Incremental Training of Support Vector Machines Using Truncated Hypercones

Shinya Katagiri and Shigeo Abe

Graduate School of Science and Technology
Kobe University
Rokkodai, Nada, Kobe, Japan
abe@eedept.kobe-u.ac.jp
http://www2.eedept.kobe-u.ac.jp/~abe

Abstract. We discuss incremental training of support vector machines in which we approximate the regions, where support vector candidates exist, by truncated hypercones. We generate the truncated surface with the center being the center of unbounded support vectors and with the radius being the maximum distance from the center to support vectors. We determine the hypercone surface so that it includes a datum, which is far away from the separating hyperplane. Then to cope with non-separable cases, we shift the truncated hypercone along the rotating axis in parallel in the opposite direction of the separating hyperplane. We delete the data that are in the truncated hypercone and keep the remaining data as support vector candidates. In computer experiments, we show that we can delete many data without deteriorating the generalization ability.

1 Introduction

The high generalization ability of support vector machines (SVMs) [1,2] lies in mapping of the input space to a high dimensional feature space, maximizing margins of separating hyperplanes in the feature space, and use of proper kernels to specific applications. Training by solving a quadratic programming problem leads to the global optimum solution. But since we need to solve a problem with the variables equal to the number of training data, training becomes slow for a large sized problem. In addition, in an incremental training environment, where training data are incrementally obtained, efficient incremental training methods are required. In SVMs, only support vectors, which are near class boundaries and which determine the decision functions, are required for training. Thus, if we can detect support vectors or support vector candidates in future incremental training, we can alleviate slow training by deleting unnecessary data before training.

In [3,4], training data other than support vectors are deleted at the incremental training step. However, this method may delete support vector candidates and thus may lead to degradation of generalization ability. Therefore, to maintain the generalization ability comparable to that of batch training, we need

F. Schwenker and S. Marinai (Eds.): ANNPR 2006, LNAI 4087, pp. 153–164, 2006.

to retain support vector candidates. Under the assumption that the separating hyperplane after incremental training does not change much, in [5] support vector candidates are selected from the data that are near the separating hyperplane. But if the separating hyperplane rotates after retraining, this method may fail in keeping support vector candidates. To cope with the rotation of separating hyperplanes, in [6,7] class regions are approximated by hyperspheres using one-class SVMs [8] and data near hyperspheres are kept as support vector candidates.

In this paper, we propose an incremental training method that is robust for rotation of separating hyperplanes, approximating the regions for support vector candidates by truncated hypercones. Since support vectors are at the vertexes of convex hulls, we can keep support vector candidates if we retain the vertexes of the convex hulls for the classes. But since it is difficult to generate a convex hull in the feature space, and the vertexes that are far away from the convex hull are unlikely to be support vectors in the future, we approximate the region of data that are near the separating hyperplane by truncated hypercones.

For each class, we generate a truncated surface with the center at the center of unbounded support vectors and with the radius being the maximum distance from the center to support vectors. The rotating axis goes through the center and is perpendicular to the truncated surface. We generate the surface of a hypercone so that it includes the datum that is far away from the separating hyperplane and the distance from the rotating axis is maximum. The data that are inside of the truncated hypercone are deleted and the remaining data are kept as support vector candidates.

In Section 2, we summarize SVMs and in Section 3, we explain two conventional methods. Then in Section 4, we propose an incremental training method using truncated hypercones and in Section 5 we compare our proposed method and the conventional methods from the standpoint of generalization ability and the deletion ratio of training data.

2 Support Vector Machines

In SVMs, the input $\mathbf{x}$ is mapped into the high dimensional feature space using the mapping function $\phi(\mathbf{x})$. For M input-output pairs $(\mathbf{x}_i, y(\mathbf{x}_i))$, $i = 1, ..., M$, let $y(\mathbf{x}_i) = 1$ if $\mathbf{x}_i$ belongs to class 1, and $y(\mathbf{x}_i) = -1$ if $\mathbf{x}_i$ belongs to class 2. We consider the following decision function in the feature space:

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

where $\mathbf{w}$ is a coefficient vector, b is a bias term, and if $\mathbf{x}$ is correctly classified, $y(\mathbf{x}_i) f(\phi(\mathbf{x}_i)) > 0$. If all the training data are correctly classified, $f(\phi(\mathbf{x})) = \mathbf{0}$ is called *separating hyperplane* and the minimum distance between the separating hyperplane and the training data is called *margin*.

