta)}{\partial \boldsymbol{\xi}} = \mathbf{0}, \tag{23}$$

$$\alpha_i \left(y_i \left(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) + b \right) - 1 + \xi_i \right) = 0 \quad \alpha_i \geq 0 \qquad \text{for} \quad i = 1, \ldots, M, \tag{24}$$

$$\alpha_{M+i} \left(h - y_i \left(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) + b \right) \right) = 0, \quad \alpha_{M+i} \geq 0 \qquad \text{for} \quad i = 1, \ldots, M, \tag{25}$$

$$\beta_i \, \xi_i = 0, \quad \beta_i \geq 0, \quad \xi_i \geq 0 \qquad \text{for} \quad i = 1, \ldots, M, \tag{26}$$

$$(h - 1) \, \eta = 0, \quad h \geq 1, \quad \eta \geq 0, \tag{27}$$

where $\mathbf{0}$ is the zero vector whose elements are zero. Equations (24) to (27) are called KKT complementarity conditions.

Substituting (19) into (20) to (23) gives, respectively,

$$\mathbf{w} = \sum_{i=1}^{M} (\alpha_i - \alpha_{M+i}) \, y_i \, \boldsymbol{\phi}(\mathbf{x}_i), \tag{28}$$

$$\sum_{i=1}^{M} \alpha_{M+i} = C_h - \eta, \tag{29}$$

$$\sum_{i=1}^{M} (\alpha_i - \alpha_{M+i}) \, y_i = 0, \tag{30}$$

$$\alpha_i + \beta_i = C \quad \text{for} \quad i = 1, \ldots, M. \tag{31}$$

Substituting (28) to (31) into (19), we obtain the following dual problem:

$$\text{maximize} \quad Q(\boldsymbol{\alpha}) = \sum_{i=1}^{M} (\alpha_i - \alpha_{M+i}) - \frac{1}{2} \sum_{i,j=1}^{M} (\alpha_i - \alpha_{M+i}) \, (\alpha_j - \alpha_{M+j}) \, y_i \, y_j \, K(\mathbf{x}_i, \mathbf{x}_j) \tag{32}$$

$$\text{subject to} \quad \sum_{i=1}^{M} y_i \, (\alpha_i - \alpha_{M+i}) = 0, \tag{33}$$

$$C_h \geq \sum_{i=1}^{M} \alpha_{M+i}, \tag{34}$$

$$C \geq \alpha_i \geq 0, \; C_h \geq \alpha_{M+i} \geq 0 \quad \text{for} \quad i = 1, \ldots, M. \tag{35}$$

The dual L1 SVM given by (4) and (5) is obtained by deleting variables α_{M+i} and C_h from the above optimization problem.

For the solution of (32) to (35), $\mathbf{x}_i$ associated with positive α_i or α_{M+j} is a support vector. However, from (28), the decision function is determined by the support vectors that satisfy $\alpha_i - \alpha_{M+i} \neq 0$.

We consider decomposing the above optimization problem into two subproblems: (1) optimizing $\alpha_i \, (i = 1, \ldots, M)$ while fixing $\alpha_{M+i} \, (i = 1, \ldots, M)$ and (2) optimizing $\alpha_{M+i} \, (i = 1, \ldots, M)$ while fixing $\alpha_i \, (i = 1, \ldots, M)$. To make this possible, we eliminate the interference between α_i and α_{M+i} in (33) by

$$\sum_{i=1}^{M} y_i \, \alpha_i = 0, \quad \sum_{i=1}^{M} y_i \, \alpha_{M+i} = 0. \tag{36}$$

Then the optimization problem given by (32) to (35) is decomposed into the following two subproblems:

Subproblem 1: Optimization of α_i

$$\text{maximize} \quad Q(\boldsymbol{\alpha}^0) = \sum_{i=1}^{M} (\alpha_i - \alpha_{M+i}) - \frac{1}{2} \sum_{i,j=1}^{M} (\alpha_i - \alpha_{M+i})(\alpha_j - \alpha_{M+j}) y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{37}$$

$$\text{subject to} \quad \sum_{i=1}^{M} y_i \alpha_i = 0, \tag{38}$$

$$C \geq \alpha_i \geq 0 \quad \text{for} \quad i = 1, \ldots, M, \tag{39}$$

where $\boldsymbol{\alpha}^0 = (\alpha_1, \ldots, \alpha_M)^\top$.

Subproblem 2: Optimization of α_{M+i}

$$\text{maximize} \quad Q(\boldsymbol{\alpha}^M) = \sum_{i=1}^{M} (\alpha_i - \alpha_{M+i}) - \frac{1}{2} \sum_{i,j=1}^{M} (\alpha_i - \alpha_{M+i})(\alpha_j - \alpha_{M+j}) y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{40}$$

$$\text{subject to} \quad \sum_{i=1}^{M} y_i \alpha_{M+i} = 0, \tag{41}$$

$$C_h \geq \sum_{i=1}^{M} \alpha_{M+i}, \tag{42}$$

$$C_h > \alpha_{M+i} \geq 0 \quad \text{for} \quad i = 1, \ldots, M, \tag{43}$$

where $\boldsymbol{\alpha}^M = (\alpha_{M+1}, \ldots, \alpha_{2M})^\top$. Here we must notice that $\alpha_{M+i} \neq C_h$. If $\alpha_{M+i} = C_h$, from (41), at least

$$\sum_{j=1,\ldots,M, y_j \neq y_i} \alpha_{M+j} = C_h \tag{44}$$

is satisfied. This contradicts (42).

We solve Subproblems 1 and 2 alternatingly until the solution converges. Subproblem 1 is very similar to the L1 SVM and can be solved by the SMO (Sequential minimal optimization). Subproblem 2, which includes the constraint (42) can also be solved by a slight modification of the SMO.

3.2. KKT Conditions

To check the convergence of Subproblems 1 and 2, we use the KKT complementarity conditions (24) to (27). However, variables h and b, which are included in the KKT conditions, are excluded from the dual problem given by (32) to (35). Therefore, as with the L1 SVM [33], to make an accurate convergence test, the exact KKT conditions that do not include h and b need to be derived.

We rewrite (24) as follows:

$$\alpha_i (y_i b - y_i F_i + \xi_i) = 0 \quad \text{for} \quad i = 1, \ldots, M, \tag{45}$$

where

$$F_i = y_i - \sum_{j=1}^{M} y_j (\alpha_j - \alpha_{M+j}) K(\mathbf{x}_i, \mathbf{x}_j). \tag{46}$$

We can classify the conditions of (45) into the following three cases:

1. $\alpha_i = 0$.

   Because $y_i b - y_i F_i + \xi_i \geq 0$ and $\xi_i = 0$,

   $$y_i b \geq y_i F_i, \quad \text{i.e.,} \quad b \geq F_i \quad \text{if} \quad y_i = 1; \quad b \leq F_i \quad \text{if} \quad y_i = -1.$$

2. $C > \alpha_i > 0$.

Because $\beta_i > 0$, $\xi_i = 0$ is satisfied. Therefore,

$$b = F_i.$$

3. $\alpha_i = C$.

Because $\beta_i = 0$, $\xi_i \geq 0$ is satisfied. Therefore,

$$y_i\, b \leq y_i\, F_i \quad \text{or} \quad b \leq F_i \quad \text{if} \quad y_i = 1; \quad b \geq F_i \quad \text{if} \quad y_i = -1.$$

Then the KKT conditions for (45) are simplified as follows:

$$\bar{F}_i \geq b \geq \tilde{F}_i \quad \text{for} \quad i = 1, \ldots, M, \tag{47}$$

where

$$\tilde{F}_i = F_i \quad \text{if} \quad (y_i = 1, \alpha_i = 0), \quad C > \alpha_i > 0 \quad \text{or} \quad (y_i = -1, \alpha_i = C), \tag{48}$$

$$\bar{F}_i = F_i \quad \text{if} \quad (y_i = -1, \alpha_i = 0), \quad C > \alpha_i > 0 \quad \text{or} \quad (y_i = 1, \alpha_i = C). \tag{49}$$

To detect the violating variables, we define b_{low} and b_{up} as follows:

$$\begin{aligned} b_{\text{low}} &= \max_i \tilde{F}_i, \\ b_{\text{up}} &= \min_i \bar{F}_i. \end{aligned} \tag{50}$$

If the KKT conditions are satisfied,

$$b_{\text{up}} \geq b_{\text{low}}. \tag{51}$$

The bias term is estimated to be

$$b_{\text{e}} = \frac{1}{2}(b_{\text{up}} + b_{\text{low}}), \tag{52}$$

where b_{e} is the estimate of the bias term using (24).

Likewise, using (46), (25) becomes

$$\alpha_{M+i}\,(h + y_i\, F_i - y_i\, b - 1) = 0 \quad \text{for} \quad i = 1, \ldots, M. \tag{53}$$

Then the conditions for (25) are rewritten as follows:

1. $\alpha_{M+i} = 0$.

From

$$h + y_i\, F_i - y_i\, b - 1 \geq 0, \tag{54}$$

we have

$$y_i\, b - h \leq y_i\, F_i - 1, \text{ i.e., } b - h \leq F_i - 1 \text{ if } y_i = 1; \; b + h \geq F_i + 1 \text{ if } y_i = -1. \tag{55}$$

2. $C_h > \alpha_{M+i} > 0$.

$$y_i\, b - h = y_i\, F_i - 1, \text{ i.e., } b - h = F_i - 1 \text{ if } y_i = 1; \; b + h = F_i + 1 \text{ if } y_i = -1.$$

The KKT conditions for (25) are simplified as follows:

$$\begin{aligned} &\text{if } y_i = -1, \quad && \bar{F}_i^{\,-} + 1 \geq b^- \geq \tilde{F}_i^{\,-} + 1, \\ &\text{if } y_i = 1, \quad && \bar{F}_i^{\,+} - 1 \geq b^+ \geq \tilde{F}_i^{\,+} - 1 \quad \text{for} \quad i = 1, \ldots, M, \end{aligned} \tag{56}$$

where $b^- = b + h$, $b^+ = b - h$, and

$$\tilde{F}_i^{\,-} = F_i + 1 \quad \text{if} \quad y_i = -1, \tag{57}$$

$$\bar{F}_i^{\,-} = F_i + 1 \quad \text{if} \quad y_i = -1, C_h > \alpha_{M+i} > 0, \tag{58}$$

$$\tilde{F}_i^{\,+} = F_i - 1 \quad \text{if} \quad y_i = 1, C_h > \alpha_{M+i} > 0, \tag{59}$$

$$\bar{F}_i^{\,+} = F_i - 1 \quad \text{if} \quad y_i = 1. \tag{60}$$

To detect the violating variables, we define b^-_{low}, b^+_{low}, b^-_{up}, and b^+_{up} as follows:

$$\begin{aligned} b^-_{\text{low}} &= \max_i \tilde{F}_i^{\,-}, \quad b^+_{\text{low}} = \max_i \tilde{F}_i^{\,+}, \\ b^-_{\text{up}} &= \min_i \bar{F}_i^{\,-}, \quad b^+_{\text{up}} = \min_i \bar{F}_i^{\,+}. \end{aligned} \tag{61}$$

In general, the distributions of Classes 1 and 2 data are different. Therefore, the upper bounds of h for Classes 1 and 2 are different. This may mean that either of b^-_{up} ($\bar{F}_i^{\,-}$) and b^+_{low} ($\tilde{F}_i^{\,+}$) may not exist. However, because of (41), both classes have at least one positive α_{M+i} each, and because of (53), the values of h for both classes can be different. This happens because we separate (33) into two equations as in (36). Then, if the KKT conditions are satisfied, both of the following inequalities hold

$$b^-_{\text{up}} \geq b^-_{\text{low}}, \quad b^+_{\text{up}} \geq b^+_{\text{low}}. \tag{62}$$

From the first inequality, the estimate of h, h^-_{e} for Class 2, is given by

$$h^-_{\text{e}} = -b_{\text{e}} + \frac{1}{2}(b^-_{\text{up}} + b^-_{\text{low}}). \tag{63}$$

From the second inequality, the estimate of h, h^+_{e} for Class 1, is given by

$$h^+_{\text{e}} = b_{\text{e}} - \frac{1}{2}(b^+_{\text{up}} + b^+_{\text{low}}). \tag{64}$$

3.3. Variant of Minimal Complexity Support Vector Machines

Subproblem 2 of the ML1 SVM is different from Subproblem 1 in that the former includes the inequality constraint given by (42). This makes the solution process more complicated. In this section, we consider making the solution process similar to that of Subproblem 1.

