DECISION-TREE-BASED MULTICLASS SUPPORT VECTOR MACHINES

Fumitake Takahashi, Shigeo Abe

Graduate School of Science and Technology, Kobe University, Kobe, Japan
(E-mail: abe@eedept.kobe-u.ac.jp)

ABSTRACT

In this paper, we propose decision-tree-based multiclass support vector machines. In training, at the top node, we determine the hyperplane that separates a class (or some classes) from the others. If the separated classes include plural classes, at the node connected to the top node, we determine the hyperplane that separates the classes. We repeat this procedure until only one class remains in the separated region. This can resolve the unclassifiable regions that exist in the conventional SVMs, but a new problem arises. Namely, the division of the feature space depends on the structure of a decision tree. To maintain high generalization ability, the most separable classes should be separated at the upper nodes of a decision tree. For this, we propose four types of decision trees based on separability measured by the Euclidean distances between class centers and Mahalanobis-distance-based classifiers. We demonstrate the effectiveness of our methods over conventional SVMs using benchmark data sets.

1. INTRODUCTION

Support vector machines (SVM) [1] are the classifiers which are formulated for a two-class problem. Though SVMs are known to have a high generalization ability for two-class problems, for $N(> 2)$-class problems, we must determine N hyperplanes, each of which separates one class from the others, and this leads to the existence of unclassifiable regions. To resolve this problem, some methods, for example, pairwise SVMs [2] and fuzzy SVMs [3], have been developed.

In this paper, we propose decision-tree-based SVMs. In training, at the top node, we determine the hyperplane that separates a class (or some classes) from the others. If the separated classes include plural classes, at the node connected to the top node, we determine the hyperplane that separates the classes. We repeat this procedure until only one class remains in the separated region.

When training is finished, the feature space is divided by $N - 1$ hyperplanes and there is no unclassifiable region in the feature space. Therefore, this method can resolve the problem of conventional SVMs. But since the region of each class depends on the structure of a decision tree, we must determine the structure of the decision tree so that the classification error is minimized. If the classification performance is not good at the upper node of the decision tree, the overall classification performance becomes worse. Therefore, more separable classes should be separated at the upper node of the decision tree. We propose four types of decision trees based on separability measured by Euclidean distances between class centers and Mahalanobis-distance-based classifiers.

This paper is organized as follows. In Section 2, we briefly describe the theory of conventional SVMs. In Section 3, we propose the decision-tree-based SVMs to resolve the problem of conventional SVMs for a multiclass problem and Section 4 demonstrates the effectiveness of our methods by computer experiments.

2. SUPPORT VECTOR MACHINES

In this section, we review the theory of SVMs for a two-class problem and discuss a problem of the conventional SVMs for a multiclass problem.

2.1. SVMs for a two-class problem

Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$ be the training data, and $y_1, \ldots, y_M$ be the associated class labels. Here if $\boldsymbol{x}_i$ belongs to Class 1, $y_i = 1$, and to Class 2, $y_i = -1$. In SVMs, we first map the input space $\boldsymbol{x}$ into a high dimensional feature space $\Phi(\boldsymbol{x})$ in order to enhance separability. Our aim is to obtain a linear decision function $f(\Phi(\boldsymbol{x})) = \boldsymbol{w}^t\Phi(\boldsymbol{x}) + b$ in the high dimensional feature space. Here $\boldsymbol{w}$ is a weight vector whose dimension is the dimension of the feature space, and b is a bias term. Mapping function Φ needs to satisfy the following equation:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)^t\Phi(\boldsymbol{x}_j), \tag{1}$$

where $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is called kernel function. The Euclidean distance from a training data $\boldsymbol{x}$ to a hyperplane $f(\Phi(\boldsymbol{x})) = 0$ is given by $|f(\Phi(\boldsymbol{x}))|/||\boldsymbol{w}||$. The optimal hyperplane in SVMs is realized by maximizing this distance from the nearest datum. So we impose the condition on the numerator,