In SVMs, the separating hyperplane is determined so that the margin is maximized while minimizing the classification error of the training data:

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 a margin parameter to control the tradeoff between maximization of margins and minimization of classification errors and ξ_i are slack variables associated with $\mathbf{x}_i$.

Introducing the Lagrange multipliers, the following dual problem is obtained:

Maximize

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

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 called *kernel function*:

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

In our study we use polynomial kernels with degree d:

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

and RBF (Radial Basis Function) kernels:

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

where $\gamma > 0$.

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 most important characteristic is that we can obtain the same solution using only support vectors.

Using support vectors, the decision function is expressed by

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

where S is the support vector indices and $\mathbf{w}$ is given by

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

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 Methods

In [5], support vector candidates are selected if

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

is satisfied, where $\beta(> 0)$ is a user defined parameter. If the separating hyperplane does not change much after retraining, future support vectors are kept by (12). But if the separating hyperplane rotates after retraining, support vector candidates tend to be deleted if the value of β is not properly selected.

In [6,7], concentric hyperspheres are used to approximate the regions for support vector candidates.

For class j $(j = 1, 2)$, we generate the minimum-volume hypersphere with radius R_j that includes all the training data in that class. Then we generate a concentric hypersphere with radius ρR_j, where $\rho\,(0 < \rho < 1)$ is a user defined parameter. Then we generate a hypercone with the vertex at the center of the hypersphere, which opens on the opposite side of the separating hyperplane. The openness of the hypercone is controlled by the angle, $\theta\,(-90 < \theta < 90)$, between the surface of the hypercone and the hyperplane that goes thorough the vertex of the hypercone and that is parallel to the separating hyperplane. We delete data that are inside of the hypersphere with radius ρR_j or in the hypercone unless they are not support vectors.

4 Proposed Method

4.1 Relations Between Support Vector Candidates and Vertexes of Convex Hulls

We consider what data should we keep to cope with the rotation of the separating hyperplane when data are added. To make matters simple we consider the separable case shown in Fig. 1. In the figure, the regions of the two classes are shown as convex hulls. The optimal separating hyperplane for this problem is shown in the figure. Now suppose that data are added but that these data are not in the regions of different classes. Then the separating hyperplane after incremental training will exist in the shaded region. Thus the data, which are the vertexes of the convex hulls and which are in the shaded region, are support vector candidates. Therefore, if we keep all the data that are vertexes, we can hold all the support vector candidates.

However, since usually we map the input space into the feature space and we do not explicitly treat variables in the feature space, generation of convex hulls is very difficult. In addition, by this method, redundant data on the vertexes that are far away from the separating hyperplane will be retained, which is inefficient. Thus, to solve this problem, we use truncated hypercones, which include the vertexes of the convex hulls but delete data that are far away from the separating hyperplane.

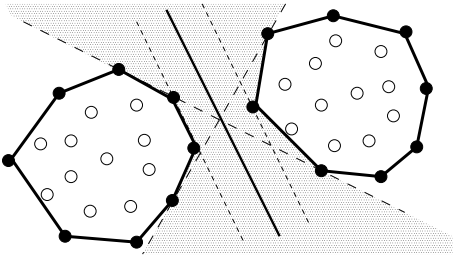

Fig. 1. Possible locations of separating hyperplanes and vertexes of convex hulls

4.2 Data Deletion Using Truncated Hypercones

Consider the case where a classification problem is linearly separable in the feature space as shown in Fig. 2. To keep the data whose images are near the separating hyperplane, we generate the truncated hypercones as shown in the figure and delete the data whose images are inside of the truncated hypercones. Data in black disks are retained and data in white disks are deleted.

Now we explain how to generate truncated hypercones. First, we generate truncated surface, whose center is the mean vector of mapped support vectors and the radius r_i $(j = 1, 2)$ is the maximum distance among the distances from the center to mapped support vectors. The rotating axis is the line that goes through the center and that is perpendicular to the separating hyperplane. If there is only one support vector like the right class in Fig. 2, a truncated hypercone shrinks to a hypercone with the mapped support vector being the center and with radius $r_2 = 0$.