Solving Subproblem 2 results in obtaining h^+_{e} and h^-_{e}. We consider assigning separate variables h^+ and h^- for Classes 1 and 2 instead of a single variable h. Then the complementarity conditions for h^+ and h^- are

$$(h^+ - 1)\,\eta^+ = 0, \quad h^+ \geq 1, \quad \eta^+ \geq 0, \quad (h^- - 1)\,\eta^- = 0, \quad h^- \geq 1, \quad \eta^- \geq 0, \tag{65}$$

where η^+ and η^- are the Lagrange multipliers associated with h^+ and h^-, respectively. To simplify Subproblem 2, we assume that $\eta^+ = \eta^- = 0$. This and (41) make (41) and (42) two equality constraints. Then the optimization problem given by (40) to (43) becomes

$$\text{maximize} \quad Q(\boldsymbol{\alpha}^M) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} (\alpha_i - \alpha_{M+i})(\alpha_j - \alpha_{M+j}) y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{66}$$

$$\text{subject to} \quad \sum_{y_i=1, i=1}^{M} \alpha_{M+i} = C_h, \quad \sum_{y_i=-1, i=1}^{M} \alpha_{M+i} = C_h, \tag{67}$$

$$C_h \geq \alpha_{M+i} \geq 0 \quad \text{for} \quad i = 1, \ldots, M. \tag{68}$$

Here, (41) is not necessary because of (67). We call the above architecture ML1_v SVM.

For the solution of the ML1 SVM, the same solution is obtained by the ML1_v SVM with the C_h value given by

$$C_h = \sum_{i=1,\ldots,M, y_i=1} \alpha_{M+i} = \sum_{i=1,\ldots,M, y_i=-1} \alpha_{M+i}. \tag{69}$$

However, the reverse is not true, namely, the solution of the ML1_v SVM may not be obtained by the ML1 SVM. As the C_h value becomes large, the value of η becomes positive for the ML1 SVM, but for the ML1_v SVM, the values of α_{M+i} are forced to become larger.

4. Training Methods

In this section we extend the training method for the L1 SVM that fuses SMO and Newton's method [31] for training the ML1 SVM. The major part of the training method consists of calculation of corrections by Newton's method and the working set selection method. The training method of the ML1_v SVM is similar to that of the L1 SVM. Therefore, we only explain the difference of the methods in Section 4.3.

4.1. Calculating Corrections by Newton's Method

In this subsection, we discuss the corrections by Newton's method for two subprograms.

4.1.1. Subprogram 1

First we discuss optimization of Subproblem 1. We optimize the variables α_i in the working set $\{\alpha_i | i \in W, i \in \{1, \ldots, M\}\}$, where W includes working set indices, fixing the remaining variables, by

$$\text{maximize} \quad Q(\boldsymbol{\alpha}_W) = \sum_{i \in W} \alpha_i - \frac{1}{2} \sum_{i \in W, j=1,\ldots,M} (\alpha_i - \alpha_{M+i})(\alpha_j - \alpha_{M+j}) y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{70}$$

$$\text{subject to} \quad \sum_{i \in W} y_i \alpha_i = -\sum_{i \notin W} y_i \alpha_i, \tag{71}$$

$$C \geq \alpha_i \geq 0, \quad \text{for } i \in W. \tag{72}$$

Here $\boldsymbol{\alpha}_W = (\ldots, \alpha_i, \ldots)^\top$, $i \in W$.

We can solve the above optimization problem by the method discussed in [31]. We select α_s in the working set and solve (71) for α_s:

$$\alpha_s = -\sum_{i \neq s, i=1}^{M} y_s y_i \alpha_i. \tag{73}$$

Substituting (73) into (70), we eliminate the equality constraint. Let $\boldsymbol{\alpha}_{W'} = (\ldots, \alpha_i, \ldots)^\top$ $(i \neq s, i \in W)$. Now because $Q(\boldsymbol{\alpha}_{W'})$ is quadratic, we can express the change of $Q(\boldsymbol{\alpha}_{W'})$, $\Delta Q(\boldsymbol{\alpha}_{W'})$, as a function of the change of $\boldsymbol{\alpha}_{W'}$, $\Delta\boldsymbol{\alpha}_{W'}$, by

$$\Delta Q(\boldsymbol{\alpha}_{W'}) = \frac{\partial Q(\boldsymbol{\alpha}_{W'})}{\partial \boldsymbol{\alpha}_{W'}} \Delta\boldsymbol{\alpha}_{W'} + \frac{1}{2} \Delta\boldsymbol{\alpha}_{W'}^\top \frac{\partial^2 Q(\boldsymbol{\alpha}_{W'})}{\partial \boldsymbol{\alpha}_{W'}^2} \Delta\boldsymbol{\alpha}_{W'}. \tag{74}$$

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

$$\Delta \boldsymbol{\alpha}_{W'} = -\left(\frac{\partial^2 Q(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}_{W'}^2}\right)^{-1} \frac{\partial Q(\boldsymbol{\alpha}_{W'})}{\partial \boldsymbol{\alpha}_{W'}}, \tag{75}$$

where

$$\frac{\partial Q(\boldsymbol{\alpha}_{W'})}{\partial \alpha_i} = \frac{\partial Q(\boldsymbol{\alpha}_W)}{\partial \alpha_i} + \frac{\partial Q(\boldsymbol{\alpha}_W)}{\partial \alpha_s}\frac{\partial \alpha_s}{\partial \alpha_i} = y_i\,(F_i - F_s) \quad \text{for } i \in W', \tag{76}$$

$$\begin{aligned}\frac{\partial^2 Q(\boldsymbol{\alpha}_{W'})}{\partial \alpha_i \partial \alpha_j} &= \frac{\partial^2 Q(\boldsymbol{\alpha}_W)}{\partial \alpha_i \partial \alpha_j} + \frac{\partial^2 Q(\boldsymbol{\alpha}_W)}{\partial \alpha_s \partial \alpha_j}\frac{\partial \alpha_s}{\partial \alpha_i} + \frac{\partial^2 Q(\boldsymbol{\alpha}_W)}{\partial \alpha_i \partial \alpha_s}\frac{\partial \alpha_s}{\partial \alpha_j} + \frac{\partial^2 Q(\boldsymbol{\alpha}_W)}{\partial \alpha_s \partial \alpha_s}\frac{\partial \alpha_s}{\partial \alpha_i}\frac{\partial \alpha_s}{\partial \alpha_j} \\ &= y_i\,y_j\,\left(-K(\mathbf{x}_i,\mathbf{x}_j) + K(\mathbf{x}_i,\mathbf{x}_s) + K(\mathbf{x}_s,\mathbf{x}_j) - K(\mathbf{x}_s,\mathbf{x}_s)\right) \quad \text{for } i,j \in W'. \end{aligned} \tag{77}$$

Here, the partial derivative of Q with substitution of (73) is calculated by the chain rule without substitution.

We assume that $-\partial^2 Q(\boldsymbol{\alpha})/\partial \boldsymbol{\alpha}_{W'}^2$ is positive definite. If not, we avoid matrix singularity adding a small value to the diagonal elements.

Then from (73) and (75), we obtain the correction of α_s:

$$\Delta \alpha_s = -\sum_{i \in W'} y_s\,y_i\,\Delta \alpha_i. \tag{78}$$

For α_i $(i \in W)$, if

$$\alpha_i = 0, \quad \Delta \alpha_i < 0 \quad \text{or} \quad \alpha_i = C, \quad \Delta \alpha_i > 0, \tag{79}$$

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 $\alpha_i + \Delta \alpha_i < 0$, $\Delta \alpha_i' = -\alpha_i$. Moreover, if $\alpha_i + \Delta \alpha_i > C$, $\Delta \alpha_i' = C - \alpha_i$. Otherwise $\Delta \alpha_i' = \Delta \alpha_i$. Then we calculate

$$r = \min_{i \in W}\left|\frac{\Delta \alpha_i'}{\Delta \alpha_i}\right|, \tag{80}$$

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

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

$$\boldsymbol{\alpha}_W^{\text{new}} = \boldsymbol{\alpha}_W^{\text{old}} + r\,\Delta \boldsymbol{\alpha}_W. \tag{81}$$

