Maximizing margins of multilayer neural networks

Nishikawa, Takahiro

Abe, Shigeo

(Citation)
Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on,1:322-326

(Issue Date)
2002-11

(Resource Type)
conference paper

(Version)
Accepted Manuscript

(URL)
https://hdl.handle.net/20.500.14094/90000227

MAXIMIZING MARGINS OF MULTILAYER NEURAL NETWORKS

Takahiro Nishikawa, Shigeo Abe

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

ABSTRACT

According to the CARVE algorithm, any pattern classification problem can be synthesized in three layers without misclassification. In this paper, we propose to train multilayer neural network classifiers based on the CARVE algorithm. In hidden layer training, we find a hyperplane that separates a set of data belonging to one class from the remaining data. Then, we remove the separated data from the training data, and repeat this procedure until only the data belonging to one class remain. In determining the hyperplane, we maximize margins heuristically so that data of one class are on one side of the hyperplane. In output layer training, we determine the hyperplane by a quadratic optimization technique. The performance of this new algorithm is evaluated by some benchmark data sets.

1. INTRODUCTION

Support vector machines (SVMs) have been proposed as a new paradigm for pattern classification. In SVMs the input space is mapped into a high dimensional feature space, and in the feature space the hyperplane is determined so that class margins are maximized. This leads to improvement in generalization ability. Using sigmoid kernel functions, a three layer neural network can be trained. But since SVMs are based on two-class problems, for multiclass problems unclassifiable regions are generated and this leads to degradation of generalization ability.

The CARVE algorithm [1, 2] guarantees that any pattern classification problem can be synthesize in three layers if we train the hidden layer in the following way. First, we separate a part of (or whole) data belonging to a class from the remaining data by a hyperplane. Then we remove the separated data from the training data. We repeat this procedure until only the data belonging to one class remain.

In this paper, we propose a new method of training neural network classifiers based on the CARVE algorithm. To improve generalization ability, we maximize margins of hidden layer hyperplanes and output layer hyperplanes. Since the training data on one side of the hidden layer hyperplane need to belong to one class, we cannot apply the quadratic optimization technique used for training SVMs. Therefore, we extend the heuristic training method called DirectSVMs [3], which sequentially search support vectors. For the output layer, since there is no such restriction, we use the quadratic optimization technique.

The paper is organized as follows. In Section 2 we presents an overview of the CARVE algorithm. In Section 3 we extend the DirectSVM so that it can be used for training hidden layer hyperplanes. In Section 4, we discuss the training method of output layer hyperplanes by the quadratic optimization technique. In Section 5, we compare performance of the proposed method with that of conventional methods using some benchmark data sets.

2. CARVE ALGORITHM

By the CARVE algorithm [1, 2] any pattern classification problem can be synthesized in three layers. In the hidden layer, we determine the hyperplane, on one side of which data of a single class exist. Then the separated data are removed from the training data. The hidden layer training is completed when only the data of a single class remain.

We explain the procedure using the example shown in Fig. 1. The class data shown in circles include a datum which is the farthest from the center of all the training data shown in the asterisk. Thus we separate the data of this class from the remaining data. As shown in Fig. 2, a hyperplane is determined so that data of this class are on one side of the hyperplane. Then the separated data colored in gray are removed from the training data. Then for the reduced training data, the class data shown in triangles include a datum which is the farthest from the center. Thus we determine a hyperplane so that only the data of this class is on one side of the hyperplane as shown in Fig. 3. We repeat this procedure until the remaining training data belong to a single class (Fig. 4). We say that a hyperplane satisfies the CARVE condition if on one side of the hyperplane only data of one class exist.

Let the input of a hidden unit be x, which is the output of the decision function associated with the hidden layer hyperplane. The output of the hidden unit, $z(x)$, is calculated

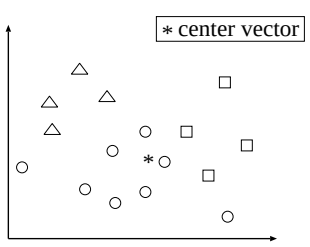

Figure 1: Sample data in the hidden layer

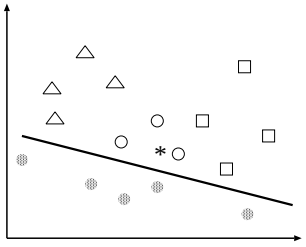

Figure 2: Create the 1st hyperplane

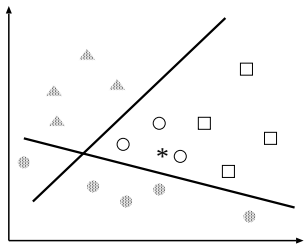

Figure 3: Create the 2nd hyperplane

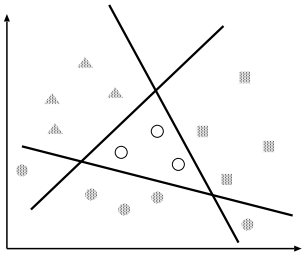

Figure 4: All active data belong to one class

by

$$z(x) = \frac{1}{1 + \exp(-x/T)}, \tag{1}$$

where T is a parameter for slope control.

In the output layer, we determine a hyperplane for class i so that class i data are separated from all other data (Fig. 5). Since there are no constraints for the output layer, we use the training method for conventional SVMs. In this way we can construct a three layer neural network for an n-class problem.

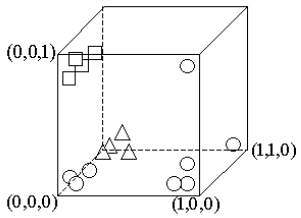

Figure 5: Sample data in the output layer

3. DETERMINATION OF HIDDEN LAYER HYPERPLANES

We determine a hidden layer hyperplane in the input space. Thus, if the problem is not linearly separable, the hyperplane determined by the quadratic programming technique may have data of one class on both sides of the hyperplane. Therefore, we cannot use the quadratic optimization technique for determining the hidden layer hyperplanes.