$\min |f(\Phi(\boldsymbol{x}))| = 1$ and the optimal hyperplane can be obtained by solving the following optimization problem:

$$\begin{array}{rc} \text{minimize} & \frac{1}{2}||\boldsymbol{w}||^2 \\ \text{subject to} & y_i(\boldsymbol{w}^t\Phi(\boldsymbol{x}_i)+b) \geq 1 \\ \text{for} & i = 1, ..., M. \end{array} \tag{2}$$

We cannot solve this problem with the infinite feature space. Therefore, we convert this problem to a dual problem using Lagrange multipliers α_i, and get the following quadratic optimization problem:

$$\begin{array}{rc} \text{maximize} & \displaystyle\sum_{i=1}^{M}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{M}\alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{subject to} & \displaystyle\sum_{i=1}^{M} y_i\alpha_i = 0. \end{array} \tag{3}$$

Solving this problem, we obtain the following decision function:

$$f(\boldsymbol{x}) = \sum_{i=1}^{M}\alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b. \tag{4}$$

Usually, most of α_i are 0, and only the data associated with non-zero α_i affect the hyperplane. These data are called support vectors. From (4), we can see that by using kernel function given by (1), we can easily treat a high dimensional feature space without using mapping functions. We classify the datum $\boldsymbol{x}$ as follows:

$$f(\boldsymbol{x}) \begin{cases} > 0 & \text{for Class 1,} \\ < 0 & \text{for Class 2.} \end{cases} \tag{5}$$

In the above discussion, we assumed that the training data are linearly separable. If not linearly separable, we use a soft margin technique, introducing slack variables. By the discussion similar to the separable case, the following quadratic optimization problem is derived:

$$\begin{array}{rc} \text{maximize} & \displaystyle\sum_{i=1}^{M}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{M}\alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{subject to} & \displaystyle\sum_{i=1}^{M} y_i\alpha_i = 0, \quad 0 \leq \alpha_i \leq C. \end{array} \tag{6}$$

Here C is the upper bound that determines the tradeoff between the maximization of margin and minimization of classification error and is set to a large value.

2.2. Problem of SVMs for a multiclass classification

As seen in Section 2.1, SVMs are originally formulated for a two-class problem, so we must convert an N-class problem into N two-class problems. In the ith two-class problem, class i is separated from the remaining classes. Namely, in the ith problem, the class labels of data which belongs to class i are set as $y = 1$, and the class labels of remaining data are set as $y = -1$. In classification, if only the value of $f_i(\boldsymbol{x})$ is positive and all the values for the remaining decision functions are negative, $\boldsymbol{x}$ is classified into class i. But in this method, plural decision functions may be positive or all the decision functions may be negative. Fig. 1 shows an example of the unclassifiable regions for a three-class problem. The data in the shaded regions cannot be classified. This leads to the degradation of generalization ability.

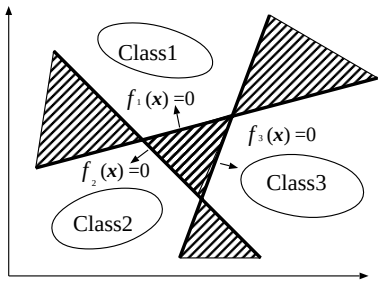

Figure 1: The existence of unclassifiable regions

3. SVMS BASED ON THE DECISION TREE

To solve the problem discussed in Section 2.2, we propose SVMs based on the decision tree formulation. In training, at the top node of the decision tree, we determine the hyperplane that separates one or some classes from remaining classes in the feature space. If plural classes are in the separated subspace, at the node connected to the top node, we determine the hyperplane that separates the classes. We repeat this procedure until there are one-class data in the separated regions.

In classification, starting from the top of the decision tree, we calculate the value of the decision function for input data $\boldsymbol{x}$ and according to the sign of the value we determine which node to go to. We iterate this procedure until we reach a leaf node and classify the input into the class associated with the node.

For an N-class problem, the number of hyperplanes to be calculated is $N - 1$. It is less than that of the conventional method, N. And as learning proceeds, the number of data needed for learning becomes smaller, so we can expect shorter training time. In classification, in the conventional method, the values for all the decision functions need to be calculated. On the contrary, in the proposed method, though it depends on the structure of the decision tree, the values of all the decision functions are not necessarily calculated.

