The paper is an extension of the ANNPR 2016 paper, and was rejected by a journal after three rounds of the review. The one reviewer among three still has something to say.

Dear Prof. Abe,

We have received the reports from our advisors on your manuscript "Extended Maximal Average Margin Classifiers".

With regret, I must inform you that, based on the advice received, the Editors have decided that your manuscript cannot be accepted for publication.
Below, please find the comments for your perusal.

I would like to thank you very much for forwarding your manuscript to us for consideration and wish you every success in finding an alternative place of publication.

With kind regards,

Comments for the Author:

Editor:
The associate editors of the special issue have decided to reject the paper, as it would still need major revision and a new submission-review process is not compatible with the timeline of a special issue. Shoud the authors be willing to take the reviewers' comments into account, they may resubmit a paper to a regulatr issue; however it will undergo a new review process.

Reviewer #1: Now I know the essence of the proposed model, which is equally to the least squares support vector machine. The authors also realized that the dual problems of them are the same, with different parameters. For the primal problem they are the same as well.

Consider (30)-(31), the proposed primal problem. Since ¥epsilon_i = 1 - y_i (w^T ¥phi(x_i) + b), then the problem becomes

Q = -1/2 w^T w + C_a/M ¥sum_i (1 - ¥epsilon_i) - C/2 ¥sum_i ¥epsilon_i^2 = -1/2 w^T w - C/2 ¥sum (¥epsilon_i^2 - p ¥epsilon_i + q), where p = 2C_a/CM and q = 2MC/C_a.

Since ¥epsilon_i^2 - p ¥epsilon_i + q could be written as (¥epsilon_i - p/2)^2 + q - p^2/2 and we can transfer the primal problem as

Q = -1/2 w^T w + C/2 ¥delta_i^2, where ¥delta_i = ¥epsilon_i - p/2 = (1-p/2) - y_i (w^T ¥phi(x_i) + b) = (1-p/2)(1 - y_i ((w/(1-p/2))^T ¥phi(x_i) + b).

Finally, we know the proposed model (30)-(31) become

Q = -1/2*(1-p/2) w^T w + C/2(1 - ¥delta_i), with ¥delta_i = 1 - y_i (w^T ¥phi(x_i) + b).

Generally speaking, adding a linear term in the LS-SVM primal problem (this is the "proposed method") is equal to change the coefficient of LS-SVM but the model itself is not changed at all. This also explains why the author observed that the dual problem keep unchanged from the LS-SVM.

Reviewer #2: Thanks for the response. I have no more questions.

Noname manuscript No.
(will be inserted by the editor)

Extended Maximal Average Margin Classifiers

Shigeo Abe

Received: date / Accepted: date

Abstract Maximal average margin classifiers (MAMCs) maximize the average margin without constraints for training data. Although training is fast, the generalization abilities are usually inferior to those of support vector machines (SVMs). In this paper, we propose three ways to improve the generalization abilities of MAMCs: 1) to optimize the bias term of the separating hyperplane, 2) to optimize the slope of the separating hyperplane, and 3) to introduce the equality constraints into MAMCs. After the coefficient vector of the hyperplane is obtained, the bias term is optimized so that the number of misclassifications is minimized. To optimize the slope, we introduce a weight to the average of mapped training data for one class and optimize the weight by cross-validation. To improve the generalization ability further, we propose equality-constrained MAMCs and show that they reduce to least squares SVMs. Using two-class and multiclass problems, we show that the generalization ability of the unconstrained MAMCs are inferior to those of the constrained MAMCs and SVMs.

Keywords Average margins · Bias terms · Cross-validation · Slope of hyperplanes · Support vector machines

1 Introduction

The support vector machine (SVM) [1, 2] maximizes the minimum margin between two classes, where the margin is the distance between the data point and the hyperplane. The margin is non-negative if the associated data point is correctly classified, and negative, if misclassified. Because the SVM does not assume a specific data distribution, a priori knowledge on the data distribution can improve the generalization ability. The Mahalanobis distance, instead of the Euclidean distance is useful for this purpose. One approach reformulates SVMs so that the margin

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

is measured by the Mahalanobis distance [3–7], and another approach uses Mahalanobis kernels, which calculate the kernel value according to the Mahalanobis distance [8–13].

The generalization ability of various conventional classifiers has also been improved by introducing the idea of maximizing the minimum margin [2]. But in AdaBoost [14], the margin distribution, instead of the minimum margin, has been known to be important in improving the generalization ability [15, 16].

Several approaches have been proposed to control the margin distribution in SVM-like classifiers [17–22]. In [18], a maximum average margin classifier (MAMC) is proposed, in which instead of maximizing the minimum margin, the margin mean for the training data is maximized without slack variables. In [21, 22], in addition to maximizing the margin mean, the margin variance is minimized and the classifier is called large margin distribution machine (LDM). According to the computer experiments in [21], the generalization ability of MAMCs is inferior to that of SVMs and LDMs. Because the architecture of MAMCs is much simpler and the training can be done by addition or subtraction of kernels, improving the generalization ability of MAMCs without incurring much computational burden is highly expected.

In this paper[1], we clarify why MAMCs perform poorly for some classification problems and propose three methods to improve the generalization ability. Because the MAMC does not include constraints associated with training data, the determined bias term depends only on the difference between the numbers of training data for the two classes. To solve this problem, after the weight vector is obtained by the MAMC, we optimize the bias term so that the classification error is minimized. Then to improve the generalization ability further, we introduce a weight parameter to the average vector of one class and determine the parameter value by cross-validation. This results in optimizing the slope of the separating hyperplane. To improve the generalization ability further, we define the equality-constrained MAMC (EMAMC), which is shown to be equivalent to the least squares (LS) SVM. Using two-class and multiclass problems, we show that the generalization ability of the unconstrained MAMCs with the optimized bias term and slopes are inferior to that of the EMAMC.

In Section 2, we explain the architecture of the MAMC and clarify the problems of MAMC. Then, in section 3, we propose bias term optimization and slope optimization and develop the EMAMC. In Section 4, using two-class and multiclass problems, we compare the generalization abilities of the MAMC with those of the proposed MAMC with optimized bias terms and slopes, the EMAMC, and the SVM.

2 Maximum Average Margin Classifiers

In this section, we explain MAMCs according to [18], and discuss their problems.

[1] This is an extended version of [23].

2.1 Architecture

We consider a classification problem with M training input-output pairs $\{\mathbf{x}_i, y_i\}$ $(i = 1, \ldots, M)$, where $\mathbf{x}_i$ are 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. We map the m-dimensional input vector $\mathbf{x}$ into the l-dimensional feature space using the nonlinear vector function $\boldsymbol{\phi}(\mathbf{x})$. In the feature space, we determine the decision function that separates Class 1 data from Class 2 data:

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

where $\mathbf{w}$ is the l-dimensional vector, $\top$ denotes the transpose of a vector (matrix), and b is the bias term.

