Incremental training of support vector machines using hyperspheres

Katagiri, Shinya

Abe, Shigeo

(Citation)
Pattern Recognition Letters,27(13):1507-1507

(Issue Date)
2006-10-01

(Resource Type)
journal article

(Version)
Accepted Manuscript

(URL)
https://hdl.handle.net/20.500.14094/90000149

Incremental Training of Support Vector Machines Using Hyperspheres

Shinya Katagiri, Shigeo Abe

Graduate School of Science and Technology, Kobe University, Kobe, Japan

Abstract

In the conventional incremental training of support vector machines, candidates for support vectors tend to be deleted if the separating hyperplane rotates as the training data are added. To solve this problem, in this paper, we propose an incremental training method using one-class support vector machines. First, we generate a hypersphere for each class. Then, we keep data that exist near the boundary of the hypersphere as candidates for support vectors and delete others. By computer simulations for two-class and multiclass benchmark data sets, we show that we can delete data considerably without deteriorating the generalization ability.

Key words: Support Vector Machines; Incremental Training; Hyperspheres; One-class Support Vector Machines; Multiclass Support Vector Machines

1 Introduction

Support vector machines (SVMs) (Vapnik, 1995, 1998) are widely used for pattern classification because of their good generalization ability compared with conventional classifiers. In a support vector machine, the input space is mapped into the high dimensional feature space and the optimal separating hyperplane is determined in that space. Since training of a support vector machine is formulated as a quadratic optimization problem with the number of variables being equal to the number of training data, we can obtain the global optimal solution. In addition, among the training data, only support vectors, which are boundary data between classes, are necessary to determine the decision function. This characteristic alleviates one of the problems of support vector machines: training becomes slow for a large number of training data. Namely, we can select support vector candidates before training and reduce the number of training data, thus memory consumption.

Therefore, support vector machines are especially suited for incremental training (Mitra, Murthy, and Pal, 2000; Xiao, Wang, and Zhang, 2000; Cauwenberghs and Poggio, 2000; Pedroso and Murata, 2000; Domeniconi and Gunopulos, 2001; Ralaivola and d'Alché-Buc, 2001; Shilton et al., 2005), where training data are obtained incrementally. By incremental training we can speedup training and delete unnecessary training data.

To reduce the memory consumption, Mitra, Murthy, and Pal (2000); Domeniconi and Gunopulos (2001) proposed a simple incremental training method that deletes data other than support vectors at each incremental training step. But this method considers only support vectors at each step and thus training data, which may become support vectors for the addition of training data, may be deleted. Therefore to keep the generalization ability of the incremental support vector machines comparable with that of the batch support vector machines, we need to consider future candidates for support vectors. Based on the assumption that the separating hyperplane before incremental training does not move very much after incremental training, Cauwenberghs and Poggio (2000) proposed to set the region for data deletion based on the distance from the separating hyperplane before incremental training. But there may be cases where the data that are far away from the hyperplane may become support vectors if the hyperplane rotates by incremental training. Thus, this method is weak for the rotation of the separating hyperplane.

In this paper, we propose an incremental training method that is robust for the rotation of the separating hyperplane. The proposed method is based on the assumption that candidates for support vectors should exist near the separating hyperplane and be located close to the surface of a region that includes training data of each class. Based on this, first, we generate the minimum-volume hypersphere that includes the training data of each class (Tax and Duin, 2001), then we generate a concentric hypersphere with a smaller radius. Next, we generate the hypercone whose vertex is at the center of the hypersphere and which opens in the opposite direction of the separating hyperplane. We delete the data that exist inside of the small hypersphere or inside of the hypercone and keep the remaining data as candidates for support vectors. We extend this method for two-class problems to multiclass problems and we evaluate the effectiveness of the method using the benchmark data sets for two-class and multi-class problems.

The structure of this paper is as follows. In Section 2, we summarize the architecture of support vector machines, and in Section 3, we discuss the conventional incremental training method. In Section 4, we discuss the proposed method for two-class and multiclass problems and in Section 5, we show the simulation results using benchmark data sets.

2 Support Vector Machines

In this section, we summarize support vector machines for two-class problems. In support vector machines, we map the input vector $\mathbf{x}$ to the high dimensional feature space using the mapping function $\phi(\mathbf{x})$ to enhance linear separability. Let the M training input-output pairs be $(\mathbf{x}_i, y(\mathbf{x}_i))$, $i = 1, ..., M$, where $y(\mathbf{x}_i) = 1$ if $\mathbf{x}_i$ belong to Class 1, and $y(\mathbf{x}_i) = -1$ if Class 2. If the training data are linearly separable in the feature space, we can obtain the decision function:

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

where $\mathbf{w}$ is a weight vector, b is a bias term, and $y(\mathbf{x}_i) f(\mathbf{x}_i) > 0$ for $i = 1, ..., M$. For unknown data $\mathbf{x}$, if $f(\mathbf{x}) \geq 0$, the data are classified into Class 1, and if $f(\mathbf{x}) < 0$, into Class 2. The distance between the separating hyperplane and the training datum nearest to the hyperplane is called the margin. The hyperplane that has the maximum margin is called optimal separating hyperplane that separates two classes.

If the classification problem is not linearly separable in the feature space, the optimal separating hyperplane can be obtained by solving the following optimization problem:

Minimize

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

subject to

$$y(\mathbf{x}_i)(\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \quad \text{for} \quad i = 1, ..., M, \tag{3}$$

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

Introducing the Lagrange multipliers α_i, we obtain the following dual problem:

Maximize

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

subject to

$$\sum_{i=1}^{M} y(\mathbf{x}_i) \alpha_i = 0, \quad 0 \leq \alpha_i \leq C. \tag{5}$$

Here, $K(\mathbf{x}, \mathbf{x}')$ is a kernel function that is given by

$$K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}'). \tag{6}$$

Some of the kernels that are used in support vector machines are as follows:

Polynomial Kernels. The polynomial kernel with degree d is given by

$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + 1)^d. \tag{7}$$

Radial Basis Function Kernels. The radial basis function (RBF) kernel is given by

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma ||\mathbf{x} - \mathbf{x}'||^2), \tag{8}$$

where γ is a positive parameter for slope control.

For the solution of (4) and (5), if $\alpha_i > 0$, $\mathbf{x}_i$ are called support vectors; especially if $\alpha_i = C$, bounded support vectors and if $0 < \alpha_i < C$, unbounded support vectors. The special feature of support vector machines is that the same solution is obtained even if we delete all the non-support vectors from the training data.

Using the support vectors, the decision function is given by

$$f(\mathbf{x}) = \sum_{i \in S} \alpha_i y(\mathbf{x}_i) K(\mathbf{x}_i, \mathbf{x}) + b, \tag{9}$$

where S is the set of support vector indices. Here the weight vector $\mathbf{w}$ is given by

$$\mathbf{w} = \sum_{j \in S} y(\mathbf{x}_j) \alpha_j \phi(\mathbf{x}_j), \tag{10}$$

and the margin δ is given by

$$\delta = \frac{1}{||\mathbf{w}||} = \frac{1}{\sqrt{\sum_{j,k \in S} y(\mathbf{x}_j) y(\mathbf{x}_k) \alpha_j \alpha_k K(\mathbf{x}_j, \mathbf{x}_k)}}. \tag{11}$$

3 Conventional Incremental Training Method

3.1 Deletion of Data near the Separating Hyperplane

We explain how to select candidates for support vectors using a two-dimensional example shown in Fig. 1. In the figure, the initial data are shown in filled circles for Class 1 or rectangles for Class 2 and the added data are shown in open circles for Class 1 or rectangles for Class 2. The optimal hyperplane $f(\mathbf{x}) = 0$ that separates Class 1 from Class 2 with the margin δ is obtained by the initial data. If we retrain the support vector machine, the data that satisfy $y(\mathbf{x})f(\mathbf{x}) \leq 1$ are candidates for support vectors. In addition, the data that satisfy $y(\mathbf{x})f(\mathbf{x}) > 1$ but are near $y(\mathbf{x})f(\mathbf{x}) = 1$ can be support vectors. Therefore we determine that the data are candidates for support vectors if

$$y(\mathbf{x})f(\mathbf{x}) \leq \beta + 1, \tag{12}$$

where $\beta(> 0)$ is a user-defined parameter. If data satisfy (12), we keep the data, and otherwise we delete them, which exist in the shaded region in Fig. 1. But if all new data satisfy

$$y(\mathbf{x})f(\mathbf{x}) \geq 1, \tag{13}$$

the separating hyperplane after retraining is the same as the separating hyperplane before retraining. Thus in this case, we only need to add the data that satisfy (12) to the training data for future training.