For each class, among the mapped data whose distances from the separating hyperplane are longer than that of the class center of mapped training data, calculate the maximum distance from the rotating axis, R_i $(i = 1, 2)$. Use this as the radius that determine the slope of the truncated hypercone and generate the truncated hypercone. The reason why we exclude the mapped data that are nearer to the separating hyperplane than the class center of mapped training data is that these data are consider to be the candidate of support vectors.

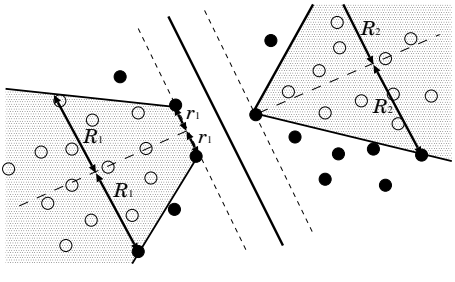

Fig. 2. Deletion of data using truncated hypercones

Generation of hypercones discussed so far is for separable problems such as shown in Fig. 2. For non-separable problems, mapped data inside of the convex hulls could be support vector candidates. To cope with this, we move the truncated hypercone in parallel along the rotating axis as shown in Fig. 3.

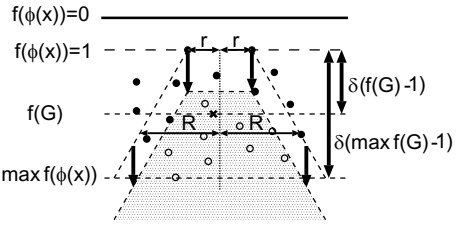

Fig. 3. Parallel movement of a truncated hypercone

To control parallel shifts we introduce three parameters: $f(\mathbf{G}_{\rm fs})$, max $f(\phi(\mathbf{x}))$, and $f(\phi(\mathbf{G}_{\rm is}))$, where $f(\mathbf{G}_{\rm fs})$ is the output of the decision function for the center vector of the mapped training data that are kept at an incremental training step, max $f(\phi(\mathbf{x}))$ is the output of the decision function for $\mathbf{x}$, whose associated distance from $\phi(\mathbf{x})$ to the separating hyperplane is the maximum among the data that are kept at an incremental training step, and $f(\phi(\mathbf{G}_{\rm is}))$ is the output of the decision function for the center vector of the training data added so far. If we calculate $f(\mathbf{G}_{\rm fs})$ using the data that are added so far we need to keep all the data. Thus, $f(\phi(\mathbf{G}_{\rm is}))$ is to approximate $f(\mathbf{G}_{\rm fs})$ calculated using all the data added so far.

Using these parameters we shift hyperplane by $m_1\,\delta\,(f(\mathbf{G}_{\rm fs})-1)$, $m_2\,\delta\,(\max f(\phi(\mathbf{x}))-1)$, or $m_3\,\delta\,(f(\phi(\mathbf{G}_{\rm is}))-1)$, where $m_1, m_2 (0 \le m_2 \le 1)$, and m_3 are user defined parameters.

The flow of incremental training using truncated hypercones is as follows:

1. Train an SVM using the initial training set X_a.
2. Add incremental data set X_b to X_a: $X_a = X_a \cup X_b$.
3. Using the data in X_a that satisfy $y(\mathbf{x})f(\phi(\mathbf{x})) \ge 1$, generate the truncated hypercones and shift them in parallel along rotating axes. If $\mathbf{x}$ does not satisfy $y(\mathbf{x})f(\phi(\mathbf{x})) \le 1$ and $\phi(\mathbf{x})$ is included in the shifted truncated hypercone, delete $\mathbf{x}$: $X_a = X_a - \{\mathbf{x}\}$.
4. If there is $\mathbf{x}$ in X_a that satisfies $y(\mathbf{x})f(\phi(\mathbf{x})) \le 1$, retrain the SVM.
5. Iterate Steps 2, 3, and 4 during incremental training.

In Step 3, we keep the data that satisfy $y(\mathbf{x})f(\phi(\mathbf{x})) \le 1$ because they are bounded support vectors and are support vector candidates after training. In Step 4, if there are no data that satisfy $y(\mathbf{x})f(\phi(\mathbf{x})) \le 1$, the separating hyperplane after retraining is the same. Thus, we do not retrain the SVM.