The margin of $\mathbf{x}_i$, δ_i, which is the distance from the hyperplane, is given by

$$\delta_i = y_i \, (\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) + b)/\|\mathbf{w}\|. \tag{2}$$

Under the assumption of

$$\mathbf{w}^\top \mathbf{w} = 1, \tag{3}$$

(2) becomes

$$\delta_i = y_i \, (\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) + b). \tag{4}$$

With $b = 0$, the MAMC, which maximizes the average margin, is defined by

$$\text{maximize} \quad Q(\mathbf{w}) = \frac{1}{M} \sum_{i=1}^{M} y_i \, \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) \tag{5}$$

$$\text{subject to} \quad \mathbf{w}^\top \mathbf{w} = 1. \tag{6}$$

Introducing the Lagrange multiplier $\lambda \, (> 0)$, we obtain the unconstrained optimization problem:

$$\text{maximize} \quad Q(\mathbf{w}) = \frac{1}{M} \sum_{i=1}^{M} y_i \, \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) - \frac{\lambda}{2} (\mathbf{w}^\top \mathbf{w} - 1). \tag{7}$$

Taking the derivative of Q with respect to $\mathbf{w}$, we obtain the optimal $\mathbf{w}$:

$$\lambda \, \mathbf{w} = \frac{1}{M} \sum_{i=1}^{M} y_i \, \boldsymbol{\phi}(\mathbf{x}_i). \tag{8}$$

In [18], λ is determined using (6) and (8), but λ can take on any positive value because that does not change the decision boundary. Therefore, in the following we set $\lambda = 1$.

In calculating the decision function given by (1), we use kernel $K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}^\top(\mathbf{x}) \, \boldsymbol{\phi}(\mathbf{x})$ to avoid treating the variables in the feature space explicitly.

The resulting decision function $f(\mathbf{x})$ with $b = 0$ is given by

$$f(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^{M} y_i \, K(\mathbf{x}, \mathbf{x}_i). \tag{9}$$

Among several kernels, radial basis function (RBF) kernels are widely used and thus in the following study we use RBF kernels:

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma||\mathbf{x} - \mathbf{x}'||^2/m), \tag{10}$$

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

2.2 Problems with MAMCs

The MAMC is derived without a bias term, i.e., $b = 0$. To include the bias term we change (7) to

$$\text{maximize} \quad Q(\mathbf{w}, b) = \frac{1}{M}\sum_{i=1}^{M} y_i \left(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) + b\right) - \frac{1}{2}(\mathbf{w}^\top \mathbf{w} + b^2), \tag{11}$$

where Q is maximized with respect to $\mathbf{w}$ and b. In (11), we replace λ with 1 and delete the constant term. The inclusion of $b^2/2$ in the objective function is used in [24] to eliminate the equality constraint in the SVM.

From (11), Q is maximized when

$$\mathbf{w} = \frac{1}{M}\sum_{i=1}^{M} y_i \, \boldsymbol{\phi}(\mathbf{x}_i), \tag{12}$$

$$b = \frac{1}{M}\sum_{i=1}^{M} y_i. \tag{13}$$

From (13), b is determined by the deference of the numbers of the data belonging to Classes 1 and 2, not by the distributions of the data belonging to the two classes. And if the numbers are the same, $b = 0$, irrespective of $\mathbf{x}_i$ $(i = 1, \ldots, M)$. This occurs because the coefficient of b becomes zero in (11); the value of b does not affect optimality of the solution.

This means that the constraints are lacking for determining the optimal value of b. Similar to SVMs, the addition of inequality or equality constraints for the training data may solve the problem, which will be discussed later.

3 Extensions of MAMCs

In this section, we discuss three ways to improve the generalization ability of MAMCs: 1) optimizing the bias term of the separating hyperplane, 2) optimizing the slope of the separating hyperplane, and 3) introducing the equality constraints into MAMCs.

Methods 1) and 2) employ two-stage training: first $\mathbf{w}$ is determined by (12). Then, the bias term b and/or slope of the hyperplane are optimized. In method 3), the equality constraints are introduced and $\mathbf{w}$ and b are determined simultaneously. Methods 1) and 2) will work well if $\mathbf{w}$ determined by (12) is a good estimate. This will be evaluated in Section 4.

3.1 Bias Term Optimization

We propose two-stage training; in the first stage, we determine the coefficient vector $\mathbf{w}$ using (12), and in the second stage, we optimize the value of b so that the number of misclassifications is minimized:

$$\text{minimize} \quad E_{\mathrm{R}}(b, \rho, \boldsymbol{\xi}) = \sum_{i=1}^{M} I(\xi_i) \tag{14}$$

$$\text{subject to} \quad y_i \left(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) + b\right) \geq \rho - \xi_i,$$
$$\rho > 0, \quad \xi_i \geq 0 \quad \text{for} \quad i = 1, \ldots M, \tag{15}$$

where $E_{\mathrm{R}}(b, \rho, \boldsymbol{\xi})$ is the number of misclassifications, ρ is a positive parameter, $\xi_i\ (\geq 0)$ are slack variables, and $I(\xi_i) = 0$ for $\xi_i = 0$ and $I(\xi_i) = 1$ for $\xi_i > 0$. Here, $\mathbf{w}$ is determined in the first stage and thus is constant.

If there are multiple b values that minimize (14), we break the tie by

$$\text{minimize} \quad E_{\mathrm{S}}(\boldsymbol{\xi}) = \sum_{i=1}^{M} \xi_i, \tag{16}$$

where $E_{\mathrm{S}}(\boldsymbol{\xi})$ is the sum of slack variables.

First we consider the case where the classification problem is separable in the feature space. Suppose that

$$\max_{\substack{j=1,\ldots,M \\ y_j=-1}} \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_j) < \min_{\substack{i=1,\ldots,M \\ y_i=1}} \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) < 0 \tag{17}$$

is satisfied, where training data belonging to Class 2 are correctly classified but some of the training data belonging to Class 1 are misclassified. Because of the first inequality in (17), by setting a proper value to b, all the training data are correctly classified.

Let

$$j = \arg_j \max_{\substack{j=1,\ldots,M \\ y_j=-1}} \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_j), \quad i = \arg_i \min_{\substack{i=1,\ldots,M \\ y_i=1}} \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i). \tag{18}$$

Then, from (15), to make $\mathbf{x}_i$ and $\mathbf{x}_j$ be correctly classified with margin ρ,

$$\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) + b = \rho, \quad -(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_j) + b) = \rho \tag{19}$$

must be satisfied. From (19),

$$b = -\frac{1}{2}(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) + \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_j)), \quad \rho = \frac{1}{2}(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_j)). \tag{20}$$

It is clear that (20) satisfies $E_{\mathrm{R}} = E_{\mathrm{S}} = 0$. From (19), the margins for $\mathbf{x}_i$ and $\mathbf{x}_j$ are the same and thus they are the maximum.

Equations (20) are also valid when

$$0 < \max_{\substack{j=1,\ldots,M \\ y_j=-1}} \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_j) < \min_{\substack{i=1,\ldots,M \\ y_i=1}} \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i), \tag{21}$$

where some of the training data for Class 2 are misclassified.