Assume that we have an initial data set X_a and an additional data set X_b. The general procedure for incremental training is as follows:

(1) Train the support vector machine using the initial data set X_a.
(2) Add the additional data set X_b to X_a: $X_a = X_a \cup X_b$.
(3) If for $\mathbf{x} \in X_a$, (12) is not satisfied, delete $\mathbf{x}$ from X_a: $X_a = X_a - \{\mathbf{x}\}$.
(4) If for $\mathbf{x} \in X_a$, (13) is not satisfied, retrain the support vector machine.
(5) Repeat (2), (3), and (4).

3.2 Problem of the Conventional Method

In the conventional method, if the separating hyperplane after retraining is almost parallel to the separating hyperplane before retraining, it is easy to hold candidates for support vectors by user-defined parameter β. But if the

separating hyperplane after retraining rotates considerably, candidates for support vectors related to the rotation of the separating hyperplane tend to be deleted. We show this using examples shown in Figs. 2 to 5. Fig. 2 shows the separating hyperplane before adding data. Filled circles and rectangles show the initial data. Fig. 3 shows the state after data are added. In Fig. 4, we show the separating hyperplane after deleting data and retraining. Dotted circles and rectangles are the deleted data. Filled circles and rectangles are the deleted candidates for support vectors. In contrast to this figure, Fig. 5 shows the separating hyperplane determined by batch training using the initial and added data. Comparing Figs. 4 and 5, the separating hyperplanes are different because candidates for support vectors are deleted in incremental training.

To avoid this, namely, to hold candidates for support vectors related to the rotation of the separating hyperplane we must set a relatively large value to β. But if we set too large a value to β, we hold the unnecessary data that exist near the center of each class and cannot reduce the memory consumption.

4 Proposed Method

4.1 Concept

First, we explain how we can estimate the candidates for support vectors using the example shown in Fig. 6. In this figure, filled circles and rectangles show the initial data and open circles and rectangles show the added data. The separating hyperplane is determined by the initial data. Then after retraining, the new separating hyperplane may exist in the shaded region. Therefore we can predict that the candidates for support vectors exist near the boundary of the shaded region. This region can be approximated by the shaded regions in Fig. 7.

To keep the candidates for support vectors that exist in the shaded regions in Fig. 7, we consider approximating the regions using hyperspheres. We explain our idea using the example shown in Fig. 8. In the figure, the data in the shaded regions for Classes 1 and 2 are candidates for deletion, filled circles and rectangles show the remained data and open circles and rectangles show the deleted data after adding data. In the following, we explain how to approximate the shaded region for each class.

First, we generate the minimum-volume hypersphere that includes the training data of class j ($j = 1, 2$) with radius R_j. Next, we define a concentric hypersphere with radius ρR_j, where $\rho\,(0 < \rho < 1)$ is the user-defined parameter. Then, we define the hypercone whose vertex is at the center of the

hyperspheres and which opens in the opposite direction of the separating hyperplane. The user-defined parameter $\theta\,(-90 < \theta < 90)$ defines the angle between the separating hyperplane and the surface of the hypercone.

4.2 Deletion of Data

If the added data are in the shaded regions in Fig. 8, we delete the data. Now we explain how to delete datum $\mathbf{x}$ using Fig. 9. If the distance $r_j(\mathbf{x})$ between $\phi(\mathbf{x})$ and the center of the hypersphere, $\mathbf{a}_j$, is smaller than ρR_j:

$$r_j(\mathbf{x}) < \rho R_j, \tag{14}$$

we delete the data. Otherwise, if the angle between $\phi(\mathbf{x}) - \mathbf{a}_j$ and the separating hyperplane, $\psi_j(\mathbf{x})$, is larger than θ:

$$\psi_j(\mathbf{x}) > \theta, \tag{15}$$

we judge that $\phi(\mathbf{x})$ exists inside of the hypercone and delete $\mathbf{x}$.

But even if (14) or (15) is satisfied, if $\mathbf{x}$ satisfies

$$y(\mathbf{x}) f(\mathbf{x}) \leq 1, \tag{16}$$

$\mathbf{x}$ is a candidate for support vectors. In such a case, we do not delete $\mathbf{x}$. In addition, we do not delete the data that are support vectors for hyperspheres because the support vectors for hyperspheres are candidates for the support vectors for hyperspheres at the next training.

The general procedure for incremental training is as follows:

(1) Train the support vector machine using the initial data set X_a.
(2) Add the additional data set X_b to X_a: $X_a = X_a \cup X_b$.
(3) If for $\mathbf{x} \in X_a$, (16) is not satisfied and $\mathbf{x}$ satisfies $r_j(\mathbf{x}) < \rho R_j$, where j is the class label for $\mathbf{x}$ or $\psi_j(\mathbf{x}) > \theta$, delete $\mathbf{x}$ from X_a: $X_a = X_a - \{\mathbf{x}\}$.
(4) If for $\mathbf{x} \in X_a$, (16) is satisfied, retrain the support vector machine.
(5) Repeat (2), (3), and (4).

4.3 Generation of Hyperspheres

We approximate the region that includes training data of a class by a hypersphere (Tax and Duin, 2001). The procedure of generating the hypersphere

for class j ($j = 1, 2$) is as follows.

To generate the minimum-volume hypersphere that includes the data for class j, we need to solve the following optimization problem:

Minimize

$$Q_p(R_j, \mathbf{a}_j, \boldsymbol{\xi}^j) = R_j^2 + C_j \sum_{i \in X_j} \xi_i^j \tag{17}$$

subject to

$$||\phi(\mathbf{x}_i) - \mathbf{a}_j||^2 \leq R_j^2 + \xi_i^j, \quad \xi_i^j \geq 0 \quad \text{for} \quad i \in X_j \tag{18}$$

where X_j are sets of training data indices for class $j\,(j = 1, 2)$, $X_1 \cup X_2 = \{1, 2, \ldots, M\}$, $X_1 \cap X_2 = \emptyset$, $\mathbf{a}_j$ is the class j center, R_j is the radius of the class j hypersphere, ξ_i^j are slack variables, and C_j is the regularization parameter that determines the tradeoff between the volume of the hypersphere and outliers.

Introducing the Lagrange multipliers α_i^j, $\mathbf{a}_j$ is given by

$$\mathbf{a}_j = \sum_{i \in X_j} \alpha_i^j \phi(\mathbf{x}_i) \tag{19}$$

and we obtain the following dual problem:

Maximize

$$Q_d(\boldsymbol{\alpha}^j) = \sum_{i \in X_j} \alpha_i^j K(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,k \in X_j} \alpha_i^j \alpha_k^j K(\mathbf{x}_i, \mathbf{x}_k) \tag{20}$$

subject to

$$\sum_{i \in X_j} \alpha_i^j = 1, \quad 0 \leq \alpha_i^j \leq C_j \quad \text{for} \quad i \in X_j. \tag{21}$$

Using the unbounded support vector, which belongs to class j, R_j is given by

$$R_j = ||\phi(\mathbf{x}) - \mathbf{a}_j||. \tag{22}$$

The bounded support vectors with $\xi_i^j > 0$ are outside of the hypersphere and thus outliers. From (21), α_i^j do not exceed 1. Thus, if we set $C_j = 1$, all the support vectors are unbounded support vectors.

The mapping of an unknown datum $\mathbf{x}$ is inside of the hypersphere if

$$K(\mathbf{x}, \mathbf{x}) - 2 \sum_{i \in S_j} \alpha_i^j K(\mathbf{x}, \mathbf{x}_i) + \sum_{i,k \in S_j} \alpha_i^j \alpha_j^j K(\mathbf{x}_i, \mathbf{x}_k) \leq R_j^2, \tag{23}$$

where S_j is the set of support vector indices of the class j hypersphere.