4.3 Determination of Inside of Truncated Hypercones

We judge whether a mapped datum is inside or outside of a truncated hypercone using r_i and R_i.

The center of the truncated surface, $\mathbf{a}_i$, for class i is given by

$$\mathbf{a}_i = \frac{1}{|S_i'|} \sum_{j \in S_i'} \phi(\mathbf{x}_j), \tag{13}$$

where S_i' is the index set of unbounded support vectors for class i and $|S_i'|$ is the number of unbounded support vectors for class i. Using (13), r_i is given by

$$\begin{aligned} r_i &= \max_{j \in S_i'} ||\phi(\mathbf{x}_j) - \mathbf{a}_i|| \\ &= \max_{j \in S_i'} \sqrt{K(\mathbf{x}_j, \mathbf{x}_j) - \frac{2}{|S_i'|} \sum_{k \in S_i'} K(\mathbf{x}_j, \mathbf{x}_k) + \frac{1}{|S'_i|^2} \sum_{k' \in S_i'} \sum_{k \in S_i'} K(\mathbf{x}_{k'}, \mathbf{x}_k)}. \end{aligned} \tag{14}$$

We use the following two center vectors to check if the distance from the separating hyperplane to a mapped datum $\phi(\mathbf{x})$ is longer than that to the center vector:

1. The use of $f(\mathbf{G}_{\rm fs})$ and max $f(\phi(\mathbf{x}))$ for the truncated hypercone shift
   The center vector in the feature space at an incremental training step is calculated using a set of class i data, X_i:

   $$\mathbf{G}_{\rm fs} = \frac{1}{|X_i|} \sum_{j \in X_i} \phi(\mathbf{x}_j). \tag{15}$$

   And the decision function $f(\mathbf{G}_{\rm fs})$ is given by

   $$f(\mathbf{G}_{\rm fs}) = \mathbf{w}^T \mathbf{G}_{\rm fs} + b = \frac{1}{|X_i|} \sum_{j \in X_i} \sum_{k \in S} y(\mathbf{x}_k) \alpha_k K(\mathbf{x}_j, \mathbf{x}_k) + b. \tag{16}$$

   For $\mathbf{x}$ that satisfies $y(\mathbf{x}) f(\phi(\mathbf{x})) \geq 1$ and $f(\phi(\mathbf{x})) > f(\mathbf{G}_{\rm fs})$, we calculate the distance from the rotating axis to $\phi(\mathbf{x})$ for the possible candidate of the radius of the truncated hypercone.
2. The use of $f(\phi(\mathbf{G}_{\rm is}))$ for the parallel shift
   For the set of data, $X_{\rm old}$, that includes the deleted data for class i, let the center vector be $\mathbf{G}_{\rm old}$. Assume that a set of data, $X_{\rm add}$, is added. Then the center vector, $\mathbf{G}_{\rm is}$, after addition is given by

   $$\mathbf{G}_{\rm is} = \frac{1}{|X_{\rm old}| + |X_{\rm add}|} (|X_{\rm old}| \mathbf{G}_{\rm old} + \sum_{k \in X_{\rm add}} \mathbf{x}_k). \tag{17}$$

   In this way, we can update the center vector in the input space without storing deletable data. Although mapping of the center vector into the feature

space does not coincide with the center vector in the feature space, it could be an approximation. The decision function for $\mathbf{G}_{\rm is}$ is given by

$$f(\phi(\mathbf{G}_{\rm is})) = \mathbf{w}^T\phi(\mathbf{G}_{\rm is}) + b = \frac{1}{|S|}\sum_{k\in S} y(\mathbf{x}_k)\alpha_k K(\mathbf{G}_{\rm is}, \mathbf{x}_k) + b. \tag{18}$$

For $\mathbf{x}$ that satisfies $y(\mathbf{x})f(\phi(\mathbf{x})) \geq 1$ and $f(\phi(\mathbf{x})) > f(\phi(\mathbf{G}_{\rm is}))$, we calculate the distance between the rotating axis and $\phi(\mathbf{x})$ for the possible candidate of the radius of the truncated hypercone.