4.1.2. Subprogram 2

We optimize the variables α_{M+i} in the working set $\{\alpha_{M+i} | i \in W, i \in \{1,\ldots,M\}\}$, fixing the remaining variables, by

$$\begin{aligned} \text{maximize} \quad & Q(\boldsymbol{\alpha}^M) = -\sum_{i \in W} \alpha_{M+i} \\ & -\frac{1}{2}\sum_{i \in W,\, j=1,\ldots,M} (\alpha_i - \alpha_{M+i})\,(\alpha_j - \alpha_{M+j})\,y_i\,y_j\,K(\mathbf{x}_i,\mathbf{x}_j) \end{aligned} \tag{82}$$

$$\text{subject to} \quad \sum_{i \in W} y_i\,\alpha_{M+i} = -\sum_{i \notin W} y_i\,\alpha_{M+i}, \tag{83}$$

$$C_h - \sum_{i=1}^{M} \alpha_{M+i} = \eta \geq 0, \tag{84}$$

$$C_h > \alpha_{M+i} \geq 0 \quad \text{for } i \in W. \tag{85}$$

We select α_{M+s} in the working set W and solve (83) for α_{M+s}:

$$\alpha_{M+s} = -\sum_{i=1, i\neq s}^{M} y_s\, y_i\, \alpha_{M+i}. \tag{86}$$

Then similar to Subproblem 1, we substitute (86) into (82) and eliminate α_{M+s}. The partial derivatives in (75) are as follows:

$$\begin{aligned} \frac{\partial Q(\boldsymbol{\alpha}_{W'})}{\partial \alpha_{M+i}} &= \frac{\partial Q(\boldsymbol{\alpha}_W)}{\partial \alpha_{M+i}} + \frac{\partial Q(\boldsymbol{\alpha}_W)}{\partial \alpha_{M+s}} \frac{\partial \alpha_{M+s}}{\partial \alpha_{M+i}} = -y_i\,(F_i - F_s) \quad \text{for } i \in W', & (87)\\ \frac{\partial^2 Q(\boldsymbol{\alpha}_{W'})}{\partial \alpha_{M+i} \partial \alpha_{M+j}} &= \frac{\partial^2 Q(\boldsymbol{\alpha}_W)}{\partial \alpha_{M+i} \partial \alpha_{M+j}} + \frac{\partial^2 Q(\boldsymbol{\alpha}_W)}{\partial \alpha_{M+s} \partial \alpha_{M+j}} \frac{\partial \alpha_{M+s}}{\partial \alpha_{M+i}} + \frac{\partial^2 Q(\boldsymbol{\alpha}_W)}{\partial \alpha_{M+i} \partial \alpha_{M+s}} \frac{\partial \alpha_{M+s}}{\partial \alpha_{M+j}} \\ &= y_i\, y_j\, (-K(\mathbf{x}_i, \mathbf{x}_j) + K(\mathbf{x}_i, \mathbf{x}_s) + K(\mathbf{x}_s, \mathbf{x}_j) - K(\mathbf{x}_s, \mathbf{x}_s)) \quad \text{for } i, j \in W'. & (88) \end{aligned}$$

From (86), we obtain the correction of α_{M+s}:

$$\Delta\alpha_{M+s} = -\sum_{i \in W'} y_s\, y_i\, \Delta\alpha_{M+i}. \tag{89}$$

Now we consider the constraint (84). If $\eta = 0$, the sum of corrections needs to be zero. This is achieved if

$$y_s = y_i \quad \text{for } i \in W'. \tag{90}$$

Namely, we select the working set from the same class.

For α_{M+i} $(i \in W)$, if

$$\alpha_{M+i} = 0, \quad \Delta\alpha_{M+i} < 0, \tag{91}$$

we delete these variables from the working set and repeat the procedure for the reduced working set. Let $\Delta\alpha'_{M+i}$ be the maximum or minimum correction of α_{M+i} that is within the bounds. Then if $\alpha_{M+i} + \Delta\alpha_{M+i} < 0$, $\Delta\alpha'_{M+i} = -\alpha_{M+i}$. Otherwise $\Delta\alpha'_{M+i} = \Delta\alpha_{M+i}$. Then we calculate

$$r = \min_{i \in W} \left| \frac{\Delta\alpha'_{M+i}}{\Delta\alpha_{M+i}} \right|, \tag{92}$$

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

If $r \sum_{i \in W} \Delta\alpha_{M+i} > \eta > 0$, we further calculate

$$r' = \frac{\eta}{r \sum_{i \in W} \Delta\alpha_{M+i}}. \tag{93}$$

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

$$\boldsymbol{\alpha}_W^{\text{new}} = \boldsymbol{\alpha}_W^{\text{old}} + r\, \Delta\boldsymbol{\alpha}_W, \tag{94}$$

where r is replaced by $r'\, r$ if $r \sum_{i \in W} \alpha_{M+i} > \eta > 0$.

4.2. Working Set Selection

At each iteration of training, we optimize either Subproblem 1 (α_i) or Subproblem 2 (α_{M+i}). To do so, we define the most violating indices as follows:

$$V_1 = b_{\text{low}} - b_{\text{up}}, \quad V_2 = b^+_{\text{low}} - b^+_{\text{up}}, \quad V_3 = b^-_{\text{low}} - b^-_{\text{up}}. \tag{95}$$

We consider that training is converged when

$$\max_i V_i \leq \tau, \tag{96}$$

where τ is a small positive parameter.

If (96) is not satisfied, we correct variables associated with V_i where i is determined by

$$\arg\max_i V_i. \tag{97}$$

According to the conditions of (96) and (97), we determine the variable pair as follows:

1. If V_1 is the maximum in (97), we optimize Subproblem 1 (α_i). Let the variable pair associated with b_{up} and b_{low} be $\alpha_{i_{\text{min}}}$ and $\alpha_{i_{\text{max}}}$, respectively.
2. If $\eta = 0$ and either V_2 or V_3 is the maximum in (97), or if $\eta \neq 0$ and either V_2 or V_3 exceeds τ but not both, we optimize Subproblem 2 (α_{M+i} belonging to either Class 1 or 2). Let the variable pair associated with b^k_{up} and b^k_{low} (k is either $+$ or $-$) be $\alpha_{M+i^k_{\text{min}}}$ and $\alpha_{M+i^k_{\text{max}}}$, respectively.
3. If $\eta \neq 0$ and both V_2 and V_3 exceed τ, we optimize Subproblem 2 (α_{M+i} selected from Classes 1 and 2). Let the variable pair be $\alpha_{M+i^-_{\text{min}}}$ and $\alpha_{M+i^+_{\text{max}}}$. This is to make the selected variables correctable as will be shown in Section 4.4.2.

For the ML1$_{\text{v}}$ SVM, $\eta = 0$. Therefore, Case (3) is not necessary.

Because the working set selection strategies for α_i and α_{M+i} are the same, in the following we only discuss the strategy for α_i.

In the first order SMO, at each iteration step, $\alpha_{i_{\text{min}}}$ and $\alpha_{i_{\text{max}}}$ that violate the KKT conditions the most are selected for optimization. This guarantees the convergence of the first order SMO [33]. In the second order SMO, the pair of variables that maximize the objective function are selected. However, to reduce computational burden, fixing $\alpha_{i_{\text{min}}}$, the variable that maximizes the objective function is searched [34]:

$$i_{\text{2nd}} = \arg\max_{i \in V_{\text{KKT}}} \Delta Q(\alpha_i, \alpha_{i_{\text{min}}}), \tag{98}$$

where V_{KKT} is the set of indices that violate the KKT conditions:

$$V_{\text{KKT}} = \{i | b_{\text{up}} < \tilde{F}_i - \tau \quad \text{or} \quad b_{\text{low}} < \bar{F}_i + \tau\} \quad \text{for} \quad i \in \{1, \ldots, M\}\}. \tag{99}$$

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

Because the second order SMO accelerates training for a large C value [35], in the following we use the second order SMO. However, for a substantially 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, in addition to the SMO variables, we add variables that are selected in the previous steps as SMO variables, into the working set.

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 loop. To avoid obtaining an infeasible solution by adding loop variables to the working set, we restrict loop variables to be unbounded support vectors, i.e., $\alpha_i \neq C$ (This happens only for Subprogram 1).

Because we optimize Subprograms 1 and 2 alternatingly, loop variables may include those for Subprograms 1 and 2. Therefore, in optimizing Subprogram 1, we need to exclude the loop variables for Subproblem 2; and vice versa. In addition, in optimizing Subproblem 2 with $\eta = 0$, we need to exclude variables belonging to the unoptimized class, in addition to those for Subproblem 1.

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. For the detailed procedure, please refer to [31].

4.3. Training Procedure of ML1 SVM

In the following we show the training procedure of the ML1 SVM.

1. (Initialization) Select α_i and α_j in the opposite classes and set $\alpha_i = \alpha_j = C$, $\alpha_k = 0, k \neq i, j, k = 1, \ldots, M$, $\alpha_{M+i} = \alpha_{M+j} = a\,C_h$, and $\alpha_{M+k} = 0, k \neq i, j, k = 1, \ldots, M$, where $a \leq 0.5$.
2. (Corrections) If Pr1 (Program 1), calculate partial derivatives (76) and (77) and calculate corrections by (75). Then, modify the variables by (81). Else, if Pr2, calculate partial derivatives (87) and (88) and calculate corrections by (75). Then, modify the variables by (94).
3. (Convergence Check) Update F_i and calculate b_{up}, b_{low}, b^+_{up}, b^+_{low}, b^-_{up}, and b^-_{low}. If (96) is satisfied, stop training. Otherwise if V_1 is the maximum, select Pr1, otherwise, Pr2. Calculate the SMO variables.
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 Subproblem 2 of the ML1_v SVM, data for Class 1 and Class 2 can be optimized separately. Therefore, because the data that are optimized belong to the same class, the sum of corrections is zero. Thus, in Step 1, $a = 1$. In Step 3, if Pr2 is optimized, the variables associated with V_2 or V_3 are optimized, not both.

4.4. Convergence Proof

Convergence of the first order SMO for the L1 SVM is proved in [33]. Similarly, we can prove that the training procedure discussed in Section 4.3 converges to the unique solution. In the following we prove the convergence for the ML1 SVM. The proof for the ML1_v SVM is evident from the discussion.

Subprograms 1 and 2 are quadratic programming problems and thus have unique maximum solutions. Therefore, it is sufficient to prove that the objective function increases by the first order SMO. For the second order SMO, the increase of the objective function is also guaranteed because the variable pair that gives the largest increase of the objective function is selected. By combining SMO with Newton's method, if some variables are not correctable, they are deleted from the working set. By this method, in the worst case, only the SMO variables remain in the working set. Therefore, we only need to show that the first order SMO converges for the ML1 SVM.

4.4.1. Convergence Proof for Subprogram 1

From (48), $\alpha_{i_{\max}}$ satisfies

$$C > \alpha_{i_{\max}} \geq 0 \quad \text{for} \quad y_{i_{\max}} = 1, \quad C \geq \alpha_{i_{\max}} > 0 \quad \text{for} \quad y_{i_{\max}} = -1. \tag{100}$$

Moreover, from (49), $\alpha_{i_{\min}}$ satisfies

$$C \geq \alpha_{i_{\min}} > 0 \quad \text{for} \quad y_{i_{\min}} = 1, \quad C > \alpha_{i_{\min}} \geq 0 \quad \text{for} \quad y_{i_{\min}} = -1. \tag{101}$$

Because the KKT conditions are not satisfied, $F_{i_{\max}} > F_{i_{\min}}$. Moreover, we set $\alpha_s = \alpha_{i_{\min}}$. Then from (76) and (77),

$$\frac{\partial Q(\alpha_{i_{\max}})}{\partial \alpha_{i_{\max}}} = y_{i_{\max}} (F_{i_{\max}} - F_{i_{\min}}), \tag{102}$$

$$\frac{\partial^2 Q(\alpha_{i_{\max}})}{\partial \alpha^2_{i_{\max}}} = -\|\boldsymbol{\phi}(\mathbf{x}_{i_{\max}}) - \boldsymbol{\phi}(\mathbf{x}_{i_{\min}})\|^2 \leq 0, \tag{103}$$

where in (103), if the equality holds, we replace zero with a small negative value to avoid zero division in (75).