4.4 Evaluation of $\psi_j(\mathbf{x})$

We discuss how to evaluate $\psi_j(\mathbf{x})$. The distance $r_j(\mathbf{x})$ between $\phi(\mathbf{x})$ and the center of the hypersphere is given by

$$\begin{aligned} r_j(\mathbf{x}) &= ||\phi(\mathbf{x}) - \mathbf{a}_j|| \\ &= \sqrt{K(\mathbf{x},\mathbf{x}) - 2\sum_{i\in S_j} \alpha_i^j K(\mathbf{x},\mathbf{x}_i) + \sum_{i,k\in S_j} \alpha_i^j \alpha_k^j K(\mathbf{x}_i,\mathbf{x}_k)}. \end{aligned} \tag{24}$$

From (10) and (19), the value of the decision function at the center of the hypersphere is given by

$$f(\mathbf{a}_j) = \mathbf{w}^T \mathbf{a}_j + b = \sum_{i\in S, k\in S_j} y(\mathbf{x}_i)\alpha_i \alpha_k^j K(\mathbf{x}_i,\mathbf{x}_k) + b. \tag{25}$$

The distance between $\phi(\mathbf{x})$ and the separating hyperplane is given by

$$\frac{|f(\mathbf{x})|}{||\mathbf{w}||} = \frac{y(\mathbf{x})f(\mathbf{x})}{||\mathbf{w}||} = y(\mathbf{x})f(\mathbf{x})\delta. \tag{26}$$

The distance between the separating hyperplane and the hyperplane, which is parallel to the separating hyperplane and which passes the center of the hypersphere, is given by

$$\frac{|f(\mathbf{a}_j)|}{||\mathbf{w}||} = \frac{y(\mathbf{x})f(\mathbf{a}_j)}{||\mathbf{w}||} = y(\mathbf{x})f(\mathbf{a}_j)\delta. \tag{27}$$

Thus the distance between $\phi(\mathbf{x})$ and the hyperplane which passes the center of the hypersphere is $|f(\mathbf{x}) - f(\mathbf{a}_j)|\delta$. Then if $y(\mathbf{x})(f(\mathbf{x}) - f(\mathbf{a}_j))\delta$ is negative, $\phi(\mathbf{x})$ lies between the separating hyperplane and the hyperplane, which is parallel to the separating hyperplane and which goes through $\mathbf{a}_j$. If it is positive, $\phi(\mathbf{x})$ and the separating hyperplane are on the opposite sides of the hyperplane which goes through $\mathbf{a}_j$. Therefore $\psi_j(\mathbf{x})$ $(-90 < \psi_j(\mathbf{x}) < 90)$ is given by

$$\psi_j(\mathbf{x}) = \sin^{-1}\left(\frac{y(\mathbf{x})(f(\mathbf{x}) - f(\mathbf{a}_j))\delta}{r_j(\mathbf{x})}\right). \tag{28}$$

If $\psi_j(\mathbf{x})$ is larger than θ, we delete $\mathbf{x}$. The deleting region using the hypercone is the shaded region shown in Fig. 9.

Support vector machines are formulated for two-class problems and there are several ways to extend to multiclass problems:

(1) one-against-all support vector machines (Vapnik, 1995, 1998),
(2) pairwise support vector machines (Kreßel, 1999),
(3) error-correcting-output code (ECOC) support vector machines (Bakiri and Dietterich, 1995),
(4) all-at-once support vector machines (Vapnik, 1995, 1998).

Among them, one-against-all and pairwise support vector machines are widely used and there is no much difference between their generalization abilities. However, for an n-class classification problem, a pairwise support vector machine needs to generate $n(n-1)/2$ decision functions, while a one-against-all support vector machine, n decision functions. Thus, because of simpler architecture, we use one-against-all support vector machines for multiclass problems. In one-against-all support vector machines, an n-class problem is converted into n two-class problems and for the ith two-class problem, class i is separated from the remaining classes. Thus we can apply the incremental training method for each decision function. Namely, for the ith decision function, we approximate class i and the remaining classes with hyperspheres. Then, if datum $\mathbf{x}$ does not satisfy $y(\mathbf{x})f(\mathbf{x}) \leq 1$ for all the n decision functions, and $\mathbf{x}$ is not support vectors for hyperspheres for all the classes, we delete $\mathbf{x}$.

5 Performance Evaluation

In this section we investigate how efficiently the proposed incremental training method deletes training data while keeping the generalization ability high. We also investigate the robustness of the method for the values of ρ and θ.

5.1 Benchmark Data and Evaluation Conditions

We use the benchmark data sets shown in Tables 1 and 2. Banana to waveform data sets in Table 1 [1] are the two-class data sets. Each data set consists of 100 (or 20) training and test data sets. Except for banana, ringnorm, and thyroid data sets, we normalize the input ranges into $[0, 1]$.

[1] http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm

Data sets shown in Table 2 are multiclass data sets: iris (Fisher, 1936; Bezdek et al., 1999), numeral (Takenaga et al., 1991), blood cell (Hashizume, Motoike, and Yabe, 1988), thyroid (Weiss and Kapouleas, 1999) [2], hiragana (Abe, 2001), and MNIST (LeCun et al., 1998) [3] data sets. Each data set consists of one training data set and one test data set. We normalize the input ranges into $[0, 1]$ for the blood cell data set. Because the MNIST data set with ten classes is very large, we use only classes 7 and 9, which have the largest classification error among the class pairs.

In approximating hyperspheres, we use the same kernels with the classifiers and set $C_j = 1$ so that all the data are inside of the hyperspheres.

For two-class problems, we randomly generate one incremental training data set for each of 100 (or 20) training data sets, dividing the training data set into the subsets whose size is about 5% of the total number. In the incremental training, we add one of the subsets to the training set at each step. Likewise for multiclass problems, for each training data set we randomly generate 100 incremental training data sets. For instance, the number of iris training data is 75. Thus, truncating 75×0.05 the number of elements for each subset is 3 and the number of subsets is 25.

In incremental training, each subset is added to the classifier one at a time and the classifier is trained by the conventional and proposed methods. Thus, for each classification problem, we train classifiers 100 (or 20) times and calculate the statistics.

The recognition rates of the test data, after incremental training is finished, are compared with those by batch training using all the training data. To evaluate the efficiency of data deletion, we evaluate the deletion ratio that is defined by the number of the deleted data divided by the number of training data. For batch training, the deletion ratio is the ratio of the deletable training data (i.e., training data minus support vectors) against training data. Thus assuming that all the support vectors are retained, the deletion ratio for the batch training is the upper bound for the deletion ratio for incremental training.

We also measure the computing time for batch training and incremental training using a personal computer (Xeon 2.8GHz, 1GB Memory) with the Linux operating system. We use the primal-dual interior-point method combined with the decomposition technique to train support vector machines.

[2] ftp://ftp.ics.uci.edu/pub/machine-learning-databases/
[3] http://yann.lecun.com/exdb/mnist/

5.2 Recognition Performance during Incremental Training

Figs. 10 and 11 show the recognition rates of the iris training and test data sets during incremental training, respectively. In calculating the recognition rate of the training data we use all the training data. Thus, some data are not added yet to the classifier. We use polynomial kernels with $d = 3$ and set the regularization parameter $C = 10000$. Here, we determine the value of d by cross validation, and use a large value of C to show that incremental training works even for the worst situation where there is a risk of overfitting. For the conventional training method we set $\beta = 1$ and for the proposed method $\rho = 0.5, \theta = 0$. In batch training, at each step, we train the support vector machine with the initial data set and all the added data sets.

The recognition rates of both the training and test data sets by the incremental training are non-decreasing at almost all steps. By the batch training, however, the variation of the recognition rate is large. This is because in batch training the support vector machine is retrained at every incremental step. Thus, overfitting to a local data set may results in the variation in the recognition rate. But in the conventional method and the proposed method, the support vector machine is not retrained if the data all satisfy $y(\mathbf{x})f(\mathbf{x}) \geq 1$.