Among $\mathbf{x}$ that satisfies $f(\phi(\mathbf{x})) > f(\mathbf{G}_{\rm fs})$ or $f(\phi(\mathbf{x})) > f(\phi(\mathbf{G}_{\rm is}))$, the maximum distance among the distances between the rotating axis and $\phi(\mathbf{x})$ is selected as the radius of the truncated hypercone. From Fig. 4, R_i is given by

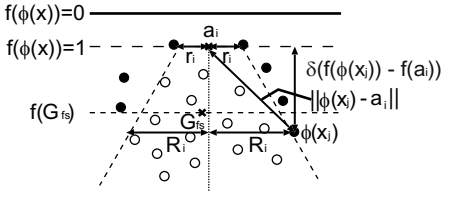

Fig. 4. Calculation of radius R_i

$$R_i = \max_{j\in X_i} \sqrt{||\phi(\mathbf{x}_j) - \mathbf{a}_i||^2 - \{\delta(f(\phi(\mathbf{x}_j)) - f(\mathbf{a}_i))\}^2}, \tag{19}$$

where

$$||\phi(\mathbf{x}_j) - \mathbf{a}_i|| = K(\mathbf{x}_j, \mathbf{x}_j) - \frac{2}{|S'_i|}\sum_{k\in S'_i} K(\mathbf{x}_j, \mathbf{x}_k) + \frac{1}{|S'_i|^2}\sum_{k\in S'_i}\sum_{k\in S'_i} K(\mathbf{x}_k, \mathbf{x}_k), \tag{20}$$

$$f(\mathbf{a}_i) = \mathbf{w}^T\mathbf{a}_i + b = \frac{1}{|S'_i|}\sum_{j\in S'_i}\sum_{k\in S} y(\mathbf{x}_k)\alpha_k K(\mathbf{x}_j, \mathbf{x}_k) + b. \tag{21}$$

Similar to the calculation of R_i, for $\mathbf{x}$ the distance, $d(\mathbf{x})$, between the rotating axis and $\phi(\mathbf{x})$ is given by

$$d(\mathbf{x}) = \sqrt{||\phi(\mathbf{x}) - \mathbf{a}_i||^2 - \{\delta(f(\phi(\mathbf{x})) - f(\mathbf{a}_i))\}^2}. \tag{22}$$

The radius L_i of the truncated hypercone, at which points the output of the decision function is the same with $f(\phi(\mathbf{x}))$ is given as follows.

1. The use of $f(\mathbf{G}_{\rm fs})$ for the truncated hypercone shift
   For $\mathbf{x}$ that satisfies $f(\phi(\mathbf{x})) > 1 + m_1(f(\mathbf{G}_{\rm fs}) - f(\mathbf{a}_i))$, we calculate L_i by

$$L_i = r_i + (R_i - r_i) \times \frac{\Big(f(\phi(\mathbf{x})) - m_1\{f(\mathbf{G}_{\rm fs}) - f(\mathbf{a}_i)\}\Big) - f(\mathbf{a}_i)}{f(R_i) - f(\mathbf{a}_i)}. \quad (23)$$

2. The use of max $f(\phi(\mathbf{x}))$ for the truncated hypercone shift
   For $\mathbf{x}_j (j \in X_i)$ belonging to class i, we calculate $\max_{j \in X_i} f(\phi(\mathbf{x}_j))$. Then for $\mathbf{x}$ that satisfies $f(\phi(\mathbf{x})) > 1 + m_2(\max_{j \in X_i} f(\phi(\mathbf{x}_j)) - f(\mathbf{a}_i))$, we calculate L_i by

$$L_i = r_i + (R_i - r_i) \times \frac{\Big(f(\phi(\mathbf{x})) - m_2\{\max_{j \in X_i} f(\phi(\mathbf{x}_j)) - f(\mathbf{a}_i)\}\Big) - f(\mathbf{a}_i)}{f(R_i) - f(\mathbf{a}_i)}. \quad (24)$$

3. The use of $f(\phi(\mathbf{G}_{\rm is}))$ in the truncated hypercone shift
   For $\mathbf{x}$ that satisfies $f(\phi(\mathbf{x})) > 1 + m_3(f(\phi(\mathbf{G}_{\rm is})) - f(\mathbf{a}_i))$, we calculate the following L_i:

$$L_i = r_i + (R_i - r_i) \times \frac{\Big(f(\phi(\mathbf{x})) - m_3\{f(\phi(\mathbf{G}_{\rm is})) - f(\mathbf{a}_i)\}\Big) - f(\mathbf{a}_i)}{f(R_i) - f(\mathbf{a}_i)}. \quad (25)$$

If we set m_1, m_2, m_3 to 0, we delete data without shifting truncated hypercones. If $d(\mathbf{x}) < L_i$
is satisfied we delete $\mathbf{x}$.