Then from (102), (103), and (75), the signs of the corrections $\Delta\alpha_{i_{\min}}$ and $\Delta\alpha_{i_{\max}}$ are given by

1. $\Delta\alpha_{i_{\max}} > 0$ and $\Delta\alpha_{i_{\min}} < 0$ for $y_{i_{\min}} = y_{i_{\max}} = 1$,
2. $\Delta\alpha_{i_{\max}} < 0$ and $\Delta\alpha_{i_{\min}} > 0$ for $y_{i_{\min}} = y_{i_{\max}} = -1$,
3. $\Delta\alpha_{i_{\max}} < 0$ and $\Delta\alpha_{i_{\min}} < 0$ for $y_{i_{\min}} = 1,\ y_{i_{\max}} = -1$,
4. $\Delta\alpha_{i_{\max}} > 0$ and $\Delta\alpha_{i_{\min}} > 0$ for $y_{i_{\min}} = -1,\ y_{i_{\max}} = 1$.

From (100) and (101), the above corrections are all possible. For instance, in (1) $\Delta\alpha_{i_{\max}} > 0$ and $C > \alpha_{i_{\max}} \geq 0$ for $y_{i_{\max}} = 1$. Therefore, $C \geq \alpha_{i_{\max}} + \Delta\alpha_{i_{\max}} > 0$.

Because the corrections are not zero, the objective function increases.

4.4.2. Convergence Proof for Subprogram 2

From (57) and (58), $\alpha_{M+i^-_{\min}}$ and $\alpha_{M+i^-_{\max}}$ satisfy

$$C_h > \alpha_{M+i^-_{\min}} > 0, \quad C_h > \alpha_{M+i^-_{\max}} \geq 0, \tag{104}$$

respectively. Likewise, from (59) and (60), $\alpha_{M+i^+_{\min}}$ and $\alpha_{M+i^+_{\max}}$ satisfy

$$C_h > \alpha_{M+i^+_{\min}} \geq 0, \quad C_h > \alpha_{M+i^+_{\max}} > 0, \tag{105}$$

respectively.

We set $\alpha_{M+s} = \alpha_{M+i^k_{\min}}$. Then from (87) and (88),

$$\frac{\partial Q(\alpha_{M+i^k_{\max}})}{\partial \alpha_{M+i^k_{\max}}} = -y_{i^k_{\max}}(F_{i^k_{\max}} - F_{i^k_{\min}}), \quad \frac{\partial^2 Q(\alpha_{M+i^k_{\max}})}{\partial (\alpha_{M+i^k_{\max}})^2} \leq 0, \tag{106}$$

If we correct $\alpha_{M+i^-_{\max}}$ and $\alpha_{M+i^-_{\min}}$ $(y_{i^-_{\max}} = y_{i^-_{\min}} = -1)$, $\Delta\alpha_{M+i^-_{\max}} > 0$ and $\Delta\alpha_{M+i^-_{\min}} < 0$. From (104), this correction is possible.

Likewise, if we correct $\alpha_{M+i^+_{\max}}$ and $\alpha_{M+i^+_{\min}}$ $(y_{i^+_{\max}} = y_{i^+_{\min}} = 1)$, $\Delta\alpha_{M+i^+_{\max}} < 0$ and $\Delta\alpha_{M+i^+_{\min}} > 0$. From (105), this correction is possible.

If both V_2 and V_3 are larger than τ, we select $\alpha_{M+i^-_{\min}}$ and $\alpha_{M+i^+_{\max}}$. Then from (104) and (105), these variables are correctable whether they be increased or decreased.

Because the corrections are not zero, the objective function increases.

5. Computer Experiments

First we analyze the behavior of the ML1 SVM and ML1_v SVM using a two-dimensional iris dataset and then to examine the superiority of the proposed classifiers over the L1 SVM, we evaluate their generalization abilities and computation time using two-class and multiclass problems. All the programs used in the performance evaluation were coded in Fortran and tested using Windows machines.

5.1. Analysis of Behaviors

We analyzed the behaviors of the ML1 SVM and ML1_v SVM using the iris dataset [36], which is frequently used in the literature. The iris dataset consists of 150 data with three classes and four features. We used Classes 2 and 3 and Features 3 and 4. For both classes, the first 25 data were used for training and the remaining data were used for testing. We used linear kernels: $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$.

We evaluated the h^+ and h^- values for the change of the C_h value from 0.1 to 2000 fixing C to 1000. Figure 1 shows the result for the ML1_v SVM. Both h^+ and h^- values are constant for $C_h = 0.1$ to 100 and they decrease as the C_h value is increased. For the ML1 SVM, the h^+ and h^- values

are constant for the change of C_h and are the same as those of the ML1 SVM with $C_h = 0.1$ to 100. For $C_h = 2000$, $\eta = 1960.00$. Thus, $\sum_{y_i=1,i=1,\ldots,M} \alpha_{M+i} = 20.00$. For $C_h = 10{,}000$, $\eta = 9732.75$. Thus, $\sum_{y_i=1,i=1,\ldots,M} \alpha_{M+i} = 133.63$. This means that C_h value is too small to obtain the solution comparable to that of the $\mathrm{ML1_v}$ SVM. Therefore, the ML1 SVM is insensitive to the C_h value compared to the $\mathrm{ML1_v}$ SVM.

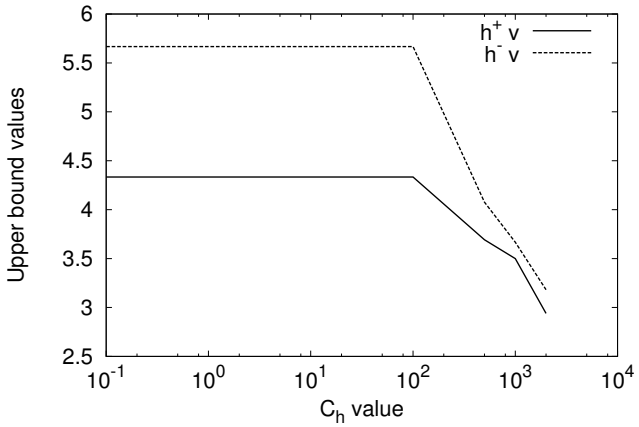

Figure 1. Upper bounds h^+ and h^- for the change of the C_h value with $C = 1000$.

Figure 2 shows the h^+ and h^- values for $C = 10$. For the $\mathrm{ML1_v}$ SVM, the h^+ and h^- values become smaller than 1 for C_h larger than 100. This means that the C_h value is so large that the solution is no longer valid. However, for the ML1 SVM, the h^+ and h^- values are larger than 1 and thus valid solutions are obtained for $C_h = 0.1$ to 2000. This is because $\sum_{y_i=1,i=1,\ldots,M} \alpha_{M+i} = 58.96$ even at $C_h = 2000$.

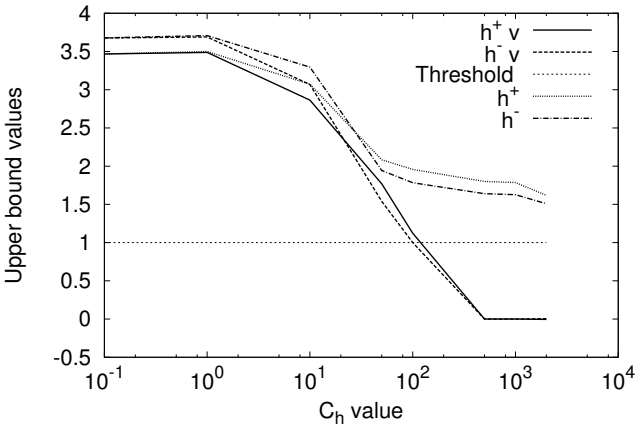

Figure 2. Upper bounds h^+ and h^- for the change of the C_h value with $C = 10$.

Figure 3 shows the decision boundary of the $\mathrm{ML1_v}$ SVM for $C = 1000$ and $C_h = 0.1$, 1000. For $C_h = 0.1$, the decision boundary is almost parallel to the x_1 axis. However, for $C_h = 1000$, the decision boundary rotates in the clockwise direction to make the data more condensed to the decision boundary. The accuracy for the test data is 92% for $C_h = 0.1$ and is increased to 94% for $C_h = 1000$.

From the above experiment we confirmed that the solutions of the ML1 SVM and the $\mathrm{ML1_v}$ SVM are the same for small C_h values but for large C_h values both are different and in extreme cases the solution of the $\mathrm{ML1_v}$ SVM may be infeasible.

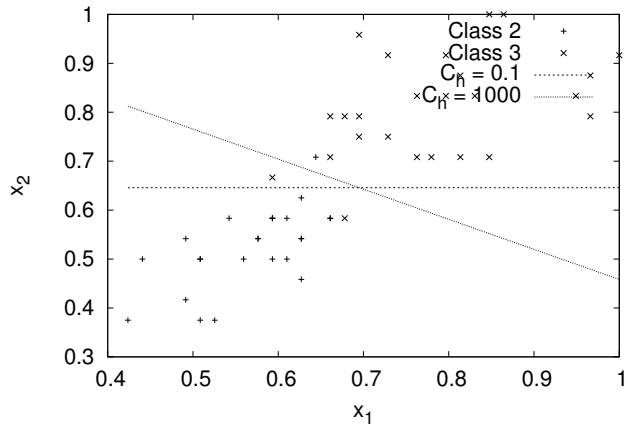

Figure 3. Separating hyperplanes for the different C_h values with $C = 1000$.

5.2. Performance Comparison

In this section, we compare the generalization performance of the ML1 SVM and $ML1_v$ SVM with the L1 SVM, MLP SVM [29], LS SVM, and ULDM [20] using two-class and multiclass problems. Our main goal is to show that the generalization abilities of the ML1 SVM and $ML1_v$ SVM are better than the generalization ability of the L1 SVM.

5.2.1. Comparison Conditions

We determined the hyperparameter values using the training data by fivefold cross-validation, trained the classifier with the determined hyperparameter values, and evaluated the accuracy for the test data.

We trained the ML1 SVM, $ML1_v$ SVM, and L1 SVM by SMO combined with Newton's method. We trained the MLP SVM by the simplex method and the LS SVM and ULDM by matrix inversion.

We used RBF kernels:

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2 / m), \tag{107}$$

where γ is the parameter to control the spread of the radius and m is the number of inputs to normalize the kernel, and polynomial kernels including linear kernels:

$$K(\mathbf{x}, \mathbf{x}') = (P + \mathbf{x}^\top \mathbf{x}')^d, \tag{108}$$

where $P = 0$ and $d = 1$ for linear kernels and $P = 1$ and $d = 2, 3, \ldots$ for polynomial kernels.

In cross-validation, we selected the γ values for RBF kernels from $\{0.01, 0.1, 0.5, 1, 5, 10, 15, 20, 50, 100, 200\}$, the d values for polynomial kernels from $\{1, 2, \ldots, 8\}$, and the C and C_h values from $\{0.1, 1, 10, 50, 100, 500, 1000, 2000\}$. For the ULDM, the C value was selected from $\{0.1, 1, 10, 100, 1000, 10^4, 10^6, 10^8\}$ [21]. The value of τ in (99) was set to 0.01 for the ML1 SVM, $ML1_v$ SVM, and L1 SVM.