5.3 Parameter Setting

In applying the proposed method to classification problems, we need to set the values to ρ and θ. But if we need to optimize the values for each problem, the applicability of the method is diminished. To solve this, we determine the values using one data set, namely, the twonorm data set, and use them throughout the study. As will be shown later, this method works very well. On the contrary, for the conventional method, we cannot set a value of β that works well for most of the data set.

Table 3 shows the results of the proposed method for the twonorm data set using polynomial kernels with $d = 4$ and the regularization parameter $C = 10000$ for different values of ρ and θ. In the table, "Test," "Trn," and "Del" denote the recognition rate of the test data, that of the training data, and deletion ratio after incremental training is finished, respectively. In the table, for example, 95.78±0.87 means that the average recognition rate is 95.78% with the standard deviation of 0.87%.

In batch training, the recognition rate of the test data is 96.35±0.45% and that of the training data is 99.93±0.13% under the same conditions. Comparing the result of the batch training with Table 3, the parameters values of $\rho = 0.5, \theta = 0$ are at the boundary where the recognition rates decrease. The

recognition rates are stable as ρ becomes smaller and θ becomes larger. When the parameters are $\rho = 0.5$ and $\theta = 0$, the recognition rates of the proposed method are equal to those of the batch training. And 51% of the data are deleted. The results for other data sets are similar to this result. Therefore we can consider the parameters values of $\rho = 0.5, \theta = 0$ as the optimal parameters to keep the recognition rates high while deleting sufficient training data. Therefore, in the following study we set $\rho = 0.5$ and $\theta = 0$.

5.4 Comparison between Conventional and Proposed Methods

We compare the conventional and proposed methods with the optimized kernels and C and with those and $C = 10000$. Here, we use a large value of C so that comparison of incremental training with the optimum and non-optimum values of C becomes clear. To optimize kernels and C, we use 5-fold cross validation. For polynomial kernels with $d = [2, 3, 4]$, and RBF kernels with $\gamma = [0.1, 1, 10]$, we perform 5-fold cross validation for the regularization parameter $C = [1, 10, 50, 100, 500, 1000, 2000, 3000, 5000, 8000, 10000, 50000, 100000]$, and select the kernel, its parameter, and the regularization parameter with the highest average recognition rate in the batch training.

Table 4 shows the parameters for the two-class problems. In the table, the column "Kernel" lists the kernels determined by cross validation. For example, $\gamma 1$ and $d2$ denote RBF kernels with $\gamma = 1$ and polynomial kernels with $d = 2$, respectively. The same kernels are used for two cases: (1) optimal C and (2) $C = 10000$. The parameters for case (1) are listed in the columns "Batch1," "Hplane1," and "Sphere1," and those for case (2), "Batch2," "Hplane2," and "Sphere2." For each case, the same regularization parameter value is used for batch training, the conventional and proposed methods.

For the proposed method, we use $\rho = 0.5$ and $\theta = 0$ for all cases as listed in "Sphere1" and "Sphere2." As will be shown immediately, for the conventional method we cannot select a value of β common to all cases because the optimal value changes as the data set changes. Thus not to favor the proposed method, we set the optimal value to β. Namely, we select the value of β so that the comparable recognition rate with that of batch training is obtained.

Table 5 shows the results for the two-class problems. In each problem the maximum deletion ratios for Hplane1 and Sphere1 and for Hplane2 and Sphere2 are shown in boldfaces. The recognition rates for Batch1 and Sphere1, and Batch2 and Sphere2 are almost the same using the user-defined parameters ($\rho = 0.5, \theta = 0$). And about 30% to 60% of the training data are deleted. Since the recognition rates of the proposed method are almost the same with those of batch training, we can consider that almost all support vectors remain af-

ter incremental training. Consequently we can conclude that we delete many unnecessary data without deleting candidates for support vectors.

The deletion ratios of Sphere1 are often larger than those of Hplane1 and the deletion ratios of Sphere2 are often larger than those of Hplane2.

The optimal value of β for the conventional method changes as the data set changes. Thus, it is difficult to set the optimal value of β in advance. But for the proposed method, the choice of $\rho = 0.5, \theta = 0$ is almost always good.

To show how the performance changes for the change of β, we calculate the averages and the standard deviations for $\beta = [0.01, 0.1, 0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 30, 50]$. Table 6 shows the results of the conventional method for the two-class data sets.

For the ringnorm to the waveform data sets except the twonorm data set, the standard deviations of the recognition rates for training and test data sets are small. Therefore, changing β has little influence on the recognition rates. But for the banana to the image data sets and the twonorm data set, the standard deviations of the recognition rates for training and test data sets are large. Therefore, the effect of changing β is large. In addition, the standard deviations of the deletion ratios for all the data sets are large. Therefore the effect of changing β is large for deleting data and it is difficult to set the optimal value of β for these data sets.

Table 7 shows the computing time for two-class problems. We divide the total computing time "Total_t" into three parts: incremental training time "Trn_t," deletion time "Del_t," and total hypersphere generation time "Sph_t." The deletion time denotes the time to check if the data can be deleted in incremental training. For Batch1, at each incremental training step, batch training using all the data obtained so far is performed and $\text{Total}_\text{t} = \text{Trn}_\text{t}$. For the conventional method, $\text{Total}_\text{t} = \text{Trn}_\text{t} + \text{Del}_\text{t}$. And for the proposed method, $\text{Total}_\text{t} = \text{Trn}_\text{t} + \text{Del}_\text{t} + \text{Sph}_\text{t}$.

We only show the results for five data sets. The tendency of the results for the remaining data sets is the same. From the table, the incremental training time, Trn_t, of the proposed method is shorter than that of batch training and comparable with that of the conventional method. The deletion time, Del_t, of the proposed method is longer than that of the proposed method because the deletion check is complicated. This tendency becomes prominent if the number of deleted data is large. In the proposed method, hypersphere generation time, Sph_t, is negligible except for the splice data set. The total computing time, Total_t, except for the image data, the ratio of the deletion time in the total training time is small. For the image data set, because of the long deletion time, the total computing time is longer than that of batch training.

Table 8 shows the parameters for the multiclass problems. Here, we also consider two cases: (1) optimal C and (2) $C = 10000$.

As before, for the proposed method, we use $\rho = 0.5$ and $\theta = 0$. For the conventional method the value of β is selected so that the comparable recognition rate with that of batch training is obtained.

Table 9 shows the results for the multiclass problems. For the iris data set for Hplane1 and Sphere1, we cannot delete data. Except for these cases, the proposed method realizes (almost) the same recognition rates with those of batch training while deleting 40% to 70% of the training data.

As before, the optimal value of β for the conventional method changes as the data set changes. Thus, it is difficult to set the optimal value of β in advance.

Table 10 shows the results of the computing time for the multiclass problems excluding the iris and the numeral data sets. The total computing time, Total_t of the proposed method is usually shorter than that of batch training and comparable with that of the conventional method. For the MNIST data set, on average the total computing time of the proposed method is shorter than that of batch training. But the large variance indicates that, in some cases the total computing time of the proposed method is longer. The training time, Trn_t, of the proposed method is usually much shorter than that of batch training. As before, the deletion time of the proposed method is longer than that of the conventional method and the hypersphere generation does not take much time.

6 Discussions

According to the simulation results, the proposed method with $\rho = 0.5$ and $\theta = 0$ is shown to have generalization ability comparable to batch training while deleting 30% to 60% of the training data. This is especially favorable since it is difficult to tune parameters in incremental training.

But since cross validation is not possible for incremental training, optimal selection of kernels and the value of C is difficult. To solve this problem, we need to optimize these kernels and parameters during incremental training. But since there is no known way, we leave this as future work.

7 Conclusions

In this paper, to reduce the memory consumption by deleting unnecessary data, we discussed incremental training of support vector machines using hyperspheres.

Namely, we generate the minimum-volume hypersphere that includes the training data of each class and a concentric hypersphere with a smaller radius. Next, we generate the hypercone whose vertex is at the center of the hypersphere and which opens in the opposite direction of the separating hyperplane. We delete the data that exist inside of the small hypersphere or inside of the hypercone and keep the remaining data as candidates for support vectors.

Using the two-class and multiclass benchmark data sets, we showed that we can delete the data significantly while keeping the generalization ability of the incremental support vector machine comparable with that of the batch support vector machine by the fixed user-defined parameters.