Now consider the inseparable case. Let the misclassified training data for Class 1 be $\mathbf{x}_{i_k}$ $(k = 1, \ldots, p)$ and

$$\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{i_1}) \le \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{i_2}) \le \cdots \le \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{i_p}) \le 0. \tag{22}$$

Likewise, let the misclassified training data for Class 2 be $\mathbf{x}_{j_k}$ $(k = 1, \ldots, n)$ and

$$0 \le \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{j_1}) \le \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{j_2}) \le \cdots \le \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{j_n}). \tag{23}$$

Similar to the separable case, the optimal b occurs at (20), where $\mathbf{x}_i$ is misclassified and $\mathbf{x}_j$ is correctly classified, or vice versa. When $\mathbf{x}_i$ is misclassified, $i = i_k$ $(k \in \{1, \ldots, p\})$ and j is given by

$$j = \arg_j \max_{\substack{\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_j) < \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{i_k}) \\ y_j = -1, j = 1, \ldots, M}} \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_j). \tag{24}$$

When j is misclassified, $j = j_k$ $(k \in \{1, \ldots, n\})$ and i is given by

$$i = \arg_i \min_{\substack{\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_{j_k}) < \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) \\ y_i = 1, i = 1, \ldots, M}} \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i). \tag{25}$$

Let $E_{\mathrm{R}}(i, j)$ and $E_{\mathrm{S}}(i, j)$ denote that E_{R} and E_{S} are evaluated with b determined using $\mathbf{x}_i$ and $\mathbf{x}_j$ by (20), where $i = i_k$ $(k \in \{1, \ldots, p\})$ and j is given by (24) or $j = j_k$ $(k \in \{1, \ldots, n\})$ and i is given by (25). For each pair of i and j, we calculate $E_{\mathrm{R}}(i, j)$ and select the value of b that minimizes $E_{\mathrm{R}}(i, j)$. If there are multiple pairs of i and j, we select the value of b that minimizes $E_{\mathrm{S}}(i, j)$.

We call the MAMC with the optimized bias term, $\mathrm{MAMC}_{\mathrm{b}}$.

3.2 Slope Optimization

To optimize the slope of the separating hyperplane, first we analyze the characteristics of the solution.

Rewriting (12),

$$\mathbf{w} = \frac{M_+}{M} \bar{\boldsymbol{\phi}}_+ - \frac{M_-}{M} \bar{\boldsymbol{\phi}}_- \tag{26}$$

where

$$\bar{\boldsymbol{\phi}}_+ = \frac{1}{M_+} \sum_{\substack{i=1 \\ y_i = 1}}^{M} \boldsymbol{\phi}(\mathbf{x}_i), \quad \bar{\boldsymbol{\phi}}_- = \frac{1}{M_-} \sum_{\substack{i=1 \\ y_i = -1}}^{M} \boldsymbol{\phi}(\mathbf{x}_i), \tag{27}$$

and $\bar{\boldsymbol{\phi}}_+$ and $\bar{\boldsymbol{\phi}}_-$ are the averages of the mapped training data belonging to Classes 1 and 2, respectively, and M_+ and M_- are the numbers of training data belonging to Classes 1 and 2, respectively.

If $M_+ = M_-$, $\mathbf{w}$ is the vector which is from $\bar{\boldsymbol{\phi}}_-/2$ to $\bar{\boldsymbol{\phi}}_+/2$. Therefore the decision function is orthogonal to the vector. If $M_+ \neq M_-$, the decision function is orthogonal to $\bar{\boldsymbol{\phi}}_+ - (M_-/M_+)\, \bar{\boldsymbol{\phi}}_-$

To control the decision function, we introduce a positive hyperparameter C_{m} as follows:

$$\mathbf{w} = \frac{M_+}{M}\bar{\boldsymbol{\phi}}_+ - \frac{C_{\mathrm{m}}\, M_-}{M}\bar{\boldsymbol{\phi}}_-, \tag{28}$$

where C_{m} works to lengthen or shorten the length of vector $\bar{\boldsymbol{\phi}}_-$ according to whether $C_{\mathrm{m}} > 1$ or $0 < C_{\mathrm{m}} < 1$. If we change C_{m} from a small value to a large one, $\mathbf{w}$ rotates in the feature space. Therefore, by changing the value of C_{m}, the slope of the decision function is changed.

Then the decision function becomes

$$f(\mathbf{x}) = \frac{1}{M}\sum_{i=1,y_i=1}^{M} K(\mathbf{x},\mathbf{x}_i) - \frac{C_{\mathrm{m}}}{M}\sum_{i=1,y_i=-1}^{M} K(\mathbf{x},\mathbf{x}_i) + b. \tag{29}$$

If we use RBF kernels, we need to determine the γ and C_{m} values before training the classifier. Consider determining the values by cross-validation. If we evaluate all combinations of the discrete γ and C_{m} values and select the best combination, we call this strategy grid search and call the resulting classifier $\mathrm{MAMC}_{\mathrm{bsg}}$. To speed this up, we consider line search: first we determine the γ value with $C_{\mathrm{m}} = 1$ by cross-validation. Then fixing the γ value with the determined value we determine the C_{m} value by cross-validation. We call the resultant classifier $\mathrm{MAMC}_{\mathrm{bs}}$.

In k-fold cross-validation, we divide the training data set into k almost-equal-size subsets and train the classifier using the $k-1$ subsets and test the trained classifier using the remaining subset. We iterate this procedure k times for different combinations and calculate the classification error.

Calculation of the classification error for the given C_{m} and γ values is as follows:

1. Calculate (29) with $b = 0$ using the $k-1$ subsets.
2. Calculate the bias term using the method discussed in Section 3.1.
3. Calculate the classification error for the remaining subset using the decision function generated in Steps 1 and 2.
4. Repeat the above procedure for the k different combinations of the subsets and calculate the total classification error.

We select the C_{m} and γ values with the minimum classification error among all the evaluated combinations of C_{m} and γ values.

For multiclass problems, the optimum value of C_{m} may be different for different decision functions. Thus, we consider setting a different value of C_{m} for each decision function. For an n-class problem, where n is a positive integer larger than 1, we need to determine $n\,(n-1)/2$ decision functions for one-against-one classification. If we evaluate cross-validation by the classification error by one-against-one classification, we need to evaluate the classification errors $p^{n\,(n-1)/2}$ times, where p is the number of discrete C_{m} values used in cross-validation. Thus the computation cost is prohibitive. To shorten the model selection time, for each decision function, we evaluate the classification error and determine the C_{m} value with the smallest classification error. By this method, for each value of C_{m}, the classification errors for all the decision functions need to be calculated $n\,(n-1)/2$ times. This is the same number as that for one-against-one classification.

By the above procedure, one-against-one classification error is not calculated but the minimization of the classification errors for all the pairs of classes is considered to give a near optimal model.

3.3 Equality-Constrained MAMCs

To improve the generalization ability of MAMCs further, we consider equality-constrained MAMCs (EMAMCs) as follows:

$$\text{maximize} \quad Q(\mathbf{w}, b, \boldsymbol{\xi}) = -\frac{1}{2}\mathbf{w}^\top \mathbf{w} + \frac{C_\mathrm{a}}{M}\sum_{i=1}^{M} y_i \left(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) + b\right) - \frac{C}{2}\sum_{i=1}^{M} \xi_i^2 \tag{30}$$

$$\text{subject to} \quad y_i \left(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) + b\right) = 1 - \xi_i \quad \text{for} \quad i = 1, \ldots, M, \tag{31}$$

where C_a and C are parameters to control the trade-off between the generalization ability and the classification error for the training data, ξ_i are the slack variables for $\mathbf{x}_i$, and $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_M)^\top$.