For the L1 SVM, LS SVM, and ULDM, we determined the $\gamma\,(d)$ and C values by grid search. For the ML1 SVM, $ML1_v$ SVM, and MLP SVM, we need to determined the value of C_h in addition to $\gamma\,(d)$ and C values. (For the MLP SVM we evaluated the performance using only RBF kernels). To shorten computation time in such a situation, first we determined the $\gamma\,(d)$ and C values with $C_h = 1$ ($C_h = 0.1$ for the MLP SVM) by grid search and then we determined the C_h value by line search fixing the $\gamma\,(d)$ and C values with the determined values.

After model selection, we trained the classifier with the determined hyperparameter values and calculated the accuracy for the test data. For two-class problems, which have multiple sets of training and test pairs, we calculated the average accuracies and their standard deviations, and performed Welch's t test with the confidence level of 5%.

5.2.2. Two-Class Problems

Table 1 lists the two-class problems used [37], which were generated using the datasets from the UC Irvine Machine Learning Repository [38]. In the table, the numbers of input variables, training data, test data, and training and test data pair sets are listed. The table also includes the maximum average prior probability shown in the accuracy (%). The numeral in parentheses shows that for the test data. According to the prior probabilities, the class data are relatively well balanced and there are not much differences between training and test data.

Table 1. Benchmark datasets for two-class problems.

Problem	Inputs	Training Data	Test Data	Sets	Prior (%)
Banana	2	400	4900	100	54.62 (55.21)
Breast cancer	9	200	77	100	70.59 (71.19)
Diabetes	8	468	300	100	65.03 (65.22)
Flare-solar	9	666	400	100	55.25 (55.27)
German	20	700	300	100	69.92 (70.18)
Heart	13	170	100	100	55.53 (55.60)
Image	18	1300	1010	20	57.40 (56.81)
Ringnorm	20	400	7000	100	50.27 (50.50)
Splice	60	1000	2175	20	51.71 (52.00)
Thyroid	5	140	75	100	69.51 (70.25)
Titanic	3	150	2051	100	67.83 (67.69)
Twonorm	20	400	7000	100	50.52 (50.01)
Waveform	21	400	4600	100	66.90 (67.07)

Table 2 shows the evaluation results using RBF kernels. In the table, for each classifier and each classification problem, the average accuracy and the standard deviation are shown. For each problem the best average accuracy is shown in bold and the worst, underlined. The "+" and "−" symbols at the accuracy show that the ML1 SVM is statistically better and worse than the classifier associated with the attached symbol, respectively. For instance, the ML1 SVM is statistically better than the ULDM for the flare-solar problem. The "Average" row shows the average accuracy of the 13 problems for each classifier and "B/S/W" denotes the number of times that the associated classifier show the best, the second best, and the worst accuracy. The "W/T/L" row denotes the number of times that the ML1 SVM is statistically better than, comparable to, and worse than the associated classifier.

According to the "W/T/L" row, the ML1 SVM is statistically better than the MLP SVM but is slightly inferior to ULDM. It is comparable to the remaining classifiers. From the "Average" measure, the ULDM is the best, the $ML1_v$ SVM, the second, the L1 SVM and the ML1 SVM, the third. From the "B/S/W" measure, the ULDM is the best and the LS SVM is the second best.

Table 3 shows the results for polynomial kernels. For each problem the best average accuracy is shown in bold and the worst, underlined. We do not include the MLP SVM for comparison. From the table, the ML1 SVM is statistically comparable to the L1 SVM, slightly better than $ML1_v$ SVM, and better than the LS SVM and ULDM. From the average accuracies, the L1 SVM is the best, ML1 SVM, the second best, and the ULDM, the worst.

The $ML1_v$ SVM is statistically inferior to ML1 SVM for the cancer and splice problems. We study why this happened. For the second file of the breast-cancer problem, the accuracy for the test data is 67.53% by the $ML1_v$ SVM, which is by 7.79% lower than by the ML1 SVM. The parameter values are selected as $d = 2$, $C = 1$, and $C_h = 10$. For the C_h value higher than or equal to 50, the accuracy is 75.32%, which is the same as that by ML1 SVM. Therefore, model selection did not work properly. For the 17th file of the splice problem, the accuracy for the $ML1_v$ SVM is 83.68% with $d = 1$, and $C = C_h = 1$. This is caused by $C_h = 1$ when the d and C values are determined by grid search. If $C_h = 0.1$, by model selection $d = 2$, $C = C_h = 0.1$ are obtained, and the accuracy is 87.36%, which is better than

87.22% by the ML1 SVM. Therefore, in this case also, the low accuracy was obtained by the problem of model selection.

Table 2. Accuracies of the test data for the two-class problems using RBF kernels.

Problem	ML1 SVM	$ML1_v$ SVM	L1 SVM	MLP SVM	LS SVM	ULDM
Banana	**89.18** $\pm$ 0.70	89.10 $\pm$ 0.70	89.17 $\pm$ 0.72	89.07 $\pm$ 0.73	89.17 $\pm$ 0.66	89.13 $\pm$ 0.72
Cancer	73.03 $\pm$ 4.45	73.12 $\pm$ 4.43	73.03 $\pm$ 4.51	72.81 $\pm$ 4.59	73.13 $\pm$ 4.68	**73.82**$^-$ $\pm$ 4.44
Diabetes	76.17 $\pm$ 2.25	76.33 $\pm$ 1.94	76.29 $\pm$ 1.73	76.05 $\pm$ 1.74	76.19 $\pm$ 2.00	**76.50** $\pm$ 1.94
Flare-solar	66.98 $\pm$ 2.14	**66.99** $\pm$ 2.16	**66.99** $\pm$ 2.12	66.62 $\pm$ 3.10	66.25$^+$ $\pm$ 1.98	66.34$^+$ $\pm$ 1.94
German	75.91 $\pm$ 2.03	75.97 $\pm$ 2.21	75.95 $\pm$ 2.24	75.63 $\pm$ 2.57	76.10 $\pm$ 2.10	**76.15** $\pm$ 2.29
Heart	82.84 $\pm$ 3.26	**82.96** $\pm$ 3.25	82.82 $\pm$ 3.37	82.52 $\pm$ 3.27	82.49 $\pm$ 3.60	82.70 $\pm$ 3.66
Image	97.29 $\pm$ 0.44	97.29 $\pm$ 0.47	97.16 $\pm$ 0.41	96.47$^+$ $\pm$ 0.87	**97.52** $\pm$ 0.54	97.15 $\pm$ 0.68
Ringnorm	98.12 $\pm$ 0.36	97.97 $\pm$ 1.11	98.14 $\pm$ 0.35	97.97$^+$ $\pm$ 0.37	**98.19** $\pm$ 0.33	98.16 $\pm$ 0.35
Splice	89.05 $\pm$ 0.83	88.99 $\pm$ 0.83	88.89 $\pm$ 0.91	86.71$^+$ $\pm$ 1.27	88.98 $\pm$ 0.70	**89.13** $\pm$ 0.60
Thyroid	95.32 $\pm$ 2.41	**95.37** $\pm$ 2.50	95.35 $\pm$ 2.44	95.12 $\pm$ 2.38	95.08 $\pm$ 2.55	95.29 $\pm$ 2.34
Titanic	77.37 $\pm$ 0.81	77.40 $\pm$ 0.79	77.39 $\pm$ 0.74	**77.41** $\pm$ 0.77	77.39 $\pm$ 0.83	77.40 $\pm$ 0.85
Twonorm	97.36 $\pm$ 0.28	97.38 $\pm$ 0.25	97.38 $\pm$ 0.26	97.13$^+$ $\pm$ 0.29	**97.43** $\pm$ 0.27	**97.43** $\pm$ 0.25
Waveform	89.72 $\pm$ 0.73	89.67 $\pm$ 0.75	89.76 $\pm$ 0.66	89.39$^+$ $\pm$ 0.53	90.05$^-$ $\pm$ 0.59	**90.24**$^-$ $\pm$ 0.50
Average (B/S/W)	85.26 (1/3/1)	85.27 (3/3/1)	85.26 (1/2/0)	84.84 (1/0/9)	85.23 (3/4/3)	**85.34** (6/2/0)
W/T/L	—	0/13/0	0/13/0	5/8/0	1/11/1	1/10/2

For each problem the best average accuracy is shown in bold and the worst underlined. The "+" and "−" symbolsatthe accuracy show that the ML1 SVM is statistically better and worse than the classifier associated with the attached symbol, respectively.

The ULDM performs worse than the ML1 SVM for banana and thyroid problems. For the banana problem by the ULDM, the average accuracy is by 6.66% lower than by the ML1 SVM. This was caused by mal-selection of the parameter ranges. By setting $C = \{10^8, 10^{10}, 10^{12}, 10^{14}, 10^{16}\}$ and $d = \{6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$, the average accuracy and standard deviation are 88.35 $\pm$ 1.10%, which are statistically comparable to those by the ML1 SVM. However, for the thyroid problem, the change of the parameter ranges does not improve the average accuracy much.

Table 3. Accuracies of the test data for the two-class problems using polynomial kernels.

Problem	ML1 SVM	$ML1_v$ SVM	L1 SVM	LS SVM	ULDM
Banana	88.97 $\pm$ 0.69	**89.01** $\pm$ 0.62	**89.01** $\pm$ 0.62	88.07$^+$ $\pm$ 1.00	82.31$^+$ $\pm$ 2.49
Cancer	**72.90** $\pm$ 5.10	71.66$^+$ $\pm$ 4.86	72.84 $\pm$ 5.25	72.75 $\pm$ 4.61	72.75 $\pm$ 4.71
Diabetes	76.19 $\pm$ 1.75	76.22 $\pm$ 1.67	76.29 $\pm$ 1.75	**76.39** $\pm$ 1.91	76.29 $\pm$ 1.66
Flare-solar	**67.30** $\pm$ 2.01	67.26 $\pm$ 2.12	67.19 $\pm$ 2.17	66.46$^+$ $\pm$ 1.92	67.09 $\pm$ 1.97
German	75.62 $\pm$ 2.16	75.71 $\pm$ 2.23	**75.79** $\pm$ 2.26	75.70 $\pm$ 2.05	75.23$^+$ $\pm$ 1.92
Heart	82.77 $\pm$ 3.62	82.93 $\pm$ 3.25	82.85 $\pm$ 3.46	**83.60**$^-$ $\pm$ 3.39	83.22$^-$ $\pm$ 3.48
Image	96.59 $\pm$ 0.51	96.62 $\pm$ 0.51	96.74 $\pm$ 0.47	**97.01**$^-$ $\pm$ 0.43	95.35$^+$ $\pm$ 0.55
Ringnorm	93.29 $\pm$ 1.02	93.31 $\pm$ 0.94	93.39 $\pm$ 0.95	92.43$^+$ $\pm$ 0.85	**94.71**$^-$ $\pm$ 0.73
Splice	87.27 $\pm$ 0.79	86.00$^+$ $\pm$ 1.47	**87.67** $\pm$ 0.68	86.22$^+$ $\pm$ 0.71	87.62 $\pm$ 0.67
Thyroid	**95.09** $\pm$ 2.58	94.99 $\pm$ 2.59	95.04 $\pm$ 2.68	91.37$^+$ $\pm$ 3.41	89.99$^+$ $\pm$ 3.56
Titanic	77.60 $\pm$ 0.72	**77.63** $\pm$ 0.66	77.61 $\pm$ 0.68	77.52 $\pm$ 0.74	77.59 $\pm$ 0.74
Twonorm	97.30 $\pm$ 0.42	97.25 $\pm$ 0.43	97.42 $\pm$ 0.34	**97.47**$^-$ $\pm$ 0.24	97.14$^+$ $\pm$ 0.51
Waveform	89.14 $\pm$ 0.73	89.16 $\pm$ 0.86	89.13 $\pm$ 0.76	89.70$^-$ $\pm$ 0.71	**90.00**$^-$ $\pm$ 0.62
Average (B/S/W)	84.62 (3/0/2)	84.44 (2/2/2)	**84.69** (3/7/0)	84.21 (4/1/3)	83.79 (2/3/5)
W/T/L	—	2/11/0	0/13/0	5/4/4	5/5/3