References

S. Abe, 2001. Pattern Classification, Neuro-fuzzy Methods and Their Comparison. Springer-Verlag, London.

T. G. Bakiri and G. Dietterich, 1995. Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research*, Vol. 2, pp. 263–286.

J. C. Bezdek et al., 1999. Will the Real Iris Data Please Stand up? *IEEE Transactions on Fuzzy Systems*, Vol. 7, No. 3, pp. 368–369.

G. Cauwenberghs and T. Poggio, 2000. Incremental and Decremental Support Vector Machine Learning. In *T. K. Leen, T. G. Dietterich, and V. Tresp, Eds., Advances in Neural Information Proceeding Systems 13*, pp. 409–415, MIT Press, Cambridge, MA.

C. Domeniconi and D. Gunopulos, 2001. Incremental Support Vector Machine Construction. In *Proc. First IEEE International Conference on Data Mining*, pp. 589–592.

R. A. Fisher, 1936. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, Vol. 7, pp. 179–188.

A. Hashizume, J. Motoike, and R. Yabe, 1988. Fully Automated Blood Cell Differential System and Its Application. In *Proc. IUPAC 3rd International Congress on Automation and New Technology in the Clinical Laboratory*, pp. 297–302, Kobe, Japan.

U. H.-G. Kreßel, 1999. Pairwise Classification and Support Vector Machines. In *B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds., Advances in Kernel*

Methods: Support Vector Learning, pp. 255–268, MIT Press, Cambridge, MA.

Y. LeCun et al., 1998. Gradient-based learning applied to document recognition. *Proc. the IEEE*, Vol. 86, No. 11, pp. 2278–2324.

P. Mitra, C. A. Murthy, and S. K. Pal, 2000. Data Condensation in Large Databases by Incremental Learning with Support Vector Machines. In *Proc. International Conference on Pattern Recognition*, pp. 2708–2711.

J. P. Pedroso and N. Murata, 2000. Optimisation on Support Vector Machines. In *Proc. IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN 2000)*, volume 6, pages 399–404.

L. Ralaivola and F. d'Alché-Buc, 2001. Incremental Support Vector Machine Learning: A Local Approach. In *Proc. International Conference on Artificial Neural Networks*, pages 322–330. Vienna, Austria.

A. Shilton, M. Palaniswami, D. Ralph, and A. C. Tsoi, 2005. Incremental Trainings of Support Vector Machines. *IEEE Transactions on Neural Networks*, Vol. 16, No. 1, pp. 114–131.

H. Takenaga et al., 1991. Input Layer Optimization of Neural Networks by Sensitivity Analysis and Its Application to Recognition of Numerals. *Electrical Engineering in Japan*, Vol. 111, No. 4, pp. 130–138.

D. M. J. Tax and R. P. W. Duin, 2001. Outliers and Data Descriptions. In *Proc. the Seventh Annual Conference of the Advanced School for Computing and Imaging*, pp. 234–241.

V. Vapnik, 1995. The Nature of Statistical Learning Theory. Springer-Verlag, London, UK.

V. Vapnik, 1998. Statistical Learning Theory. John Wiley & Sons, New York, N.Y.

S. M. Weiss and I. Kapouleas, 1999. An Empirical Comparison of Pattern Recognition, Neural Nets, and Machine Learning Classification Methods. In *Proc. IJCAI-99, Workshop ML3*, pp. 55–60, 1999.

R. Xiao, J. Wang, and F. Zhang, 2000. An Approach to Incremental SVM Learning Algorithm. In *Proc. Twelfth IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2000)*, pages 268–273.

Figure captions:

Fig. 1 Selection of support vector candidates
Fig. 2 Before adding data
Fig. 3 After adding data
Fig. 4 After deleting data and retraining
Fig. 5 Batch Training
Fig. 6 Estimation of the separating hyperplane after incremental training
Fig. 7 Estimation of candidates for support vectors
Fig. 8 Deletion of the data using the hyperspheres
Fig. 9 Judging whether the data are inside of the hypercone or not
Fig. 10 Recognition rate of the iris training data during incremental training

Fig. 11 Recognition rate of the iris test data during incremental training

Table captions:

Table 1 Two-class benchmark data specification
Table 2 Multiclass benchmark data specification
Table 3 Effect of parameters on performance for the twonorm data set (%)
Table 4 Parameters for two-class problems
Table 5 Comparison between conventional and proposed methods for two-class problems (%)
Table 6 Effect of a parameter on performance by the conventional method (%)
Table 7 Training time and deleting time for two-class problems (s)
Table 8 Parameters for multiclass problems
Table 9 Comparison between conventional and proposed methods for multiclass problems (%)
Table 10 Learning time and deleting time for multiclass problems (%)