If we delete the second term of the right-hand side of (30), the EMAMC reduces to the least squares (LS) SVM [2]. The first term is to maximize the minimum margin and the third term is to minimize the square sum of slack variables: As the C value becomes larger, the margins for the training data are more enforced to 1. The second term is to maximize the average margin; the larger value of C_a makes the larger average margin. In the following, however, we show that C_a does not work.

We solve (30) and (31) in the empirical feature space in the primal form [2]. Usually, the dimension of the empirical feature space is lower than M, but here we use the following mapping function to avoid selecting linearly independent data:

$$\boldsymbol{\phi}(\mathbf{x}) = (K(\mathbf{x}, \mathbf{x}_1), \ldots, K(\mathbf{x}, \mathbf{x}_M))^\top. \tag{32}$$

Solving (31) for ξ_i and substituting it into (30), we obtain the unconstrained optimization problem:

$$\begin{aligned} \text{maximize} \quad & Q(\mathbf{w}, b) = -\frac{1}{2}\mathbf{w}^\top \mathbf{w} + \frac{C_\mathrm{a}}{M}\sum_{i=1}^{M} y_i \left(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) + b\right) \\ & -\frac{C}{2}\sum_{i=1}^{M} (1 - y_i \left(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i) + b\right))^2. \end{aligned} \tag{33}$$

As explained before, if we delete the second term (the average margin) in the above equation, the optimization problem results in the least squares (LS) SVM defined in the empirical feature space [2].

Taking the partial derivative of (33) with respect to $\mathbf{w}$ and b and setting the results to zero, we obtain the optimality conditions:

$$\left(1 + C\sum_{i=1}^{M} \boldsymbol{\phi}(\mathbf{x}_i)\, \boldsymbol{\phi}^\top(\mathbf{x}_i)\right)\mathbf{w} + C\sum_{i=1}^{M} \boldsymbol{\phi}(\mathbf{x}_i)\, b = \left(\frac{C_\mathrm{a}}{M} + C\right)\sum_{i=1}^{M} y_i\, \boldsymbol{\phi}(\mathbf{x}_i), \tag{34}$$

$$C\sum_{i=1}^{M} \boldsymbol{\phi}^\top(\mathbf{x}_i)\, \mathbf{w} + C\, M\, b = \left(\frac{C_\mathrm{a}}{M} + C\right)\sum_{i=1}^{M} y_i. \tag{35}$$

The above optimality conditions can be solved for $\mathbf{w}$ and b by matrix inversion. The coefficient $(C_\mathrm{a}/M + C)$ can be deleted because it is a scaling factor and does

not change the decision boundary. Then, because $C_{\rm a}$ is not included in the left-hand sides of (34) and (35), the value of $C_{\rm a}$ does not influence the location of the decision boundary. This means that the second term in (33) can be safely deleted.

In addition, if we delete the $\mathbf{w}^\top\,\mathbf{w}$ term from (33), all the terms in the left-hand sides of (34) and (35) include C, thus C can be deleted; C does not work to control the trade-off. Therefore, the $\mathbf{w}^\top\,\mathbf{w}$ term is essential.

Accordingly, dividing (34) and (35) by C and deleting the constant term $(C_{\rm a}/C\,M+1)$ from the right-hand sides of (34) and (35), we obtain

$$\left(\frac{1}{C}+\sum_{i=1}^{M}\boldsymbol{\phi}(\mathbf{x}_i)\,\boldsymbol{\phi}^\top(\mathbf{x}_i)\right)\mathbf{w}+\sum_{i=1}^{M}\boldsymbol{\phi}(\mathbf{x}_i)\,b=\sum_{i=1}^{M}y_i\,\boldsymbol{\phi}(\mathbf{x}_i), \tag{36}$$

$$\sum_{i=1}^{M}\boldsymbol{\phi}^\top(\mathbf{x}_i)\,\mathbf{w}+M\,b=\sum_{i=1}^{M}y_i. \tag{37}$$

The above formulation is exactly the same as the LS SVM defined in the empirical feature space.

Now we solve (30) and (31) in the dual form. First we multiply the terms in (31) with y_i and change $y_i\,\xi_i$ to y_i as follows:

$$\mathbf{w}^\top\boldsymbol{\phi}(\mathbf{x}_i)+b-y_i+\xi_i=0\quad\text{for}\quad i=1,\ldots,M. \tag{38}$$

Then, introducing the Lagrange multipliers $\alpha_i(i=1,\ldots,M)$, we convert the constrained problem into the unconstrained one:

$$\begin{aligned}\text{maximize } Q(\mathbf{w},b,\boldsymbol{\alpha},\boldsymbol{\xi}) &= -\frac{1}{2}\mathbf{w}^\top\,\mathbf{w}+\frac{C_{\rm a}}{M}\sum_{i=1}^{M}y_i\,(\mathbf{w}^\top\boldsymbol{\phi}(\mathbf{x}_i)+b)-\frac{C}{2}\sum_{i=1}^{M}\xi_i^2\\ &\quad+\sum_{i=1}^{M}\alpha_i\,(\mathbf{w}^\top\boldsymbol{\phi}(\mathbf{x}_i)+b-y_i+\xi_i)\\ &= -\frac{1}{2}\mathbf{w}^\top\,\mathbf{w}+\frac{C_{\rm a}}{M}\sum_{i=1}^{M}(1-y_i\,\xi_i)-\frac{C}{2}\sum_{i=1}^{M}\xi_i^2\\ &\quad+\sum_{i=1}^{M}\alpha_i\,(\mathbf{w}^\top\boldsymbol{\phi}(\mathbf{x}_i)+b-y_i+\xi_i)\end{aligned} \tag{39}$$

Taking the partial derivatives of (39), with respect to $\mathbf{w}$, b, and ξ_i, and setting the results to zero, together with the equality constraints, we obtain the following optimality conditions:

$$\mathbf{w}=\sum_{i=1}^{M}\alpha_i\,\boldsymbol{\phi}(\mathbf{x}_i), \tag{40}$$

$$\sum_{i=1}^{M}\alpha_i=0, \tag{41}$$

$$C\,\xi_i=\alpha_i-\frac{C_{\rm a}}{M}\,y_i\quad\text{for}\quad i=1,\ldots,M, \tag{42}$$

$$\mathbf{w}^\top\boldsymbol{\phi}(\mathbf{x}_i)+b-y_i+\xi_i=0\quad\text{for}\quad i=1,\ldots,M, \tag{43}$$

Substituting (40) and (42) into (43) and arranging the resultant equations and (41) into matrix form:

$$\begin{pmatrix} K + \frac{1}{C} I & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ b \end{pmatrix} = \left(1 + \frac{C_{\mathrm{a}}}{C\,M}\right) \begin{pmatrix} \mathbf{y} \\ 0 \end{pmatrix}, \tag{44}$$

where $K = \{K(\mathbf{x}_i, \mathbf{x}_j)\}\,(i, j = 1, \ldots, M)$, I is the $M \times M$ unit matrix, and $\mathbf{1}$ is the M-dimensional column vector whose elements are 1.

Because $1 + C_{\mathrm{a}}/(C\,M)$ is a scale factor and does not change the direction of $\mathbf{w}$, the dual EMAMC is equivalent to the LS SVM.