For each problem the best average accuracy is shown in bold and the worst underlined. The "+" and "−" symbolsatthe accuracy show that the ML1 SVM is statistically better and worse than the classifier associated with the attached symbol, respectively.

According to the experiment for the two-class problems, generally the accuracies using the RBF kernels are better than those using polynomial kernels, but for both RBF and polynomial kernels, the ML1 SVM, $ML1_v$ SVM, and L1 SVM perform well, while the LS SVM and ULDM do not for the polynomial kernels.

5.2.3. Multiclass Problems

We used pairwise (one-vs-one) classification for multiclass problems. To resolve unclassifiable regions occurred by pairwise classification, we used fuzzy classification, introducing the membership function for each decision function [2].

Table 4 shows the ten multiclass problems used for performance evaluation. Unlike the two-class problems, for each problem there is one training dataset and one test dataset. The numeral problem [2] is to classify numerals in the Japanese license plates, and the thyroid problem [38] is a medical diagnosis problem. The blood cell problem [2] classifies white blood cells labeled according to the maturity of the growing stage. Three hiragana problems [2] are to classify hiragana characters in the Japanese license plates. The satimage problem [38] classifies lands according to the satellite images. The USPS [39] and MNIST [40,41] problems treat numeral classification and the letter problem [38] treats alphabets. Because the original training dataset of the MNIST problem is too large especially for the LS SVM and ULDM, we switched roles of the training data and the test data.

Except for the thyroid problem, the class data are relatively well balanced. For the thyroid data, almost all data belong to one class. Moreover, the classification accuracy of a classifier smaller than 92.41% is meaningless.

Table 4. Benchmark datasets for the multiclass problems.

Problem	Inputs	Classes	Training Data	Test Data	Prior (%)
Numeral	12	10	810	820	10.00 (10.00)
Thyroid	21	3	3772	3428	92.47 (92.71)
Blood cell	13	12	3097	3100	12.92 (12.90)
Hiragana-50	50	39	4610	4610	12.90 (5.64)
Hiragana-13	13	38	8375	8356	6.29 (6.29)
Hiragana-105	105	38	8375	8356	6.29 (6.29)
Satimage	36	6	4435	2000	24.17 (23.50)
USPS	256	10	7291	2007	16.38 (17.89)
MNIST	784	10	10,000	60,000	11.35 (11.23)
Letter	16	26	16,000	4000	4.05 (4.20)

Table 5 lists the accuracies using the RBF kernels for the test data. For each problem, the best accuracy is shown in bold, and the worst, underlined. For the MLP SVM, the accuracies for the thyroid, MNIST, and letter problems were not available. The "Average" row shows the average accuracy of the associated classifier for the ten problems and "B/S/W" shows the numbers of times that the best, the second best, and the worst accuracies are obtained. Among the ten problems, the accuracies of the ML1 SVM and $ML1_v$ SVM are better than or equal to those of the L1 SVM for nine and eight problems, respectively. In addition, the best average accuracy is obtained for the $ML1_v$ SVM, the second best, the ML1 SVM, and the third best, the L1 SVM. This is very different from the two-class problems where the ML1 SVM and $ML1_v$ SVM are comparable to the L1 SVM.

Table 6 shows the accuracies of the test data using polynomial kernels. For each problem the best accuracy is shown in bold and the worst, underlined. From the average accuracy, the $ML1_v$ SVM performs best, LS SVM performs the second best, and the L1 SVM and ULDM the worst. The difference between the LS SVM and ML1 SVM is very small. Improvement of the ML1 SVM and $ML1_v$ SVM over the L1 SVM is larger than that using the RBF kernels. For all the 10 problems, they are better than or equal to the L1 SVM. However, as seen from the Average rows in Tables 5 and 6, the accuracies using polynomial kernels are in general worse than those using RBF kernels. The reason for this is not clear but the ranges of parameter values in model selection might not be well tuned for polynomial kernels.

In the previous experiment we evaluated RBF kernels and polynomial kernels separately, but we can choose the best kernel from RBF and polynomial kernels. If we have cross-validation results for both kernels, we can select the better one that has the higher accuracy. Table 7 shows the accuracies by cross-validation for RBF and polynomial kernels. For each problem and for each classifier, the better

accuracy is shown in bold. The last row of the table shows the average accuracies. The average accuracies for the polynomial kernels are worse than those for the RBF kernels for all the classifiers. For each classifier, the number of problems that the polynomial kernels perform better or equal is one to two. Moreover, if we select RBF kernels when both average accuracies are the same, selecting kernels with the better average accuracy results in improving the accuracy for the test dataset as seen from Tables 5 and 6. However, employing polynomial kernels in addition to RBF kernels does not improve the accuracy significantly.

Table 5. Accuracies of the test data using RBF kernels for the multiclass problems.

Problem	ML1 SVM	$ML1_v$ SVM	L1 SVM	MLP SVM	LS SVM	ULDM
Numeral	**99.76**	**99.76**	**99.76**	99.27	99.15	99.39
Thyroid	**97.26**	97.23	**97.26**	—	95.39	95.27
Blood cell	93.45	93.55	93.16	93.36	94.23	**94.32**
Hiragana-50	99.11	99.22	99.00	98.96	**99.48**	98.96
Hiragana-13	99.89	**99.94**	99.79	99.90	99.87	99.89
Hiragana-105	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
Satimage	91.85	91.85	91.90	91.10	91.95	**92.25**
USPS	95.42	95.37	95.27	95.17	**95.47**	95.42
MNIST	96.96	96.95	96.55	—	96.99	**97.03**
Letter	**98.03**	**98.03**	97.85	—	97.88	97.75
Average (B/S/W)	97.17 (4/1/0)	**97.19** (4/1/0)	97.05 (3/0/3)	—(1/1/3)	97.04 (3/3/1)	97.03 (4/1/3)

For each problem, the best accuracy is shown in bold, and the worst underlined.

Table 6. Accuracies of the test data using polynomial kernels for the multiclass problems.

Problem	ML1 SVM	$ML1_v$ SVM	L1 SVM	LS SVM	ULDM
Numeral	**99.63**	**99.63**	**99.63**	99.02	99.27
Thyroid	**97.38**	**97.38**	**97.38**	94.66	94.66
Blood cell	94.32	93.77	92.13	94.39	**94.55**
Hiragana-50	98.92	99.05	98.81	**99.24**	98.76
Hiragana-13	99.75	99.77	99.64	**99.89**	99.44
Hiragana-105	**100.00**	**100.00**	99.99	**100.00**	**100.00**
Satimage	89.25	89.25	89.25	**90.30**	88.85
USPS	94.92	95.27	94.42	**95.42**	95.37
MNIST	96.14	96.39	95.85	**96.84**	96.54
Letter	96.90	97.10	96.10	**97.53**	96.10
Average (B/S/W)	96.72 (3/1/0)	**96.76** (3/4/0)	96.32 (2/1/5)	96.73 (7/1/2)	96.35 (2/2/5)

For each problem the best accuracy is shown in bold and the worst underlined.

Table 7. Cross-validation accuracies using RBF and polynomial kernels for the multiclass problems.

Problem	ML1 SVM		$ML1_v$ SVM		L1 SVM		LS SVM		ULDM	
	RBF	Poly	RBF	Poly	RBF	Poly	RBF	Poly	RBF	Poly
Numeral	**99.63**	99.51	**99.63**	**99.63**	**99.63**	99.51	**99.63**	99.51	**99.51**	**99.51**
Thyroid	97.61	**98.20**	97.61	**98.20**	97.59	**98.20**	**95.97**	95.71	**95.81**	94.75
Blood cell	**94.87**	94.77	**94.93**	94.74	**94.87**	94.64	94.83	**94.87**	**94.80**	94.70
Hiragana-50	**99.70**	99.63	**99.72**	99.65	**99.57**	99.46	**99.67**	99.63	**99.63**	99.59
Hiragana-13	**99.81**	99.76	**99.80**	99.77	**99.64**	99.51	99.86	**99.90**	**99.83**	99.63
Hiragana-105	**99.95**	99.86	**99.94**	99.86	**99.89**	99.79	**99.98**	99.92	**99.95**	99.89
Satimage	**92.60**	89.83	**92.56**	89.90	**92.47**	89.83	**92.72**	89.85	**92.36**	89.11
USPS	**98.37**	98.01	**98.37**	98.24	**98.27**	97.64	**98.44**	98.42	**98.46**	98.31
MNIST	**97.56**	96.86	**97.58**	97.14	**97.31**	96.68	**97.60**	97.48	**97.65**	97.14
Letter	**97.64**	96.97	**97.72**	97.10	**97.43**	95.52	**97.83**	97.37	**97.73**	96.18
Average	**97.77**	97.34	**97.79**	97.42	**97.67**	97.08	**97.65**	97.27	**97.57**	96.88

For each problem and for each classifier, the better accuracy is shown in bold.

5.3. Training Time Comparison

First, we examine the complexity of computation for each classifier excluding the MLP SVM. We trained the ML1 SVM, $ML1_v$ SVM, and L1 SVM by SMO combined with Newton's method. Therefore, the complexity of computation for the subproblem with working set size W is $O(W^3)$. Because the ML1 SVM and $ML1_v$ SVM solve two quadratic programming programs, each having the same number of variables, M, the complexity of computation is the same with that of the L1 SVM. Therefore, the three classifiers are considered to be trained in comparable time.

Because the matrices associated with the LS SVM and ULDM are positive definite, they can be solved by iterative methods such as stochastic gradient methods [17]. However, we trained the LS SVM and ULDM by Cholesky factorization to avoid the inaccuracy caused by insufficient convergence. Therefore, the complexity of computation of both methods is $O(M^3)$.