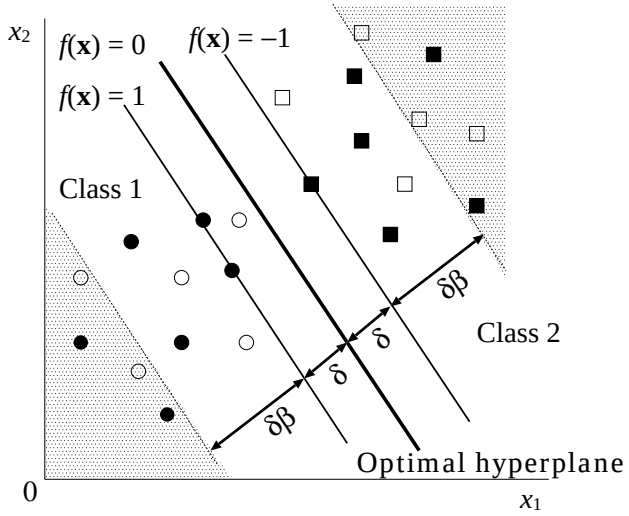

Fig. 1. Selection of support vector candidates

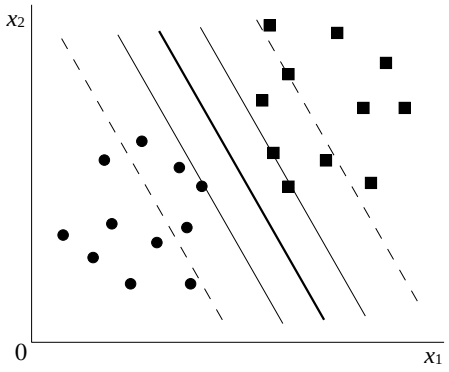

Fig. 2. Before adding data

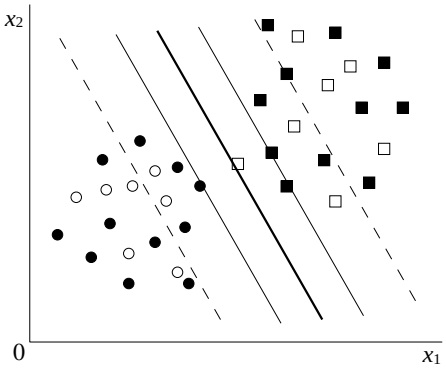

Fig. 3. After adding data

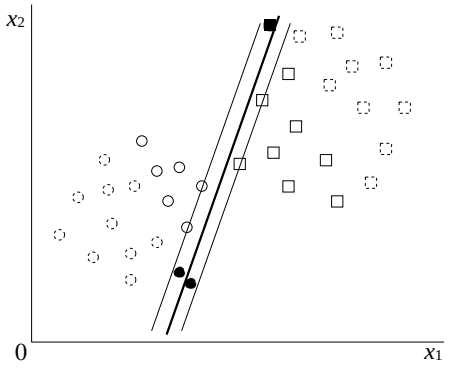

Fig. 4. After deleting data and retraining

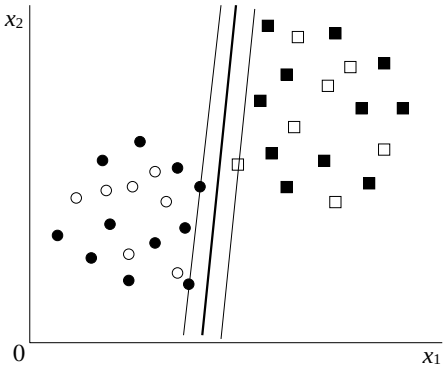

Fig. 5. Batch training

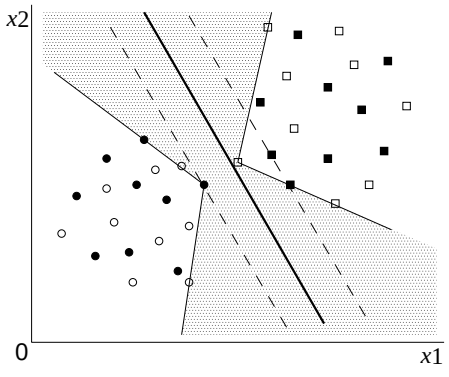

Fig. 6. Estimation of the separating hyperplane after incremental training

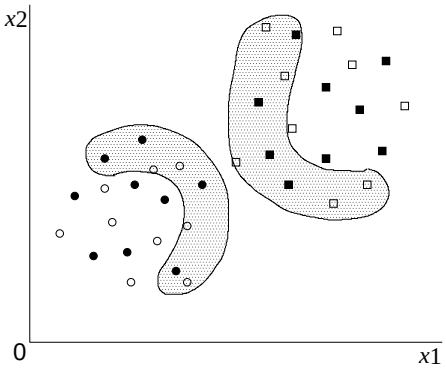

Fig. 7. Estimation of candidates for support vectors

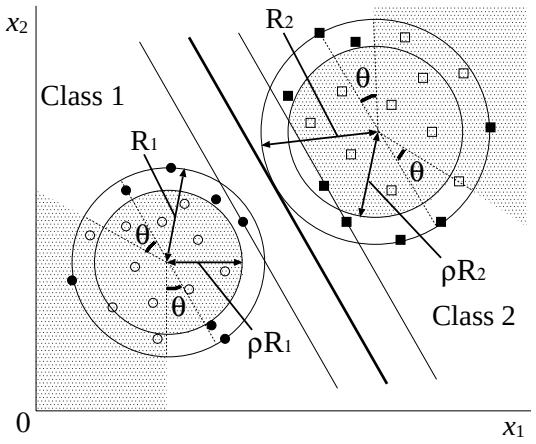

Fig. 8. Deletion of the data using the hyperspheres

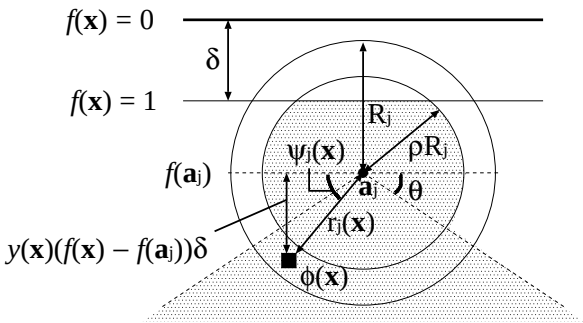

Fig. 9. Judging whether the data are inside of the hypercone or not

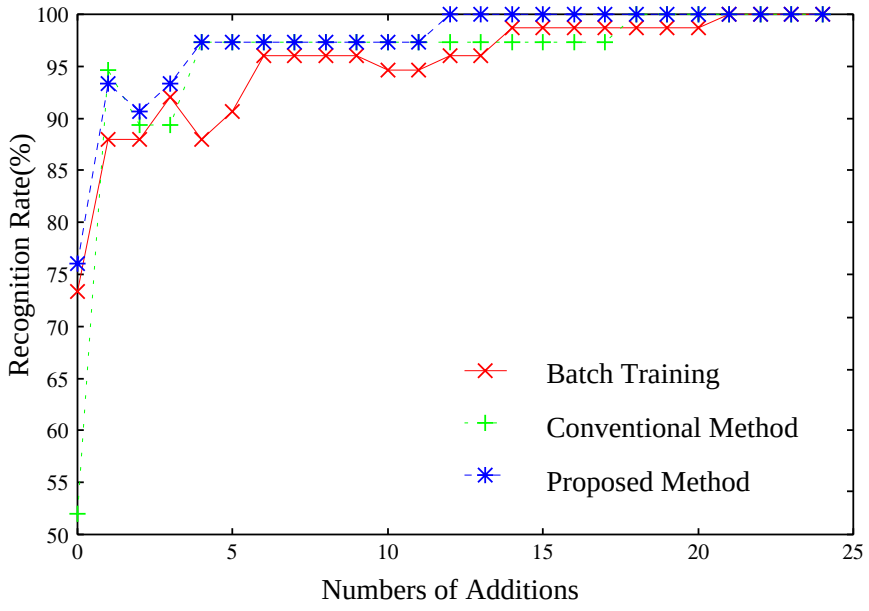

Fig. 10. Recognition rate of the iris training data during incremental training

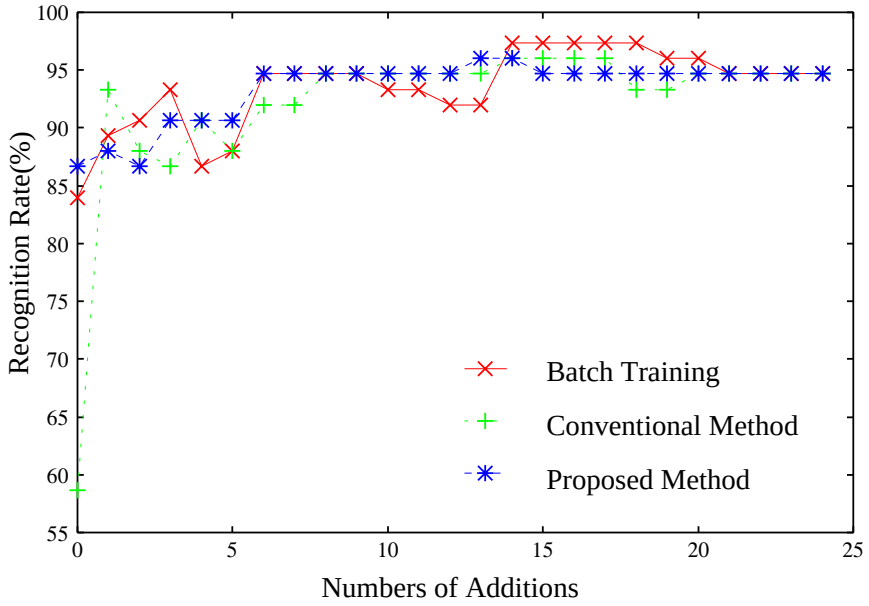

Fig. 11. Recognition rate of the iris test data during incremental training

Table 1
Two-class benchmark data specification

	Trn	Test	Inputs	Classes
Banana	400	4900	2	2
B. cancer	200	77	9	2
Diabetes	468	300	8	2
German	700	300	20	2
Heart	170	100	13	2
Image	1300	1010	18	2
Ringnorm	400	7000	20	2
F. solar	666	400	9	2
Splice	1000	2175	60	2
Thyroid	140	75	5	2
Twonorm	400	7000	20	2
Waveform	400	4600	5	2

Table 2
Multiclass benchmark data specification

	Trn	Test	Inputs	Classes
Iris	75	75	4	3
Numeral	810	820	12	10
Blood cell	3097	3100	13	13
Thyroid (M)	3772	3428	21	3
Hiragana-50	4610	4610	50	39
Hiragana-13	8375	8356	13	38
Hiragana-105	8375	8356	105	38
MNIST	12214	2037	784	2

Table 3
Effect of parameters on performance for the twonorm data set (%)