The dual EMAMC (LS SVM) is equivalent to the primal EMAMC if, instead of (32) the following mapping function is used [25]:

$$\boldsymbol{\phi}(\mathbf{x}) = \Lambda^{-1/2}\, P^\top\, (K(\mathbf{x}_1, \mathbf{x}), \ldots, K(\mathbf{x}_M, \mathbf{x}))^\top, \tag{45}$$

where $K = P\,\Lambda\,P^\top$, Λ is the diagonal matrix with the positive eigenvalues of K, the column vectors of P consist of the eigenvectors of K associated with the positive eigenvectors. If the rank of K is $N(\leq M)$, the empirical feature space is the N-dimensional subspace.

But if we use (32), the input space is mapped into the same empirical feature space but with coordinates different from those by (45). Therefore, so long as we use mapping function (32), the primal EMAMC is not equivalent to the LS SVM. In the following we simply call the primal EMAMC, EMAMC.

3.4 Computational Complexity

We investigate the computational complexity of the classifiers from the number of distinct kernels necessary for training; training complexity; and the number that cross-validation is repeated. The results are shown in Table 1. In the following we explain the table in detail.

Calculations of kernels occupy a large part of training time and thus the kernel values once calculated are usually stored into a memory for a later use. Therefore, here we evaluate the number of distinct kernels calculated during training.

In training, the MAMC does not require any computation in deriving (12) and (13). But in cross-validation, we need to evaluate (1) for all the training data. Therefore, we include calculating the kernels in training. Because (12) is expressed using all the training data, all the kernels associated with the training data need to be calculated: $K(\mathbf{x}_i, \mathbf{x}_j)$ for $i, j = 1, \ldots, M$. Because $K(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_j, \mathbf{x}_i)$ and for the RBF kernels $K(\mathbf{x}_i, \mathbf{x}_i) = 1$, the number of distinct kernels calculated is $M\,(M-1)/2$. The same is true for the $\mathrm{MAMC_b}$, $\mathrm{MAMC_{bs}}$, and $\mathrm{MAMC_{bsg}}$. For the EMAMC, so long as we use (32) or (45), the number of kernels is the same as that for the MAMC. For the L1 SVM, if the number of distinct data that are selected as support vector candidates during training is $M'\,(\leq M)$, the number of distinct kernels calculated is at most $M'\,(M'-1)/2$. Therefore, the number of distinct kernels for the L1 SVM is the smallest.

In investigating training complexity, we assume that the kernels have been calculated. Then the major task of training MAMC including cross-validation

Table 1 Computational complexity of the classifiers

Item	MAMC	MAMC_b	MAMC_bsg	MAMC_bs	EMAMC	L1 SVM
Kernels	$M(M-1)/2$	$M(M-1)/2$	$M(M-1)/2$	$M(M-1)/2$	$M(M-1)/2$	$M'(M'-1)/2$
Training	$O(M^2)$	$O(M^2)$	$O(M^2)$	$O(M^2)$	$O(M^3)$	$O(N\,W^3)$
CV	N_γ	N_γ	$N_\gamma\,N_{C_\text{m}}$	$N_\gamma + N_{C_\text{m}}$	$N_\gamma\,N_C$	$N_\gamma\,N_C$

is to calculate the values of (1) for all the training data: $O(M^2)$ additions or subtractions.

In MAMC_b, the critical computation is sorting for implementing (22) and (23); the computational complexity of sorting is $O(M^2)$. In MAMC_bs and MAMC_bsg, the crucial part of calculations is determination of the C_m value by cross-validation. Excluding cross-validation, the complexity is equivalent to that of MAMC_b. The EMAMC is trained by solving the set of the $(M+1)$th order linear equations and thus the complexity of computation is $O(M^3)$. The L1 SVM is usually trained by solving a quadratic programing problem with M variables. Using the decomposition technique [26] with the working set size of $W\,(\geq 2)$, the complexity of computation is $O(N\,W^3)$, where N is the number of iterations.

In cross validation, MAMC and MAMC_b need to determine the value of γ. Let the number of γ values evaluated by cross-validation be N_γ. Then, cross-validation needs to be repeated N_γ times. For MAMC_bsg, the γ and C_m values are determined by grid search. Therefore, cross-validation needs to be repeated $N_\gamma\,N_{C_\text{m}}$ times, where N_{C_m} is the number of C_m values evaluated by cross-validation. While for MAMC_bs, the γ and C_m values are determined by line search. Therefore, cross validation is repeated $N_\gamma + N_{C_\text{m}}$ times. For the EMAMC and L1 SVM, the γ and C values are determined by grid search. Therefore, cross-validation is repeated $N_\gamma\,N_C$ times, where N_C is the number of C values evaluated.

According to the above analysis, the MAMC, MAMC_b, and MAMC_bs are in a fastest group and the EMAMC is the slowest especially for a large data set. This is because the set of the $(M+1)$th order linear equations needs to be solved. Speeding up equivalent LS SVM has been discussed extensively. For instance in [27], an iterative procedure with momentum terms is proposed. And in [28,29], instead of using (32), selected $M'\,(\leq M)$ data are used for mapping to the feature space. In this method, the number of kernels to be calculated is reduced to $M'\,(2M-M'-1)/2$ and the training complexity is reduced to $O(M'^3)$. But to avoid obtaining an inexact solution caused by iterative methods or approximation of the mapping function, we train the EMAMC by solving a set of the $(M+1)$th order linear equations.

4 Performance Evaluation

We compared the proposed MAMC including the EMAMC (primal LS SVM) with the plain MAMC and the L1 SVM, which is the most widely-used SVM architecture, using two-class data sets [30] and multiclass data sets.

Table 2 Benchmark data sets for two-class problems

Data	Inputs	Train	Test	Sets	Prior (%)
Banana	2	400	4,900	100	54.62±2.50
Breast cancer	9	200	77	100	70.59±1.75
Diabetes	8	468	300	100	65.03±1.38
Flare-solar	9	666	400	100	55.25±1.09
German	20	700	300	100	69.92±0.84
Heart	13	170	100	100	55.53±2.36
Image	18	1,300	1,010	20	57.40±0.86
Ringnorm	20	400	7,000	100	50.27±2.40
Splice	60	1,000	2,175	20	51.71±1.41
Thyroid	5	140	75	100	69.51±2.34
Titanic	3	150	2,051	100	67.83±4.08
Twonorm	20	400	7,000	100	50.52±2.27
Waveform	21	400	4,600	100	66.90±2.22

4.1 Experimental Conditions

The L1 SVM we used is as follows:

$$\text{maximize} \quad Q(\boldsymbol{\alpha}) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j \, y_i \, y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{46}$$

$$\text{subject to} \quad \sum_{i=1}^{M} y_i \, \alpha_i = 0, \quad 0 \le \alpha_i \le C \quad \text{for} \quad i = 1, ..., M, \tag{47}$$

where α_i are Lagrange multipliers associated with $\mathbf{x}_i$ and $C\,(> 0)$ is a margin parameter that controls the trade-off between the classification error of the training data and the generalization ability.