The purpose of this section is to confirm that the ML1 SVM, $ML1_v$ SVM, and L1 SVM can be trained in comparable time.

Excluding that of the MLP SVM, we compared the time for training and testing a classifier using a Windows machine with 3.2 GHz CPU and 16 GB memory. For the two-class problems, we set the parameter values with the frequently selected values and trained the classifier using a training dataset and tested the trained classifier using the associated test dataset. For the multiclass problems, we trained a classifier with the parameter values obtained by cross-validation and tested the trained classifier with the test dataset.

Because the tendency is similar we only show the results using RBF kernels. Table 8 shows the parameter values selected for the two-class and multiclass problems. In the table Thyroid (m) denotes the multiclass thyroid problem. For each problem, the γ values in bold, the γ and C values in bold, and γ, C, and C_h values in bold show that they appear more than once.

Table 8. Selected parameter values for the RBF kernels. For two-class problems, most frequently selected values (γ, C, C_h) are shown.

Data	ML1 SVM	$ML1_v$ SVM	L1 SVM	LS SVM	ULDM
Banana	10, 1, 1	**50**, 1, 1	20, 1	**50**, 10	**50**, 10^4
B. cancer	**0.5**, **1**, **1**	**0.5**, **1**, **1**	**0.5**, **1**	5, 1	10, 10
Diabetes	**0.1**, 50, 1	**0.1**, 1, 1	**0.1**, 500	0.5, 1	5, 100
Flare-solar	**0.01**, **50**, **1**	**0.01**, **50**, **1**	**0.01**, **50**	**0.01**, 10	**0.01**, 0.1
German	**0.1**, **1**, **1**	**0.1**, **1**, **1**	**0.1**, **1**	**0.1**, **1**	10, 100
Heart	**0.01**, **500**, **1**	**0.01**, **500**, **1**	**0.01**, 100	**0.01**, 10	**0.01**, 10^8
Image	**100**, 10, 1	**100**, **50**, 1	**100**, **50**	50, 50	15, 10^8
Ringnorm	100, 0.1, 1	**50**, **1**, 1	**50**, **1**	**50**, 0.1	**50**, 10
Splice	**10**, **10**, 10	5, 10, 1	**10**, **10**	**10**, **10**	**10**, 10^4
Thyroid	**5**, **50**, 1	**5**, 500, 1	**5**, **50**	100, 1	50, 10
Titanic	**0.01**, **50**, **1**	**0.01**, **50**, **1**	**0.01**, **50**	**0.01**, 10	**0.01**, 10^4
Twonorm	**0.01**, **50**, **1**	**0.01**, **50**, **1**	**0.01**, 1	**0.01**, **50**	**0.01**, 1000
Waveform	5, 1, 1	**50**, 1, 1	15, 1	20, 1	**50**, 100
Numeral	**5**, **10**, **1**	**5**, **10**, **1**	**5**, **10**	1, 100	15, 10^4
Thyroid (m)	**10**, **2000**, **1**	**10**, **2000**, 10	**10**, **2000**	50, 2000	200, 10^8
Blood cell	**5**, **100**, 100	**5**, **100**, 50	**5**, **100**	**5**, 500	**5**, 10^8
Hiragana-50	**5**, **100**, **50**	**5**, **100**, **50**	**5**, **100**	**10**, 100	**10**, 10^6
Hiragana-13	**50**, **50**, 500	**50**, **50**, 50	**15**, 1000	**15**, 2000	20, 10^8
Hiragana-105	**20**, **10**, 500	**20**, **10**, 50	**10**, 10	15, 2000	**10**, 10^6
Satimage	**200**, **10**, **10**	**200**, **10**, **10**	**200**, **10**	**200**, **10**	**200**, 10^4
USPS	**10**, **50**, **2000**	**10**, **50**, **2000**	**10**, 100	**5**, 500	**5**, 10^8
MNIST	**20**, **10**, **500**	**20**, **10**, **500**	**20**, **10**	**10**, 50	**10**, 10^6
Letter	**100**, **10**, 10	**100**, **10**, 100	200, 10	**50**, 50	**50**, 10^6

For each problem, the γ values in bold, the γ and C values in bold, and γ, C, and C_h values in bold show that they appear more than once.

From the table, it is clear that the ML1 SVM, $ML1_v$ SVM, and L1 SVM selected the same γ and C values frequently and the ML1 SVM and $ML1_v$ SVM selected the same γ, C, and C_h values 10 times out of 23 problems.

Table 9 sows the CPU time for training and test with the optimized parameter values listed in Table 8. For each problem the shortest CPU time is shown in bold and the longest, underlined. The CPU time for the two-class problem is that per file. For the multiclass problems, we used fuzzy pairwise classification. Therefore, the training time includes that for determining $n(n-1)/2$ decision functions where n is the number of classes. For example, for the letter problem, 329 decision functions need to be determined. The last row of the table shows the numbers of times that the associated classifier are the fastest (B), the second fastest (S), and the slowest (W).

From the table, the ML1 SVM and $ML1_v$ SVM show comparable computation time. Moreover, except for the hiragana-50, hiragana-13, hiragana-105, USPS, MNIST, and letter problems, computation time for the ML1 SVM and $ML1_v$ SVM is comparable to that of the L1 SVM. For these problems, the ML1 SVM and $ML1_v$ SVM are much slower than L1 SVM. Analyzing the convergence process, we found that for these problems, monotonicity of the objective function value was sometimes violated. To improve convergence, we need to clarify why it happens and to find a way to speed up training in such a situation. However we leave this problem in the future study.

Table 9. Computation time using the RBF kernels (in seconds).

Data	ML1 SVM	$ML1_v$ SVM	L1 SVM	LS SVM	ULDM
Banana	0.096	**0.053**	0.067	0.192	0.254
B. cancer	0.006	**0.005**	**0.005**	0.010	0.018
Diabetes	**0.025**	0.026	0.029	0.119	0.222
Flare-solar	0.057	0.059	**0.055**	0.341	0.693
German	0.059	**0.055**	0.059	0.418	0.783
Heart	**0.004**	**0.004**	0.005	0.010	0.013
Image	**0.306**	0.354	0.327	8.24	21.7
Ringnorm	0.226	0.141	**0.130**	0.362	0.420
Splice	5.29	**1.79**	8.52	3.77	8.52
Thyroid	0.003	**0.002**	**0.002**	0.005	0.008
Titanic	0.017	**0.016**	0.017	0.028	0.031
Twonorm	**0.244**	0.250	0.336	0.422	0.484
Waveform	0.122	0.156	**0.106**	0.268	0.334
Numeral	0.125	0.125	**0.047**	1.17	1.14
Thyroid (m)	1.53	1.52	**0.938**	621	1452
Blood cell	2.91	3.08	**0.734**	33.1	49.7
Hiragana-50	47.3	97.7	**8.67**	244	268
Hiragana-13	348	295	**9.67**	740	920
Hiragana-105	950	920	**48.5**	1779	1997
Satimage	27.4	28.9	**19.7**	292	693
USPS	513	634	**35.5**	1089	1996
MNIST	6143	6670	**1435**	8323	11,372
Letter	1390	2123	**439**	3036	6544
B/S/W	4/11/0	7/7/0	15/4/1	0/1/1	0/0/22

For each problem the shortest CPU time is shown in bold and the longest underlined.

5.4. Discussions

As discussed in Section 3.3, the ML1 SVM and $ML1_v$ SVM are equivalent for a small C_h value but for a large C_h value they are different. The computer experiments in Section 5.1 also revealed that the ML1 SVM is insensitive to the change of a C_h value. However, the difference of the generalization performance between the ML1 SVM and $ML1_v$ SVM is not large for the two-class and multiclass problems.

The execution time for the ML1 and $ML1_v$ SVM was sometimes longer than that for the L1 SVM. This will cause problem in model selection. While we leave the discussions of speeding up training, for the model selection, this problem may be alleviated for the $ML1_v$ SVM. If $h^+ < 1$ or $h^- < 1$ is satisfied, the solution is infeasible. Therefore, we can skip cross-validation at the current C_h value and the larger ones.

We used line search to speed up cross-validation. If grid search was used, in fivefold cross-validation we needed to train the ML1 SVM or $ML1_v$ SVM 3520 $(11 \times 8 \times 8 \times 5)$ times, instead of 480 $((11 \times 8 + 8) \times 5)$ times. The speed up ratio is estimated to be 7.3. We evaluated the difference between the grid search and line search for the heart problem. We measured the execution time of cross-validation, training the classifier with the determined parameter values, and testing the classifier using the test data. For the ML1 SVM, the speed up ratio by line search was 35.7, and the average accuracy with the standard deviation of the grid search was 82.78 $\pm$ 3.45%, which was slightly lower. By the $ML1_v$ SVM, the speed up ratio was 40.0, and the average accuracy with the standard deviation was 82.85 $\pm$ 3.31%, which was also lower than that by line search. Therefore, because model selection slowed down very much and the improvement of the average accuracy was not obtained at least for the heart problem, grid search will not be a good selection.

To speed up model selection of the ML1 SVM or $ML1_v$ SVM, we may use the L1 SVM considering that the same values were selected frequently for the $\gamma(d)$ and C values (see Table 8 for γ values). For the multiclass problems, four problems do not have the same values. To check whether the idea works, we performed model selection of C_h values fixing the values of γ and C determined by model selection of the L1 SVM for the hiragana-13, hiragana-105, USPS, and letter problems. Among the four problems, the ML1 SVM and $ML1_v$ SVM performed worse than the L1 SVM for the letter problem. Moreover, the resulting average accuracies of the ML1 SVM and $ML1_v$ SVM for ten problems were 97.14% and 97.16%, respectively, which were lower than by the original model selection by 0.03% (see Table 5) but were still better than the accuracy of the L1 SVM. If we switch back the roles of the training and test data for the MNIST problem, the selected parameter values for the L1 SVM were the same. The accuracies for the test data were 98.55%, 98.77%, and 98.78% for the L1 SVM, ML1 SVM, and $ML1_v$ SVM, respectively.

For the polynomial kernels, different kernel parameters were selected for six problems: the numeral (only for the $ML1_v$ SVM), blood cell, hiragana-50, hiragana-13, hiragana-105, and USPS problems. For each problem we determined the C_h value by cross-validation fixing the values of d and C determined by the L1 SVM. The accuracies for the test datasets were all better than or equal to those by the L1 SVM. The average accuracies of the ML1 SVM or $ML1_v$ SVM for all the ten problems were 96.53% and 96.64%, respectively, which were lower than by the original model selection by 0.19% and 0.12%, respectively. For the MNIST problem with the switched training and test data, the selected parameter values for the L1 SVM were the same. Moreover, the accuracies for the test data were 98.17%, 98.23%, and 98.34% for the L1 SVM, ML1 SVM, and $ML1_v$ SVM, respectively.