Fig. 2 shows the example of the division of the feature space, and Fig. 3 expresses this by the decision tree. In

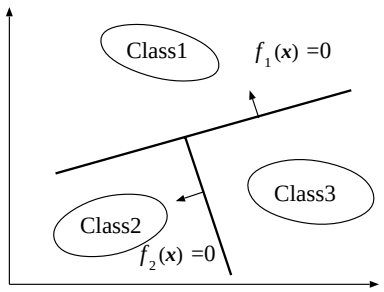

Figure 2: The example of the division of feature space

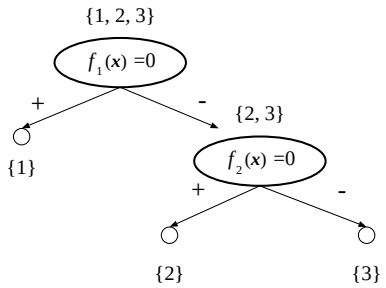

Figure 3: Expression by decision tree

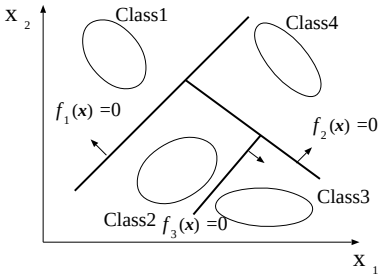

Figure 4: A four-class problem example 1

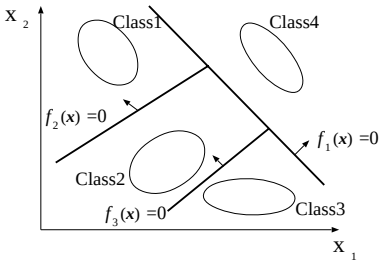

Figure 5: A four-class problem example 2

the training step, in this example, at first, the hyperplane $f_1(\boldsymbol{x})$ which separates Class 1 from Classes 2 and 3 is calculated. Secondly, for remaining classes (Classes 2 and 3), the hyperplane $f_2(\boldsymbol{x})$ which separates Class 2 from Class 3 is calculated. In the classification step, for input datum $\boldsymbol{x}$, we first calculate a value of $f_1(\boldsymbol{x})$. If it is positive, $\boldsymbol{x}$ is classified into Class 1, and if negative, calculate a value of $f_2(\boldsymbol{x})$. If it is positive, $\boldsymbol{x}$ is classified into Class 2, but if negative, classified into Class 3. So we can see the testing time can be reduced. From this example, we can see that there is no unclassifiable region in the feature space.

But a problem is how to determine the structure of the decision tree. Figs. 4 and 5 are the examples of a four-class problem. In Fig. 4, at first the hyperplane which separates Class 1 from Classes 2, 3, 4 is calculated. Next, the hyperplane which separates Class 4 from Classes 2, 3 is calculated and finally the hyperplane which separates Class 3 from Class 2 is calculated. In Figure 5, the separating order is Classes 4, 1, 2. As seen from this example, the region of each class depends on the structure of a decision tree.

Since the more the data are misclassified at the upper node of the decision tree, the worse the classification performance becomes, the classes that are easily separated need to be separated at the upper node of the decision tree. To determine the decision tree, we use the fact that the physical relationship of data in input space is kept in the feature space. We propose four types of decision trees as follows.

1. **Type 1 decision tree.** At each node one class is separated from the remaining classes. The Euclidean distance is used as a separability measure.
2. **Type 2 decision tree.** At each node some classes are separated from the remaining classes. The Euclidean distance is used as a separability measure.
3. **Type 3 decision tree.** At each node one class is separated from the remaining classes. Classification errors by the Mahalanobis distance is used as a separability measure.
4. **Type 4 decision tree.** At each node some classes are separated from the remaining classes. Classification errors by the Mahalanobis distance is used as a separability measure.

In the following, we discuss these algorithms in detail for an N-class problem.

3.1. Type 1 decision tree

In this method, we calculate the Euclidean distances between the class centers, and recursively separate the farthest class from the remaining classes.