5 Performance Evaluation

We evaluated the effectiveness of the proposed method using the two-class benchmark problems listed in Table 1.[1] Each problem consists of 100 or 20 training and test data sets. Except for banana, ringnorm, and thyroid data sets, we normalized the input range into $[0, 1]$.

For each of 100 or 20 training data sets in a two-class problem, we generated the incremental training data sets, dividing each training data set into subsets with 5% of the training data. In each incremental training step, we added a subset to the classifier.

Training of SVMs was done by the primal-dual interior-point method combined with the decomposition techniques.

We compared the proposed and conventional methods for the optimal kernel and margin parameter C. We determined the optimal kernel by 5-fold cross-validation using the first five training data sets for the polynomial kernels with $d = [2, 3, 4]$ and RBF kernels with $\gamma = [0.1, 1, 10]$, and the margin parameter $C =$

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

Table 1. Specifications of two-class data sets

	Trn	Test	Inputs	Classes	Sets
Banana	400	4900	2	2	100
B. cancer	200	77	9	2	100
Diabetes	468	300	8	2	100
German	700	300	20	2	100
Heart	170	100	13	2	100
Image	1300	1010	18	2	20
Ringnorm	400	7000	20	2	100
F. solar	666	400	9	2	100
Splice	1000	2175	60	2	20
Thyroid	140	75	5	2	100
Twonorm	400	7000	20	2	100
Waveform	400	4600	21	2	100

Table 2. Parameter setting

Data	Kernel	Batch	Hplane	Sphere		Tcone-1	Tcone-2	Tcone-3
		C	β	ρ	θ	m_1	m_2	m_3
Banana	$\gamma 1$	10	2	0.5	0	0.5	0.08	0.5
B. cancer	$\gamma 1$	1	0.01	0.5	0	0.5	0.08	0.5
Diabetes	$d2$	50	0.5	0.5	0	0.5	0.08	0.5
German	$\gamma 1$	10	0.1	0.5	0	0.5	0.08	0.5
Heart	$\gamma 1$	50	0.5	0.5	0	0.5	0.08	0.5
Image	$\gamma 1$	1000	2	0.5	0	0.5	0.08	0.5
Ringnorm	$\gamma 0.1$	1	0.1	0.5	0	0.5	0.08	0.5
F. solar	$d2$	10	0.5	0.5	0	0.5	0.08	0.5
Splice	$\gamma 10$	1	0.1	0.5	0	0.5	0.08	0.5
Thyroid	$d2$	1	1	0.5	0	0.5	0.08	0.5
Twonorm	$d4$	50	2	0.5	0	0.5	0.08	0.5
Waveform	$\gamma 1$	1	0.1	0.5	0	0.5	0.08	0.5

$[1, 10, 50, 100, 500, 1000, 2000, 3000, 5000, 8000, 10000, 50000, 100000]$. (Please refer to [7] for the detailed selection procedure.)

For the conventional method that uses the hyperplane (Hplane), we set β so that the generalization ability is comparable with that of batch training. This parameter setting is to compare the numbers of deletable data and is not realizable. For the conventional method that uses hyperspheres (Sphere) we set $\rho = 0.5$ and $\theta = 0$ [7].

For the proposed method, since m_1 and m_3 are not upper bounded, we set m_1 and $m_3 = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 3, 5, 8, 10]$. Since m_2 satisfies $0 \leq m_2 \leq 1$, we set $m_2 = [0.01, 0.03, 0.05, 0.08, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$. As an initial test, using the diabetes data set, which has the medium number of training data and the medium number of the input variables (8 variables), the twonorm data set with 20 input variables, and waveform with

Table 3. Performance comparison (%)