6. Conclusions

The minimal complexity machine (MCM) minimizes the VC dimension and generalizes better than the standard support vector machine (L1 SVM). However, according to our previous analysis, the solution of the MCM is non-unique and unbounded.

In this paper, to solve the problem of the MCM and to improve the generalization ability of the L1 SVM, we fused the MCM and the L1 SVM, namely, we introduced minimizing the upper bound of the absolute decision function values to the L1 SVM. This corresponds to minimizing the maximum margin. Converting the original classifier into dual one, we derived two subproblems: the first subproblem corresponds to the L1 SVM and the second subproblem corresponds to minimizing the upper bound. We further modified the second subproblem by converting the inequality constraint into two equality constraints: one for optimizing the variables associated with the positive class and

the other for the negative class. We call this architecture $ML1_v$ SVM and the original architecture, ML1 SVM.

We derived the exact KKT conditions for the first and second subproblems that exclude the bias term and the upper bound and discussed training the two subproblems alternatingly, fusing sequential minimal optimization (SMO) and Newton's method.

According to computer experiments of the two-class problems using RBF kernels, the average accuracy of the ML1 SVM is statistically comparable to that of the $ML1_v$ SVM and L1 SVM. Using polynomial kernels, the ML1 SVM is statistically comparable to the L1 SVM but is slightly better than $ML1_v$ SVM.

For the multiclass problems using RBF kernels, the $ML1_v$ SVM and ML1 SVM generalize better than the L1 SVM and the $ML1_v$ SVM performs best, and the ML1 SVM, the second, among six classifiers tested. Using polynomial kernels, the $ML1_v$ SVM performs best, the LS SVM the second best, ML1 SVM the third, and the L1 SVM worst.

Therefore, the idea of minimizing the VC dimension for the L1 SVM worked to improve the generalization ability of the L1 SVM.

Execution time for the ML1 SVM and $ML1_v$ SVM is comparable to that for the L1 SVM for most of the problems tested, but in some problems, execution time is much longer. In the future study, we would like to clarify the reason and propose a fast training method in such cases. Another study will be to consider robustness for outliers by the soft upper bound, instead of the hard upper bound.

Funding: This work was supported by JSPS KAKENHI Grant Number 19K04441.

Conflicts of Interest: The author declares no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. Vapnik, V.N. *Statistical Learning Theory*; John Wiley & Sons: New York, NY, USA, 1998.
2. Abe, S. *Support Vector Machines for Pattern Classification*, 2nd ed.; Springer: London, UK, 2010.
3. Abe, S. Training of Support Vector Machines with Mahalanobis Kernels. In *Artificial Neural Networks: Formal Models and Their Applications (ICANN 2005)—Proceedings of Fifteenth International Conference, Part II, Warsaw, Poland*; Duch, W., Kacprzyk, J., Oja, E., Zadrożny, S., Eds.; Springer-Verlag: Berlin, Germany, 2005; pp. 571–576.
4. Wang, D.; Yeung, D.S.; Tsang, E.C.C. Weighted Mahalanobis Distance Kernels for Support Vector Machines. *IEEE Trans. Neural Netw.* **2007**, *18*, 1453–1462. [CrossRef]
5. Shen, C.; Kim, J.; Wang, L. Scalable Large-Margin Mahalanobis Distance Metric Learning. *IEEE Trans. Neural Netw.* **2010**, *21*, 1524–1530. [CrossRef]
6. Liang, X.; Ni, Z. Hyperellipsoidal Statistical Classifications in a Reproducing Kernel Hilbert Space. *IEEE Trans. Neural Netw.* **2011**, *22*, 968–975. [CrossRef]
7. Fauvel, M.; Chanussot, J.; Benediktsson, J.; Villa, A. Parsimonious Mahalanobis kernel for the classification of high dimensional data. *Pattern Recognit.* **2013**, *46*, 845–854. [CrossRef]
8. Reitmaier, T.; Sick, B. The responsibility weighted Mahalanobis kernel for semi-supervised training of support vector machines for classification. *Inf. Sci.* **2015**, *323*, 179–198. [CrossRef]
9. Jiang, H.; Ching, W.K.; Yiu, K.F.C.; Qiu, Y. Stationary Mahalanobis kernel SVM for credit risk evaluation. *Appl. Soft Comput.* **2018**, *71*, 407–417. [CrossRef]
10. Sun, G.; Rong, X.; Zhang, A.; Huang, H.; Rong, J.; Zhang, X. Multi-Scale Mahalanobis Kernel-Based Support Vector Machine for Classification of High-Resolution Remote Sensing Images. *Cogn. Comput.* **2019**. [CrossRef]
11. Lanckriet, G.R.G.; Cristianini, N.; Bartlett, P.; Ghaoui, L.E.; Jordan, M.I. Learning the Kernel Matrix with Semidefinite Programming. *J. Mach. Learn. Res.* **2004**, *5*, 27–72.
12. Shivaswamy, P.K.; Jebara, T. Ellipsoidal Kernel Machines. In Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS 2007), San Juan, Puerto Rico, 21–24 March 2007.

13. Xue, H.; Chen, S.; Yang, Q. Structural Regularized Support Vector Machine: A Framework for Structural Large Margin Classifier. *IEEE Trans. Neural Netw.* **2011**, *22*, 573–587. [CrossRef]
14. Peng, X.; Xu, D. Twin Mahalanobis distance-based support vector machines for pattern recognition. *Inf. Sci.* **2012**, *200*, 22–37. [CrossRef]
15. Ebrahimpour, Z.; Wan, W.; Khoojine, A.S.; Hou, L. Twin Hyper-Ellipsoidal Support Vector Machine for Binary Classification. *IEEE Access* **2020**, *8*, 87341–87353. [CrossRef]
16. Pelckmans, K.; Suykens, J.; Moor, B.D. A Risk Minimization Principle for a Class of Parzen Estimators. In *Advances in Neural Information Processing Systems 20*; Platt, J., Koller, D., Singer, Y., Roweis, S., Eds.; Curran Associates, Inc.: New York, NY, USA, 2008; pp. 1137–1144.
17. Zhang, T.; Zhou, Z.H. Large Margin Distribution Machine. In Proceedings of the Twentieth ACM SIGKDD Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 313–322.
18. Zhu, Y.; Wu, X.; Xu, J.; Zhang, D.; Zuo, W. Radius-margin based support vector machine with LogDet regularizaron. In Proceedings of the 2015 International Conference on Machine Learning and Cybernetics (ICMLC), Guangzhou, China, 12–15 July 2015; Volume 1, pp. 277–282.
19. Abe, S. Improving Generalization Abilities of Maximal Average Margin Classifiers. In *Artificial Neural Networks in Pattern Recognition, Proceedings of the 7th IAPR TC3 Workshop (ANNPR 2016), Ulm, Germany, 28–30 September 2016*; Schwenker, F., Abbas, H.M., Gayar, N.E., Trentin, E., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 29–41.
20. Abe, S. Unconstrained Large Margin Distribution Machines. *Pattern Recognit. Lett.* **2017**, *98*, 96–102. [CrossRef]
21. Abe, S. Effect of Equality Constraints to Unconstrained Large Margin Distribution Machines. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*; Lecture Notes in Computer Science; Pancioni, L., Schwenker, F., Trentin, E., Eds.; Springer: Cham, Switzerland, 2018; Volume 11081, pp. 41–53.
22. Zhang, T.; Zhou, Z. Optimal Margin Distribution Machine. *IEEE Trans. Knowl. Data Eng.* **2020**, *32*, 1143–1156. [CrossRef]
23. Burges, C.J.C. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.* **1998**, *2*, 121–167. [CrossRef]
24. Duarte, E.; Wainer, J. Empirical comparison of cross-validation and internal metrics for tuning SVM hyperparameters. *Pattern Recognit. Lett.* **2017**, *88*, 6–11. [CrossRef]
25. Du, J.Z.; Lu, W.G.; Wu, X.H.; Dong, J.Y.; Zuo, W.M. L-SVM: A radius-margin-based SVM algorithm with LogDet regularization. *Expert Syst. Appl.* **2018**, *102*, 113–125. [CrossRef]
26. Wu, X.; Zuo, W.; Lin, L.; Jia, W.; Zhang, D. F-SVM: Combination of Feature Transformation and SVM Learning via Convex Relaxation. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 5185–5199. [CrossRef]
27. Jayadeva. Learning a hyperplane classifier by minimizing an exact bound on the VC dimension. *Neurocomputing* **2015**, *149*, 683–689. [CrossRef]
28. Jayadeva; Soman, S.; Pant, H.; Sharma, M. QMCM: Minimizing Vapnik's bound on the VC dimension. *Neurocomputing* **2020**, *399*, 352–360. [CrossRef]
29. Abe, S. Analyzing Minimal Complexity Machines. In Proceedings of the International Joint Conference on Neural Networks, Budapest, Hungary, 14–19 July 2019.
30. Abe, S. Minimal Complexity Support Vector Machines. In *Artificial Neural Networks in Pattern Recognition*; Lecture Notes in Computer Science; Schilling, F.P., Stadelmann, T., Eds.; Springer: Cham, Switzerland, 2020; Volume 12294, pp. 89–101.
31. Abe, S. Fusing Sequential Minimal Optimization and Newton's Method for Support Vector Training. *Int. J. Mach. Learn. Cybern.* **2016**, *7*, 345–364. [CrossRef]
32. Abe, S. Sparse Least Squares Support Vector Training in the Reduced Empirical Feature Space. *Pattern Anal. Appl.* **2007**, *10*, 203–214. [CrossRef]
33. Keerthi, S.S.; Gilbert, E.G. Convergence of a generalized SMO algorithm for SVM classifier design. *Mach. Learn.* **2002**, *46*, 351–360. [CrossRef]
34. Fan, R.E.; Chen, P.H.; Lin, C.J. Working Set Selection Using Second Order Information for Training Support Vector Machines. *J. Mach. Learn. Res.* **2005**, *6*, 1889–1918.
35. Barbero, A.; Dorronsoro, J.R. Faster Directions for Second Order SMO. In *Artificial Neural Networks—ICANN 2010*; Lecture Notes in Computer Science; Diamantaras, K., Duch, W., Iliadis, L.S., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6353, pp. 30–39.

36. Bezdek, J.C.; Keller, J.M.; Krishnapuram, R.; Kuncheva, L.I.; Pal, N.R. Will the real iris data please stand up? *IEEE Trans. Fuzzy Syst.* **1999**, *7*, 368–369. [CrossRef]
37. Rätsch, G.; Onoda, T.; Müller, K.R. Soft Margins for AdaBoost. *Mach. Learn.* **2001**, *42*, 287–320. [CrossRef]
38. Asuncion, A.; Newman, D.J. UCI Machine Learning Repository. 2007. Available online: https://archive.ics.uci.edu/ml/index.php (accessed on 23 October 2020).
39. USPS Dataset. Available online: https://www.kaggle.com/bistaumanga/usps-dataset (accessed on 23 October 2020).
40. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
41. LeCun, Y.; Cortes, C. The MNIST Database of Handwritten Digits. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 23 October 2020).