Step 1 Calculate the class centers $\boldsymbol{c}_i(i = 1, ..., N)$ by

$$\boldsymbol{c}_i = \frac{1}{|X_i|} \sum_{\boldsymbol{x} \in X_i} \boldsymbol{x} \tag{7}$$

and the distance between class i and class j, $d_{ij}(i, j = 1, ..., N)$, by

$$d_{ij}(= d_{ji}) = ||\boldsymbol{c}_i - \boldsymbol{c}_j||. \tag{8}$$

Here X_i is a set of training data included in class i, and $|X_i|$ is the number of elements included in X_i.

Step 2 Find the smallest value of d_{ij} for each class. Namely, the smallest value of class i,

$$l_i = \min_{j=1,\ldots,N, j\neq i} d_{ij}, \tag{9}$$

and regard the class which has the largest l_i as the farthest class and calculate a hyperplane which separates this class. Namely, separate class k from the others. Here $k = \arg\max_{i=1,..,N} l_i$. If k exists for plural i, for these classes, compare the next smallest distance l_i' and $k = \arg\max_i l_i'$.

Step 3 For the remaining classes, repeat Step 2.

3.2. Type 2 decision tree

Here using the distances between class centers as Section 3.1, repeat merging the two classes which are the nearest until two clusters are obtained, and separate the clusters by a hyperplane.

Step 1 Using (7), calculate the class centers, and calculate the distances between class i and class j, $d_{ij}(i, j = 1, ..., N)$ by (8). Here let all the classes belong to different clusters.

Step 2 For the classes which belong to different clusters, calculate the smallest value of distances by (9) and let the associated two classes belong to the same cluster.

Step 3 Repeat Step 2 $N-2$ times to merge to two clusters.

Step 4 Calculate a hyperplane which separates the clusters generated in Step 3.

Step 5 If the separated cluster in Step 4 has $N'(>2)$ classes, regard the classes as belonging to different clusters and repeat Step 3 $N'-2$ times and go to Step 4. If $N'=2$, calculate a hyperplane which separate the two classes.

3.3. Type 3 decision tree

First, we classify the training data using the Mahalanobis distance and determine a hyperplane which separates the class with the smallest misclassifications from the remaining classes.

Step 1 For each class, calculate the covariance matrix Q_i $(i = 1, ..., N)$ by

$$Q_i = \frac{1}{|X_i|} \sum_{\boldsymbol{x}\in X_i} (\boldsymbol{x} - \boldsymbol{c}_i)(\boldsymbol{x} - \boldsymbol{c}_i)^t, \tag{10}$$

where $\boldsymbol{c}_i$ is a center vector of class i given by (7). Calculate the Euclidean distance between class centers using (8). For all the data, calculate the Mahalanobis distance $d_i(\boldsymbol{x})(i = 1, ..., N)$ and classify to the nearest class. Here

$$d_i{}^2(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{c}_i)^t {Q_i}^{-1} (\boldsymbol{x} - \boldsymbol{c}_i). \tag{11}$$

Step 2 Let e_{ij} be the number of misclassified data of class i into class j.

Step 3 Calculate the number of misclassified data for class $i(i = 1, \ldots, N)$ by $\sum_{j=1,\ j\neq i}^{n} (e_{ij} + e_{ji})$ and the class which has the smallest value is separated from the others. If plural classes have the same value, separate the class with the farthest Euclidean distance among these classes.

Step 4 Repeat Steps 2, 3 for the remaining classes.

3.4. Type 4 decision tree

In this method, we at first classify the data using Mahalanobis distance as in Section 3.3, and repeat merging the most misclassified two classes.

Step 1 Do Steps 1, 2 in Section 3.3.

Step 2 For the two classes in different clusters, find the largest value of e_{ij} and regard classes i and j as belonging to the same cluster. If plural classes have the same value, similar to type 2 decision tree, merge the nearest two classes.

Step 3 Repeat Step 2 until the number of clusters becomes two, and calculate a hyperplane which separates these two clusters.

Step 4 If the number of classes in a cluster separated in Step 3 is $N' > 2$, let these classes belong to different clusters and repeat Step 2 $N'-2$ times. And go to Step 3. If $N'=2$, calculate a hyperplane which separates these two classes.