θ	Term	$\rho = 0.3$	$\rho = 0.4$	$\boldsymbol{\rho = 0.5}$	$\rho = 0.6$	$\rho = 0.7$
	Test	95.78±0.87	95.78±0.87	95.78±0.87	95.76±0.87	95.8±0.81
−30	Trn	99.42±0.61	99.42±0.61	99.42±0.61	99.41±0.61	99.41±0.59
	Del	87.59±2.17	87.59±2.17	87.59±2.17	87.66±1.95	87.77±1.89
	Test	96.00±0.65	96.00±0.66	96.00±0.65	95.98±0.66	95.93±0.78
−20	Trn	99.66±0.42	99.66±0.42	99.66±0.42	99.61±0.47	99.55±0.75
	Del	84.26±4.24	84.25±4.24	84.24±4.24	84.51±4.13	85.06±3.75
	Test	96.19±0.52	96.19±0.52	96.19±0.51	96.17±0.54	96.14±0.54
−10	Trn	99.78±0.32	99.78±0.32	99.79±0.31	99.76±0.30	99.78±0.32
	Del	70.01±6.86	70.01±6.86	69.99±7.00	70.62±6.59	74.33±5.70
	Test	96.35±0.45	96.35±0.45	**96.35±0.45**	96.34±0.44	96.31±0.48
0	Trn	99.93±0.13	99.93±0.13	**99.93±0.13**	99.93±0.13	99.92±0.14
	Del	50.81±4.25	50.81±4.27	**51.00±4.36**	53.17±4.40	61.26±4.29
	Test	96.35±0.45	96.35±0.45	96.35±0.45	96.34±0.44	96.33±0.44
10	Trn	99.93±0.13	99.93±0.13	99.93±0.13	99.93±0.13	99.93±0.13
	Del	29.80±5.69	29.81±5.70	30.30±5.58	33.75±4.89	45.72±4.81
	Test	96.35±0.45	96.35±0.45	96.35±0.45	96.35±0.44	96.32±0.45
20	Trn	99.93±0.13	99.93±0.13	99.93±0.13	99.93±0.13	99.92±0.14
	Del	12.07±5.17	12.07±5.17	12.18±4.90	16.05±5.28	31.84±4.89

Table 4
Parameters for two-class problems

Data	Kernel	Bacth1	Hplane1	Sphere1		Batch2	Hplane2	Sphere2	
		C	β	ρ	θ	C	β	ρ	θ
Banana	$\gamma 1$	10	2	0.5	0	10000	10	0.5	0
B. cancer	$\gamma 1$	1	0.01	0.5	0	10000	4	0.5	0
Diabetes	$d2$	50	0.5	0.5	0	10000	1	0.5	0
German	$\gamma 1$	10	0.1	0.5	0	10000	2	0.5	0
Heart	$\gamma 1$	50	0.5	0.5	0	10000	4	0.5	0
Image	$\gamma 1$	1000	2	0.5	0	10000	5	0.5	0
Ringnorm	$\gamma 0.1$	1	0.1	0.5	0	10000	0.1	0.5	0
F. solar	$d2$	10	0.5	0.5	0	10000	2	0.5	0
Splice	$\gamma 10$	1	0.1	0.5	0	10000	0.1	0.5	0
Thyroid	$d2$	1	1	0.5	0	10000	3	0.5	0
Twonorm	$d4$	50	2	0.5	0	10000	2	0.5	0
Waveform	$\gamma 1$	1	0.1	0.5	0	10000	2	0.5	0

Table 5
Comparison between conventional and proposed methods for two-class problems (%)

Data	Term	Bacth1	Hplane1	Sphere1	Batch2	Hplane2	Sphere2
Banana	Test	89.31±0.53	89.31±0.53	89.31±0.53	87.05±0.99	87.04±0.99	87.05±0.99
	Trn	91.95±1.30	91.93±1.30	91.93±1.30	94.67±1.33	94.64±1.33	94.67±1.33
	Del	73.58±2.30	64.29±3.69	**69.77±2.68**	76.91±3.05	47.11±7.28	**53.21±6.59**
B. cancer	Test	73.25±4.53	73.25±4.53	73.25±4.53	65.40±4.65	65.40±4.57	65.40±4.65
	Trn	82.80±1.72	82.80±1.72	82.80±1.72	98.33±0.62	98.14±0.75	98.33±0.62
	Del	34.59±2.73	**0.07±0.27**	0.06±0.22	50.06±3.35	26.31±8.36	**37.42±5.52**
Diabetes	Test	76.46±1.85	76.46±1.85	76.46±1.85	74.58±1.72	74.52±1.84	74.58±1.72
	Trn	78.48±1.22	78.48±1.22	78.48±1.22	81.82±1.23	81.70±1.14	81.82±1.23
	Del	45.53±1.73	15.68±8.32	**27.83±13.44**	49.28±2.32	36.60±5.14	**44.13±8.07**
German	Test	76.63±2.14	76.70±2.26	76.63±2.14	69.88±2.54	69.82±2.54	69.88±2.54
	Trn	81.14±1.27	80.94±1.41	81.14±1.27	99.04±1.10	98.81±1.13	99.04±1.10
	Del	43.83±1.52	33.84±5.65	**35.49±3.72**	49.92±2.44	44.94±3.34	**52.81±5.65**
Heart	Test	83.68±3.39	83.68±3.39	83.68±3.39	79.68±3.43	79.75±3.49	79.75±3.49
	Trn	85.95±1.92	85.95±1.92	85.95±1.92	93.06±1.65	92.96±1.62	93.06±1.65
	Del	56.33±3.43	24.89±21.91	**32.36±27.62**	58.63±4.07	39.31±7.91	**47.07±6.44**
Image	Test	97.14±0.48	97.12±0.47	97.13±0.48	96.87±0.38	96.78±0.45	96.86±0.38
	Trn	98.60±0.18	98.60±0.17	98.60±0.18	99.51±0.17	99.42±0.21	99.50±0.17
	Del	88.31±0.71	60.22±3.76	**61.66±4.67**	91.22±0.58	59.11±3.79	**61.64±6.83**
Ringnorm	Test	98.41±0.10	98.41±0.10	98.41±0.10	98.34±0.12	98.34±0.12	98.34±0.12
	Trn	99.91±0.15	99.91±0.15	99.91±0.15	100.0±0.00	100.0±0.00	100.0±0.00
	Del	61.48±2.29	**45.59±16.27**	31.03±11.14	62.14±2.23	**51.86±2.93**	34.94±2.51
F. solar	Test	68.29±1.85	68.29±1.85	68.29±1.85	66.30±2.06	66.25±2.02	66.25±2.02
	Trn	68.19±1.26	68.19±1.26	68.19±1.26	69.22±1.24	69.11±1.24	69.22±1.24
	Del	18.93±1.93	4.51±3.46	**11.45±8.63**	18.15±2.12	8.61±3.48	**15.33±5.58**
Splice	Test	88.66±0.71	88.66±0.72	88.66±0.71	89.23±0.71	89.20±0.71	89.23±0.71
	Trn	99.09±0.24	99.09±0.24	99.09±0.24	99.98±0.04	99.98±0.04	99.98±0.04
	Del	26.49±1.29	11.85±10.74	**13.17±11.95**	25.77±1.40	20.26±1.32	**23.51±1.65**
Thyroid	Test	96.31±1.90	96.31±1.90	96.31±1.90	95.51±2.23	95.51±2.23	95.51±2.23
	Trn	99.34±0.49	99.34±0.49	99.34±0.49	100.0±0.00	100.0±0.00	100.0±0.00
	Del	88.92±1.33	**66.49±11.29**	60.89±6.68	91.78±1.02	55.66±19.45	**61.19±7.90**
Twonorm	Test	97.57±0.12	97.57±0.12	97.57±0.12	96.35±0.45	96.35±0.45	96.35±0.45
	Trn	98.24±0.55	98.24±0.55	98.24±0.55	99.93±0.13	99.93±0.13	99.93±0.13
	Del	79.09±1.56	**57.47±6.42**	51.21±6.82	90.53±1.94	**72.24±6.73**	51.00±4.36
Waveform	Test	90.00±0.45	90.00±0.45	90.00±0.45	87.96±0.80	87.96±0.80	87.96±0.80
	Trn	93.51±1.37	93.51±1.37	93.51±1.37	100.0±0.00	100.0±0.00	100.0±0.00
	Del	61.63±2.24	36.79±28.02	**38.16±25.66**	72.99±3.06	45.55±4.08	**61.77±3.22**

Table 6
Effect of a parameter on performance by the conventional method (%)