Data	Term	Batch	Hplane	Sphere	Tcone-1	Tcone-2	Tcone-3
Banana	Test	89.31±0.53	89.31±0.53	89.31±0.53	*88.40±1.92*	*88.20±1.83*	89.31± 0.53
	Trn	91.95±1.30	91.93±1.30	91.93±1.30	91.14±2.45	91.03±2.24	91.94±1.30
	Del	73.6±2.3	64.3±3.7	**69.8±2.7**	70.5±4.1	71.5±3.2	54.2±5.4
B. cancer	Test	73.25±4.53	73.25±4.53	73.25±4.53	73.25±4.53	73.25±4.53	73.25±4.53
	Trn	82.80±1.72	82.80±1.72	82.80±1.72	82.80±1.71	82.80±1.72	82.80±1.72
	Del	34.6±2.7	0.1±0.3	0.1±0.2	0.0±0.1	0.0±0.2	0.0±0.0
Diabetes	Test	76.46±1.85	76.46±1.85	76.46±1.85	76.44±1.88	76.44±1.88	76.39±1.90
	Trn	78.48±1.22	78.48±1.22	78.48±1.22	78.52±1.18	78.48±1.23	78.43±1.27
	Del	45.5±1.7	15.7±8.3	**27.8±13.4**	**31.4±16.1**	**26.9±16.6**	**31.6±16.2**
German	Test	76.63±2.14	76.70±2.26	76.63±2.14	76.63±2.14	76.65±2.17	76.63±2.12
	Trn	81.14±1.27	80.94±1.41	81.14±1.27	81.14±1.27	81.14±1.29	81.14±1.30
	Del	43.8±1.5	33.8±5.7	**35.5±3.7**	30.8±6.8	33.4±4.7	21.4±6.7
Heart	Test	83.68±3.39	83.68±3.39	83.68±3.39	82.84±4.58	82.83±5.57	83.34±3.43
	Trn	85.95±1.92	85.95±1.92	85.95±1.92	85.29±3.90	85.32±4.00	85.74±2.18
	Del	56.3±3.4	24.9±21.9	**32.4±27.6**	**35.4±25.8**	**32.3±23.2**	**36.0±26.1**
Image	Test	97.14±0.48	97.12±0.47	97.13±0.48	*96.30±0.71*	*96.48±0.50*	*96.03±1.11*
	Trn	98.60±0.18	98.60±0.17	98.60±0.18	97.82±0.44	98.08±0.31	97.41±0.80
	Del	88.3±0.7	**60.2±3.8**	**61.7±4.7**	80.6±4.7	83.4±1.9	83.1±5.4
Ringnorm	Test	98.41±0.10	98.41±0.10	98.41±0.10	98.41±0.10	98.41±0.10	98.41±0.10
	Trn	99.91±0.15	99.91±0.15	99.91±0.15	99.91±0.15	99.91±0.15	99.91±0.15
	Del	61.5±2.3	**45.6±16.3**	31.0±11.1	31.5±11.7	35.9±13.0	24.2±9.6
F. solar	Test	68.29±1.85	68.29±1.85	68.29±1.85	*67.44±2.34*	68.30±1.85	*67.47±2.56*
	Trn	68.19±1.26	68.19±1.26	68.19±1.26	67.82±1.84	68.19±1.26	67.81±1.85
	Del	18.9±1.9	4.5±3.5	**11.5±8.6**	12.8±10.5	**11.4±9.7**	11.5±10.5
Splice	Test	88.66±0.71	88.66±0.72	88.66±0.71	88.66±0.71	88.66±0.71	88.66±0.71
	Trn	99.09±0.24	99.09±0.24	99.09±0.24	99.09±0.24	99.10±0.24	99.09±0.24
	Del	26.5±1.3	**11.9±10.7**	**13.2±12.0**	**13.5±12.2**	**13.6±12.3**	**8.4±7.7**
Thyroid	Test	96.31±1.90	96.31±1.90	96.31±1.90	95.43±2.63	95.84±2.29	95.33±4.34
	Trn	99.34±0.49	99.34±0.49	99.34±0.49	98.50±2.03	98.82±1.02	97.84±4.70
	Del	88.9±1.3	66.5±11.3	60.9±6.7	**83.0±3.6**	**82.0±7.4**	78.1±17.3
Twonorm	Test	97.57±0.12	97.57±0.12	97.57±0.12	97.56±0.13	97.57±0.12	97.57±0.12
	Trn	98.24±0.55	98.24±0.55	98.24±0.55	98.23±0.56	98.24±0.55	98.24±0.55
	Del	79.1±1.6	57.5±6.4	51.2±6.8	47.6±5.8	**71.3±7.5**	47.4±5.6
Waveform	Test	90.00±0.45	90.00±0.45	90.00±0.45	90.00±0.43	90.00±0.44	90.00±0.44
	Trn	93.51±1.37	93.51±1.37	93.51±1.37	93.52±1.38	93.51±1.38	93.51±1.39
	Del	61.6±2.2	**36.8±28.0**	**38.2±25.7**	**37.3±25.2**	**39.8±26.8**	**34.1±23.0**