Table 2 lists the numbers of inputs, training data, test data, data set pairs of the two-class problems, and the maximum prior class probability shown in percent with the standard deviation. (Here, the non-zero standard deviation shows the variation of the ratio of Class 1 data to Class 2 data.) The maximum prior class probability is calculated by the maximum number of class data divided by the total number of data and is used as a lower bound in evaluating poor classifiers. Each data set pair consists of the training data set and the test data set. We trained classifiers using the training data set and evaluated the performance using the test data set. Then we calculated the average accuracy and the standard deviation for all the data set pairs. We determined the parameter values by fivefold cross-validation. The parameter values are often selected from powers of two but they are minute for the values smaller than 1 and coarse for the values larger than 1. Therefore, in the following we used the values different from the powers of 2.

We selected the γ value of the RBF kernels from {0.01, 0.1, 0.5, 1, 5, 10, 15, 20, 50, 100, 200, 300, 400, 500, 600, 700}. And we selected the C_{m} value from {0.05, 0.1, 0.2, . . . ,0.9,1.0, 1/0.9,. . . , 1/0.2, 1/0.1, 1/0.05}. For the EMAMC and L1 SVM, we selected the γ value from 0.01 to 200 and the C value, from {0.1, 1, 10, 50, 100, 500, 1000, 2000}. We trained the EMAMC by solving the set of the $(M + 1)$th order linear equations and trained the L1 SVM using SMO-NM [31], which fuses SMO (Sequential minimal optimization) and NM (Newton's method).

Table 3 Benchmark data for multiclass problems

Data	Inputs	Classes	Train	Test	Prior (%)
Numeral [2]	12	10	810	820	10.00
Thyroid [32]	21	3	3,772	3,428	92.47
Blood cell [2]	13	12	3,097	3,100	12.92
Hiragana-50 [2]	50	39	4,610	4,610	5.64
Hiragana-13 [2]	13	38	8,375	8,356	6.29
Hiragana-105 [2]	105	38	8,375	8,356	6.29
Satimage [32]	36	6	4,435	2,000	24.17
USPS [33]	256	10	7,291	2,007	16.38
Letter [32]	16	26	16,000	400	4.05

For multiclass problems, we used nine data sets listed in Table 3. We trained the classifiers using fuzzy one-against-one classification [2], to resolve unclassifiable regions occurred in multiclass problems. Because we need to train $n(n-1)/2$ classifiers for each problem, where n is the number of classes, we only used one training data set and one test data set to avoid long computation time.

4.2 Results for Two-class Problems

Table 4 shows the average accuracies and their standard deviations of the six classifiers with RBF kernels. In the table, MAMC is given by (12) and (13) and the γ value is optimized by cross-validation. In MAMC_b, the bias term is optimized as discussed in Section 3.1. In MAMC_bs, γ and C_m values are optimized by line search: after γ value is optimized with $C_\text{m} = 1$, the C_m value is optimized. In MAMC_bsg, the γ and C_m values are optimized by grid search. The EMAMC is trained using (36) and (37) and the L1 SVM is trained using (46) and (47).

Among the six classifiers including the L1 SVM the best average accuracy is shown in bold and the worst average accuracy is underlined. The "Average" row shows the average accuracy of the 13 average accuracies and the two numerals in the parentheses show the numbers of the best and worst accuracies in the order. We performed Welch's t test with the confidence intervals of 95%. The "W/T/L" row shows the results; W, T, and L denote the numbers that the MAMC_bs shows statistically better than, the same as, and worse than the remaining five classifiers, respectively.

From the "Average" row, the EMAMC performs best in the average accuracy and the L1 SVM the second best. The difference between MAMC_bsg and MAMC_bs is small. The MAMC is the worst. For image and Titanic problems, the average accuracies are even worse than the associated prior class probabilities shown in Table 2. The MAMC_b is the second worst. But the MAMC_b performed best for the breast cancer and ringnorm problems. In general, it is difficult to say why one classifier performs best for a specific problem. But for the ringnorm data set, because each class is generated by a normal distribution, the decision function by the MAMC_b discussed in Section 3.2 seems to work well.

From the "W/T/L" row, the accuracies of the MAMC_bs and the MAMC_bsg are statistically comparable and the accuracy of the MAMC_bs is slightly better than that of the the MAMC_b but always better than that of the MAMC. The accuracy of the MAMC_bs is worse than that of the EMAMC and L1 SVM.

Table 4 Accuracy comparison for the two-class problems

Data	MAMC_{bs}	MAMC_{bsg}	MAMC_{b}	MAMC	EMAMC	L1 SVM
Banana	88.46±0.85	88.49±0.86	88.51±0.73	59.69±9.17	89.13±0.63	**89.17**±0.72
B. cancer	74.14±4.33	74.00±4.43	**74.27**±4.36	71.19±4.53	73.57±4.55	73.03±4.51
Diabetes	73.03±2.26	72.19±2.49	72.66±2.32	65.22±2.15	**76.67**±1.76	76.29±1.73
Flare-solar	66.10±2.00	65.64±2.05	66.35±2.03	59.48±5.83	66.25±2.00	**66.99**±2.12
German	75.53±2.18	75.93±2.21	69.51±2.28	70.18±1.95	**76.27**±2.04	75.95±2.24
Heart	81.00±3.58	80.99±3.24	81.32±3.38	58.87±8.84	82.70±3.70	**82.82**±3.37
Image	93.59±1.22	94.11±1.24	92.90±1.12	56.81±1.10	96.97±0.74	**97.16**±0.41
Ringnorm	**98.27**±0.27	98.25±0.30	**98.27**±0.25	70.26±21.99	98.04±0.43	98.14±0.35
Splice	84.43±1.11	85.17±0.88	84.08±0.99	54.57±8.76	**89.07**±0.59	88.89±0.91
Thyroid	95.15±2.24	95.41±2.30	95.11±2.17	70.25±4.36	**95.43**±2.35	95.35±2.44
Titanic	77.68±0.84	**77.70**±1.03	77.64±0.84	67.69±0.30	77.69±0.82	77.39±0.74
Twonorm	97.28±0.31	97.21±0.40	97.35±0.27	79.25±16.20	**97.41**±0.23	97.38±0.26
Waveform	88.93±1.53	89.23±1.29	88.84±1.64	67.07±0.19	**90.20**±0.50	89.76±0.66
Average	84.12 (1/0)	84.18 (2/0)	83.60 (2/1)	65.43 (0/11)	85.34 (6/0)	85.26 (4/0)
W/T/L	—	1/11/1	1/12/0	13/0/0	1/4/8	2/3/8

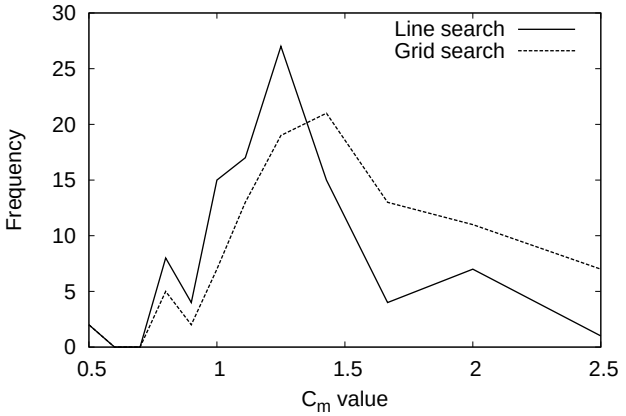

Fig. 1 Distribution of C_{m} values for the diabetes problem

In Section 2.2, we clarified that the bias term is not optimized by the original MAMC formulation. This is exemplified by the experiments. By optimizing the bias term as proposed in Section 3.1, the accuracy is improved drastically. The effect of slope optimization to the accuracy is small. However, by the bias term and slope optimization, the generalization ability is still below that of EMAMC or L1 SVM. This indicates that the equality or inequality constraints are essential in realizing the high generalization ability.