Data	Kernel	C	Trn	Test	Del
Banana	$\gamma 1$	10000	88.80±5.43	82.18±4.30	59.03±23.95
B. cancer	$\gamma 1$	10000	94.13±4.92	63.83±1.67	25.80±18.61
Diabetes	$d2$	10000	80.91±1.58	74.11±0.96	15.44±21.73
German	$\gamma 1$	10000	94.11±6.70	67.84±2.68	27.55±27.21
Heart	$\gamma 1$	10000	91.25±2.64	78.57±1.54	21.06±21.62
Image	$\gamma 1$	10000	98.18±1.90	95.68±1.65	55.87±31.24
Ringnorm	$\gamma 0.1$	10000	100.00±0.00	98.34±0.00	9.47±19.40
F. solar	$d2$	10000	68.74±0.76	65.99±0.61	6.59±8.02
Splice	$\gamma 10$	10000	99.98±0.00	89.23±0.01	3.43±7.96
Thyroid	$d2$	10000	99.47±0.69	95.18±0.39	44.05±32.34
Twonorm	$d4$	10000	99.15±1.92	95.65±1.71	38.47±35.31
Waveform	$\gamma 1$	10000	99.68±0.67	87.78±0.38	27.10±30.40

Table 7
Training time and deleting time for two-class problems (s)

Data		Bacth1	Hplane1	Sphere1
Diabetes	Total_t	24.15±3.56	19.72±8.43	**14.52±5.79**
	Trn_t	24.15±3.56	19.35±8.26	**11.85±4.67**
	Del_t	—	**0.37±0.18**	2.35±1.34
	Sph_t	—	—	0.32±0.05
German	Total_t	90.11±15.60	**58.54±13.30**	62.65±10.48
	Trn_t	90.11±15.60	57.37±13.13	**55.27±10.44**
	Del_t	—	**1.17±0.21**	6.58±1.20
	Sph_t	—	—	0.80±0.08
Image	Total_t	27.00±3.90	**14.48±3.02**	30.00±3.45
	Trn_t	27.00±3.90	13.84±2.98	**11.04±1.42**
	Del_t	—	**0.64±0.07**	18.25±2.45
	Sph_t	—	—	0.71±0.07
F. solar	Total_t	108.19±11.45	75.14±41.38	**73.79±41.20**
	Trn_t	108.19±11.45	74.18±40.73	**65.66±35.39**
	Del_t	—	**0.96±0.66**	7.54±7.18
	Sph_t	—	—	0.59±0.08
Splice	Total_t	514.33±27.07	**314.28±161.69**	314.40±146.27
	Trn_t	514.33±27.07	309.11±158.15	**256.04±123.84**
	Del_t	—	**5.17±3.55**	32.18±27.92
	Sph_t	—	—	26.18±3.90

Table 8
Parameters for multiclass problems

Data	Kernel	Bacth1	Hplane1	Sphere1		Batch2	Hplane2	Sphere2	
		C	β	ρ	θ	C	β	ρ	θ
Iris	$d3$	10	0.01	0.5	0	10000	1	0.5	0
Numeral	$d2$	10	0.5	0.5	0	10000	0.5	0.5	0
Blood	$\gamma 10$	500	2	0.5	0	10000	3	0.5	0
Thyroid(M)	$d3$	50000	3	0.5	0	10000	2	0.5	0
Hiragana-50	$\gamma 0.1$	100000	0.01	0.5	0	10000	0.1	0.5	0
Hiragana-13	$\gamma 10$	100000	0.01	0.5	0	10000	0.01	0.5	0
Hiragana-105	$\gamma 0.1$	100000	0.01	0.5	0	10000	0.01	0.5	0
MNIST	$d2$	1000	1	0.5	0	10000	2	0.5	0

Table 9
Comparison between conventional and proposed methods for multiclass problems (%)

Data	Term	Bacth1	Hplane1	Sphere1	Batch2	Hplane2	Sphere2
Iris	Test	97.33±0.00	97.33±0.00	97.33±0.00	94.67±0.00	94.67±0.00	94.67±0.00
	Trn	98.67±0.00	98.67±0.00	98.67±0.00	100.0±0.00	100.0±0.00	100.0±0.00
	Del	57.33±0.00	0.00±0.00	0.0±0.00	84.00±0.00	39.07±1.80	**55.47±2.08**
Numeral	Test	99.63±0.00	99.63±0.00	99.63±0.00	99.39±0.00	99.39±0.00	99.39±0.00
	Trn	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00
	Del	79.44±0.15	21.94±1.27	**34.31±1.61**	85.16±0.46	23.42±1.38	**53.82±2.07**
Blood	Test	93.59±0.02	93.59±0.05	93.59±0.04	93.59±0.02	93.57±0.15	93.59±0.04
	Trn	97.92±0.02	97.91±0.02	97.92±0.07	99.26±0.00	99.22±0.04	99.26±0.09
	Del	79.09±0.12	**31.37±0.53**	30.20±2.00	80.80±0.08	55.60±0.50	**69.69±1.97**
Thyroid(M)	Test	97.55±0.02	97.53±0.07	97.55±0.05	97.52±0.01	97.47±0.05	97.51±0.10
	Trn	99.44±0.02	99.40±0.04	99.43±0.01	99.29±0.02	99.27±0.05	99.28±0.11
	Del	89.82±0.28	38.73±0.66	**53.97±19.83**	88.93±0.50	35.52±1.25	**50.46±19.74**
Hiragana-50	Test	99.35±0.00	99.34±0.01	99.35±0.02	99.35±0.00	99.34±0.02	99.35±0.02
	Trn	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00
	Del	72.05±0.00	**67.38±0.09**	58.09±0.40	72.02±0.20	50.89±0.37	**57.80±0.83**
Hiragana-13	Test	99.87±0.00	99.86±0.01	99.87±0.00	99.87±0.00	99.86±0.01	99.87±0.00
	Trn	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00
	Del	92.67±0.45	57.01±0.12	**66.79±0.14**	93.03±0.21	56.92±0.34	**66.81±0.07**
Hiragana-105	Test	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00
	Trn	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00
	Del	81.22±0.05	**76.22±0.11**	73.37±0.15	81.17±0.08	**76.33±0.19**	73.50±0.37
MNIST	Test	92.83±0.83	92.83±0.81	92.82±1.37	91.05±0.25	90.85±1.32	91.05±0.37
	Trn	94.04±0.69	94.04±0.44	94.03±1.15	92.54±0.03	92.34±1.42	92.54±0.36
	Del	93.50±0.09	39.01±1.71	**58.60±0.30**	93.26±0.02	58.81±2.96	**62.33±0.62**

Table 10
Learning time and deleting time for multiclass problems (s)

Data		Bacth1	Hplane1	Sphere1
Blood	Total_t	138.15±2.34	**98.01±2.16**	116.65±2.57
	Trn_t	138.15±2.34	91.61±2.07	**87.16±2.31**
	Del_t	—	**6.40±0.09**	27.33±0.33
	Sph_t	—	—	2.16±0.07
Thyroid(M)	Total_t	124.46±3.51	**74.19±1.86**	131.55±26.91
	Trn_t	124.46±3.51	**72.13±1.82**	76.67±12.00
	Del_t	—	**2.06±0.08**	53.56±14.93
	Sph_t	—	—	1.32±0.04
Hiragana-50	Total_t	855.93±6.05	**278.86±3.71**	370.36±4.85
	Trn_t	855.93±6.05	**241.41±3.33**	313.69±3.96
	Del_t	—	**37.45±0.40**	53.61±0.84
	Sph_t	—	—	3.07±0.12
Hiragana-13	Total_t	4495.12±20.34	1980.50±49.79	**1588.12±11.19**
	Trn_t	4495.12±20.34	1887.94±48.23	**1455.31±10.17**
	Del_t	—	**92.56±1.66**	121.49±1.10
	Sph_t	—	—	11.32±0.15
Hiragana-105	Total_t	4387.78±17.40	**1099.37±14.55**	1358.60±23.59
	Trn_t	4387.78±17.40	**916.54±12.72**	1103.29±19.99
	Del_t	—	**182.83±1.88**	244.29±3.57
	Sph_t	—	—	11.02±0.11
MNIST	Total_t	8399.60±828.14	**6168.05±1335.35**	7755.64±2867.97
	Trn_t	8399.60±828.14	5999.85±1344.04	**4541.91±4111.39**
	Del_t	—	**168.21±8.73**	3185.4±2455.8
	Sph_t	—	—	28.33±0.51