4. SIMULATION EXPERIMENTS

To evaluate the performance of our methods, we simulated using hiragana data for license plate recognition and the blood cell data. Table 1 shows the numbers of inputs, classes, training data, and test data. In simulation, we used a Pentium III 1GHz PC to measure the number of misclassified data for training and test data, training and testing time. To see the effectiveness of our method, we compared our methods with a method, in which the class is separated in reverse

Table 1: Feature of benchmark data

Data	Inputs	Classes	Train.	Test
Hiragana	50	39	4610	4610
Blood cell	13	12	3097	3100

Table 2: Number of misclassified data for hiragana data

	$d=2$	$d=3$	$d=4$
Type 1 rev.	146	125	117
Type 1	104	92	89
Type 2	87	77	57
Type 3	108	103	98
Type 4	68	66	69
C-SVM	197	171	163

order of type 1 decision tree (type 1 rev.), and the conventional SVMs. We use the polynomial kernels:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = [(\boldsymbol{x}_i^t \boldsymbol{x}_j) + 1]^d. \tag{12}$$

Table 2 shows the performance for the hiragana data. For hiragana data, the performance for the training data is 100% and we show the result for the test data. The generalization ability by the proposed methods is superior to those by the type 1 rev. and conventional SVMs. The best performance for hiragana data is realized by the type 4 decision tree for $d = 2, 3$, and the type 2 decision tree for $d = 4$. Table 3 shows the executing time for training and testing, the testing time is in the brackets. The executing time by the proposed methods is shorter than that by the conventional SVM.

Table 4 shows the performance for blood cell data. The performance for training data is in the brackets. In regard to the training data, the best performance was realized by the proposed method, but the type 1 rev. shows the best performance for the test data for $d = 2, 3$. All the proposed methods are superior to the conventional SVM. Table 5 shows the executing time for training and testing. The executing time by the proposed methods is shorter than that by the conventional SVM.

Table 3: Learning and testing time for hiragana data [sec]

	$d=2$	$d=3$	$d=4$
Type 1	84 (14)	86 (16)	97 (20)
Type 2	77 (12)	81 (14)	83 (14)
Type 3	100 (15)	101 (17)	104 (18)
Type 4	90 (12)	97 (13)	96 (14)
C-SVM	112 (18)	115 (18)	120 (20)

Table 4: Number of misclassified data for blood cell data

	$d=2$	$d=3$	$d=4$
Type 1 rev.	257	271	308
	(59)	(18)	(3)
Type 1	284	272	288
	(43)	(20)	(8)
Type 2	287	289	292
	(27)	(9)	(0)
Type 3	296	299	292
	(26)	(4)	(0)
Type 4	272	274	284
	(33)	(8)	(0)
C-SVM	350	340	405
	(191)	(118)	(65)

Table 5: Learning and testing time for blood data [sec]

	$d=2$	$d=3$	$d=4$
Type 1	16 (1.1)	12 (1.0)	12 (0.94)
Type 2	10 (0.72)	9.2 (0.69)	9.7 (0.70)
Type 3	8.4 (0.70)	8.8 (0.68)	8.9 (0.71)
Type 4	12 (0.91)	9.5 (0.69)	10 (0.73)
C-SVM	35 (3.5)	30 (3.0)	29 (2.8)

5. CONCLUSION

In this paper, we proposed new formulation of SVMs for a multiclass problem. This method can resolve the existence of unclassifiable regions and has higher generalization ability than that of the conventional method. And the executing time was also shown to be shortened. Though the performance of the four decision trees depends on the data, all the algorithms realized high generalization ability.

6. REFERENCES

[1] V. N. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, 1998.

[2] U. H.-G. Kreßel, Pairwise Classification and Support Vector Machines, *Advances in Kernel Methods: Support Vector Learning* (B. Schölkopf, C. J. C. Burges, and A. J. Smola (Eds.)), pp. 255–268. The MIT Press, 1999.

[3] T. Inoue and S. Abe, Fuzzy Support Vector Machines for Pattern Classification, *Proc. IJCNN'01*, pp. 1449–1454, 2001.