21 input variables, we incrementally trained the SVM with the optimal kernel and the optimal value of C for the above parameter setting of m_1, m_2, and m_3 and selected $m_1 = 0.5$, $m_2 = 0.08$, and $m_3 = 0.5$, by which setting sufficient data deletion was done with the generalization performance comparable to that of batch training.

Table 2 lists the parameter values used in the experiments. In the table, "Kernel" denotes the kernel and the parameter value determined by cross-validation. For instance, $\gamma 1$ means RBF kernels with $\gamma = 1$ and $d2$ means polynomial kernels with degree $d = 2$. The results are shown in "Batch," "Hplane," "Sphere," and "Tcone" columns. For the proposed method, "Tcone-i" ($i = 1, 2, 3$) denotes that the parameter m_i is used. From the table, the optimal value of β changes as the classification problem changes.

Table 3 lists the results. "Test" and "Trn" rows show the average recognition rates and their standard deviations of the test and training data when incremental training is finished. The "Del" row shows the number of deleted training data divided by the number of training data. For batch training, the number of deleted training data is calculated by the number of training data minus the number of support vectors. Thus, the "Del" value for batch training is the upper bound of that of incremental training if all the support vectors are kept by incremental training.

The recognition rates of the test data shown in italic are statistically different from those of batch training with the significance level of 0.05. The deletion ratios in boldface are the group that realizes the best deletion of data, in which the deletion ratio of any member of the group is statistically the same.

From Table 3, Sphere and Tcone-2 show the best performance and Tcone-1, Tcone-3, and Hplane show the second best. Although the recognition rates of the test data for Sphere are comparable with those of batch training, those for Tcone are in some cases lower. Thus, the parameter selection of Sphere is more robust than that of Tcone.

6 Conclusions

In this paper, to reduce memory cost by deleting unnecessary data, we proposed an incremental training method for SVMs, which is robust for rotation of separating hyperplanes after incremental training.

Based on the fact that support vectors form vertexes of convex hulls for classes, we use truncated hypercones to keep the data that are near the separating hyperplanes. We generate the truncated surface with the center being the center of unbounded support vectors and the radius being the maximum distance between the center and unbounded support vectors. The rotating axis goes through the center and is perpendicular to the truncated surface. The rotating surface includes the datum, which is far away from the separating hyperplane and which is farthest from the rotating axis. We delete data that are inside of the truncated hypercone and keep the remaining data.

For two-class problems, we showed that, in most cases, we could delete many training data while keeping the generalization ability comparable to that of batch training.

References

1. V. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, 1998.
2. V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, 1995.
3. P. Mitra, C. A. Murthy, S. K. Pal, "Data Condensation in Large Databases by Incremental Learning with Support Vector Machines," *Proc. ICPR 2000*, pp. 2708–2711, 2000.
4. C. Domeniconi and D. Gunopulos, "Incremental Support Vector Machine Construction," *Proc. ICDM 2001*, pp. 589–592, 2001.
5. G. Cauwenberghs and T. Poggio, "Incremental and Decremental Support Vector Machine Learning," In T. K. Leen, T. G. Dietterich, and V. Tresp, Eds., *Advances in Neural Information Processing Systems 13*, pp. 409–415, MIT Press, 2000.
6. S. Katagiri and S. Abe, "Selecting Support Vector Candidates for Incremental Training," *Proc. SMC 2005*, pp. 1258–1263, 2005.
7. S. Katagiri and S. Abe, "Incremental Training of Support Vector Machines Using Hyperspheres," *Pattern Recognition Letters* (accepted).
8. D. M. J. Tax and R. P. W. Duin, "Outliers and Data Descriptions," *Proc. Seventh Annual Conference of the Advanced School for Computing and Imaging*, pp. 234–241, 2001.