Figure 1 shows the frequency of C_{m} values for the diabetes problem selected by line search and grid search of γ and C_{m} values. For both search methods, the C_{m} values larger than 1 tend to be selected and this tendency is clearer for the grid search. For the diabetes problem, line search gives the statistically better results than by grid search.

We measured the average CPU time per data set including time for model selection by fivefold cross-validation, training a classifier, and classifying the test

Table 5 Computation time comparison for the two-class problems (in seconds)

Data	MAMC_{bs}	MAMC_{bsg}	MAMC_{b}	MAMC	EMAMC	L1 SVM
Banana	1.10	9.74	**0.59**	**0.59**	22.40	4.92
B. cancer	0.31	2.77	**0.13**	0.14	2.95	7.08
Diabetes	1.51	14.39	0.78	**0.73**	35.86	22.96
Flare-solar	3.24	29.89	1.61	**1.51**	111.43	218.67
German	4.42	40.03	2.07	**2.04**	148.65	776.53
Heart	0.24	2.13	0.12	**0.11**	2.05	1.75
Image	15.04	144.43	7.62	**7.18**	2290.87	56.7
Ringnorm	1.60	12.91	0.99	**0.96**	23.27	12.57
Splice	13.34	126.51	7.16	**6.87**	887.16	30.71
Thyroid	0.15	1.35	**0.07**	**0.07**	1.16	0.33
Titanic	0.17	1.31	**0.09**	**0.09**	1.32	21.25
Twonorm	1.67	12.92	0.99	**0.92**	22.51	10.46
Waveform	1.51	12.84	**0.88**	**0.88**	22.44	35.61
B/W	0/0	0/1	5/0	12/0	0/7	0/5

data by the trained classifier. We used a personal computer with 3.4GHz CPU and 16GB memory. Table 5 shows the results. The shortest computation time is shown in bold and the longest time is underlined. The last row of the tables shows the numbers of shortest/longest computation time among 13 problems. From the table the MAMC is the fastest and the MAMC_{bs} and MAMC_{b} are comparable to the MAMC. Comparing the MAMC_{bs} and the MAMC_{bsg}, the MAMC_{bsg} requires more time because of the grid search. Because the classification performance is comparable, line search seems to be sufficient. The EMAMC, L1 SVM and MAMC_{bsg} are in the slowest group.

4.3 Results for Multiclass Problems

Table 6 shows the accuracy of the six classifiers using RBF kernels for one set of training and test data. The "Average" row shows the average accuracies for nine data sets and the "B/W" row shows the numbers that the classifiers take the best/worst accuracies. From the standpoint of average accuracies, the L1 SVM performed best and the EMAMC, the second best. And from the "B/W" standpoint, the EMAMC performed best and the L1 SVM, the second best. Thus, the EMAMC and the L1 SVM are considered to be comparable. From the "Average" and "B/W" standpoints, the MAMC is the worst and the MAMC_{bs}, MAMC_{bsg}, and the MAMC_{b} are comparable.

Figure 2 shows the distribution of C_{m} values selected by line search and grid search for the blood cell problem. The C_{m} value of 1 is most frequently selected by both methods and the distributions of both methods are similar.

Table 7 shows the execution time of the six classifiers. The time includes model selection by fivefold cross-validation, training for the selected parameter values, and classifying the test data. The "B/W" row shows the numbers that the classifier shows the best/worst computation time among nine data sets. From the table, MAMC is the fastest. For the four MAMC based classifiers, the MAMC, MAMC_{b}, and MAMC_{bs} show comparable computation time but the MAMC_{bsg} is the slowest. Therefore, for the multiclass problems also, MAMC_{bs} is sufficient to realize the comparable generalization ability of MAMC_{bsg} in a shorter training time.

Table 6 Accuracy comparison of the multiclass problems

Data	MAMC_{bs}	MAMC_{bsg}	MAMC_{b}	MAMC	EMAMC	L1 SVM
Numeral	99.15	99.02	99.15	99.27	99.63	**99.76**
Thyroid	93.06	93.32	92.63	92.71	94.84	**97.26**
Blood cell	88.81	88.03	88.45	50.00	**93.97**	93.16
Hiragana-50	96.62	96.44	96.51	22.76	**99.26**	99.00
Hiragana-13	97.71	97.71	97.38	20.63	**99.89**	99.79
Hiragana-105	99.96	99.96	99.94	19.47	**100.00**	**100.00**
Satimage	88.20	87.60	87.75	63.95	**92.10**	91.90
USPS	87.20	87.20	84.11	30.89	**95.52**	95.27
Letter	93.30	93.30	92.30	56.18	97.83	**97.85**
Average	93.78	93.62	93.13	50.65	97.00	**97.11**
B/W	0/0	0/1	0/1	0/7	6/0	4/0

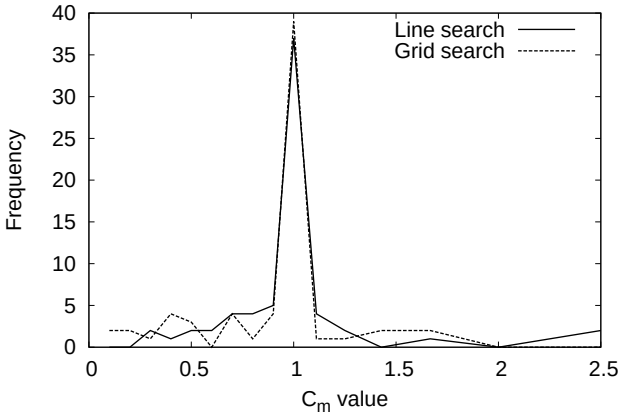

Fig. 2 Distribution of C_{m} values for the blood cell problem

The EMAMC is the slowest because we trained the EMAMC by solving the set of the $(M+1)$th order linear equations. As we explained in Section 3.4, we can speed up training by iterative methods or selection of the training data for mapping the input space into the feature space.

4.4 Discussions

The advantage of the MAMC is its simplicity: The coefficient vector of the decision hyperplane is calculated by addition or subtraction of kernel values. The major disadvantage, however, is the inferior generalization ability of the original MAMC. This is mitigated by bias and slope optimization, but the improvement is still not sufficient compared to the EMAMC and L1 SVM. Therefore, the introduction of the equality or inequality constraints are essential. But it leads to the LS SVM or L1 SVM and the simplicity of the MAMC is completely lost.

Table 7 Execution time comparison of the multiclass problems (in seconds)

Data	$MAMC_{bs}$	$MAMC_{bsg}$	$MAMC_{b}$	MAMC	EMAMC	L1 SVM
Numeral	2.48	13.22	2.08	**1.88**	60.89	11.84
Thyroid	153.45	1,197.72	75.06	**60.92**	152,356.61	518.41
Blood cell	81.75	500.41	53.50	**50.25**	4,785.94	129.75
Hiragana-50	429.52	1,116.03	402.44	**396.48**	5,941.55	1,465.81
Hiragana-13	1,596.80	4,999.03	1,398.11	1,369.63	47,965.70	**1,225.30**
Hiragana-105	2,137.48	5,700.22	2,018.88	**1,983.59**	45,391.41	14,501.95
Satimage	187.06	1,371.38	101.89	**88.77**	66,903.45	782.53
USPS	956.72	4,359.95	693.11	**658.30**	177,728.94	277,967.67
Letter	15,474.63	56,645.84	12,094.63	**11,988.27**	661,397.75	17,162.48
B/W	0/0	0/0	0/0	8/0	0/8	1/1

5 Conclusions

We discussed three ways to improve the generalization ability of the maximum average margin classifier (MAMC). The first one is to optimize the bias term after calculating the weight vector, and the second one is to optimize the slope of the decision function by introducing the weight parameter to the average vector of one class. The parameter value is determined by cross-validation. To improve the generalization ability further, in the third way, we introduced the EMAMC, which is the equality constrained MAMC, but this is shown to be equivalent to the LS SVM defined in the empirical feature space.

According to the computer experiments for two-class and multiclass problems, we show that the generalization ability is improved by the bias term and slope optimization. However, the obtained generalization ability is inferior to the EMAMC and L1 SVM. Therefore, the unconstrained MAMC is not so powerful as the EMAMC and L1 SVM are.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 25420438.

References

1. V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, NY, 1998.
2. S. Abe. *Support Vector Machines for Pattern Classification*. Springer-Verlag, London, UK, second edition, 2010.
3. G. R. G. Lanckriet, L. El Ghaoui, C. Bhattacharyya, and M. I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3:555–582, 2002.
4. K. Huang, H. Yang, I. King, and M. R. Lyu. Learning large margin classifiers locally and globally. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML 2004)*, pages 1–8, 2006.
5. D. S. Yeung, D. Wang, W. W. Y. Ng, E. C. C. Tsang, and X. Wang. Structured large margin machines: sensitive to data distributions. *Machine Learning*, 68(2):171–200, 2007.
6. H. Xue, S. Chen, and Q. Yang. Structural regularized support vector machine: A framework for structural large margin classifier. *IEEE Transactions on Neural Networks*, 22(4):573–587, 2011.
7. X. Peng and D. Xu. Twin Mahalanobis distance-based support vector machines for pattern recognition. *Information Sciences*, 200:22–37, 2012.

8. S. Abe. Training of support vector machines with Mahalanobis kernels. In W. Duch, J. Kacprzyk, E. Oja, and S. Zadrożny, editors, *Artificial Neural Networks: Formal Models and Their Applications (ICANN 2005)—Proceedings of Fifteenth International Conference, Part II, Warsaw, Poland*, pages 571–576. Springer-Verlag, Berlin, Germany, 2005.
9. D. Wang, D. S. Yeung, and E. C. C. Tsang. Weighted Mahalanobis distance kernels for support vector machines. *IEEE Transactions on Neural Networks*, 18(5):1453–1462, 2007.
10. C. Shen, J. Kim, and L. Wang. Scalable large-margin Mahalanobis distance metric learning. *IEEE Transactions on Neural Networks*, 21(9):1524–1530, 2010.
11. X. Liang and Z. Ni. Hyperellipsoidal statistical classifications in a reproducing kernel Hilbert space. *IEEE Transactions on Neural Networks*, 22(6):968–975, 2011.
12. M. Fauvel, J. Chanussot, J.A. Benediktsson, and A. Villa. Parsimonious Mahalanobis kernel for the classification of high dimensional data. *Pattern Recognition*, 46(3):845–854, 2013.
13. T. Reitmaier and B. Sick. The responsibility weighted Mahalanobis kernel for semi-supervised training of support vector machines for classification. *Information Sciences*, 323:179–198, 2015.
14. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
15. L. Reyzin and R. E. Schapire. How boosting the margin can also boost classifier complexity. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 753–760. ACM, 2006.
16. W. Gao and Z.-H. Zhou. On the doubt about margin explanation of boosting. *Artificial Intelligence*, 203:1–18, 2013.
17. A. Garg and D. Roth. Margin distribution and learning. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), 2003, Washington, DC, USA*, pages 210–217, 2003.
18. K. Pelckmans, J. Suykens, and B. D. Moor. A risk minimization principle for a class of Parzen estimators. In J.C. Platt, D. Koller, Y. Singer, and S.T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1137–1144. Curran Associates, Inc., 2008.
19. F. Aiolli, G. Da San Martino, and A. Sperduti. A kernel method for the optimization of the margin distribution. In V. Kůrková, R. Neruda, and J. Koutnik, editors, *Artificial Neural Networks (ICANN 2008)—Proceedings of the Eighteenth International Conference, Prague, Czech Republic, Part I*, pages 305–314. Springer-Verlag, Berlin, Germany, 2008.
20. L. Zhang and W.-D. Zhou. Density-induced margin support vector machines. *Pattern Recognition*, 44(7):1448–1460, 2011.
21. Z.-H. Zhou T. Zhang. Large margin distribution machine. In *Twentieth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 313–322, 2014.
22. Z.-H. Zhou. Large margin distribution learning. In *Artificial Neural Networks in Pattern Recognition*, pages 1–11. Springer, 2014.
23. S. Abe. Improving generalization abilities of maximal average margin classifiers. In F. Schwenker, H. M. Abbas, N. El Gayar, and E. Trentin, editors, *Artificial Neural Networks in Pattern Recognition: 7th IAPR TC3 Workshop, ANNPR 2016, Ulm, Germany, September 28–30, 2016, Proceedings*, pages 29–41. Springer International Publishing, Cham, 2016.
24. O. L. Mangasarian and D. R. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 10(5):1032–1037, 1999.
25. H. Xiong, M. N. S. Swamy, and M. O. Ahmad. Optimizing the kernel in the empirical feature space. *IEEE Transactions on Neural Networks*, 16(2):460–474, 2005.
26. E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Neural Networks for Signal Processing VII—Proceedings of the 1997 IEEE Signal Processing Society Workshop*, pages 276–285, 1997.
27. 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.
28. J. A. K. Suykens, L. Lukas, and J. Vandewalle. Sparse least squares support vector machine classifiers. In *Proceedings of the Eighth European Symposium on Artificial Neural Networks (ESANN 2000)*, pages 37–42, Bruges, Belgium, 2000.

29. S. Abe. Sparse least squares support vector training in the reduced empirical feature space. *Pattern Analysis and Applications*, 10(3):203–214, 2007.
30. G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001.
31. S. Abe. Fusing sequential minimal optimization and Newton's method for support vector training. *International Journal of Machine Learning and Cybernetics* (DOI: 10.1007/s13042-014-0265-x), 7(3):345–364, 2016.
32. A. Asuncion and D. J. Newman. UCI machine learning repository, http://www.ics.uci.edu/∼mlearn/MLRepository.html. 2007.
33. USPS Dataset. http://www-i6.informatik.rwth-aachen.de/˜keysers/usps.html.