To overcome this problem, we extend the DirectSVM [3], which determines the optimal hyperplane geometrically. Let the farthest datum from the center of the training data belong to class i. Initially, we consider separating class i data from the remaining data. And we set the target values of the data which belong to class i to $+1$, the target values of the data which belong to other classes to -1. We call the side of the hyperplane where the data with the positive targets reside positive side of the hyperplane.

First, as initial support vectors we choose the data pair that has the minimum distance among the data pairs with opposite target values. The initial hyperplane is determined so that it goes through the center of the data pair and orthogonal to the line segment that connects the data pair. If there are data with negative targets on the positive side of the hyperplane, these data violate the CARVE condition. Thus, to satisfy the CARVE condition, we rotate the hyperplane until no data violate the condition. The previously violating data are added to the support vector. In the following, we describe the procedure more in detail.

3.1. Rotation of a hyperplane

Let the initial support vectors be $\mathbf{x}_0^+$ and $\mathbf{x}_0^-$. Then, the weight vector of the initial hyperplane is given by

$$\mathbf{w}_0 = \mathbf{x}_0^+ - \mathbf{x}_0^-. \tag{2}$$

The hyperplane can be written by

$$\mathbf{w}_0 \cdot \mathbf{x} - \mathbf{w}_0 \cdot \mathbf{c}_0 = 0, \tag{3}$$

where $\mathbf{c}_0$ is the center of the data pair $(\mathbf{x}_0^+, \mathbf{x}_0^-)$, and the second term is the bias. If there are no data that violate the CARVE condition in the remaining data, we finish training. If there are violating data of negative targets, we add the most violating data to the support vectors. Let the obtained support vector be $\mathbf{x}_1^-$, and the center of the data pair $(\mathbf{x}_0^+, \mathbf{x}_1^-)$ be $\mathbf{c}_1$. The hyperplane is updated so that it passes through the two data $\mathbf{c}_0$ and $\mathbf{c}_1$. If there are still violating data, we repeat updating.

Consider the m-th updating in n-dimensional space. Here, in case of the m-th updating, we need to update the hyperplane so that it passes through the centers of support vector pairs $\mathbf{c}_0, ..., \mathbf{c}_m$. Thus the maximum number of updating is $(n-1)$. When we update the hyperplane $(n-1)$

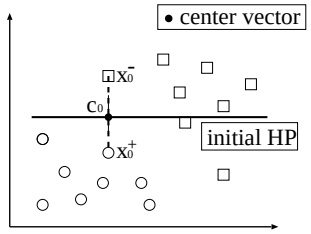

Figure 6: Initial HP

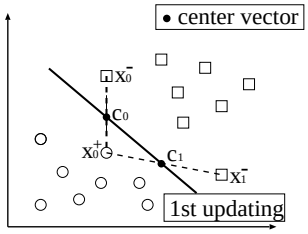

Figure 7: First updating of HP

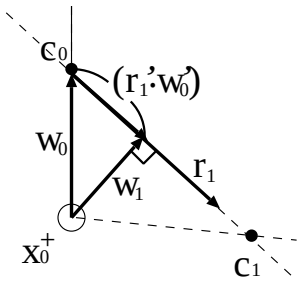

Figure 8: Detail of updating

times, the hyperplane must go through n centers and no further rotation is possible. For the m-th updating, we find the most violating data with negative targets. And we set the data to $\mathbf{x}_m^-$. Then, $\mathbf{c}_m$ is the center of $(\mathbf{x}_0^+, \mathbf{x}_m^-)$, $\mathbf{r}_m$ is given by $\mathbf{r}_m = \mathbf{c}_m - \mathbf{c}_0$.

The hyperplane with $\mathbf{w}_{m-1}$ passes through the vector $\{\mathbf{r}_1, ..., \mathbf{r}_{m-1}\}$. Let the vectors $\{\mathbf{p}_1, ..., \mathbf{p}_{m-1}\}$ be the orthogonal system of $\{\mathbf{r}_1, ..., \mathbf{r}_{m-1}\}$. Obviously, the hyperplane with $\mathbf{w}_{m-1}$ passes through $\{\mathbf{p}_1, ..., \mathbf{p}_{m-1}\}$.

We use the Gram-Schmidt orthogonalization to calculate $\mathbf{p}_m$. Then, $\mathbf{p}_m$ is given by

$$\mathbf{p}_m = \mathbf{r}'_m - \sum_{k=1}^{m-1} (\mathbf{r}'_m \cdot \mathbf{p}'_k)\mathbf{p}'_k, \tag{4}$$

where $\mathbf{a}'$ is the normalized vector of $\mathbf{a}$. The weight vector $\mathbf{w}_{m-1}$ of the hyperplane is determined so that it is orthogonal to the orthogonal system $\{\mathbf{p}_1, ..., \mathbf{p}_{m-1}\}$. For the m-th update, it is updated so that it is orthogonal to $\mathbf{p}_m$. Namely, $\mathbf{w}_m$ is obtained by rotating the hyperplane in the direction of $\mathbf{p}_m$ with $\mathbf{c}_0$ as the center:

$$\mathbf{w}_m = \mathbf{w}'_{m-1} - \left(\mathbf{w}'_{m-1} \cdot \mathbf{p}'_m\right) \mathbf{p}'_m. \tag{5}$$

The updated hyperplane passes through the vector $\{\mathbf{r}_1, ..., \mathbf{r}_m\}$.

Next, we show the updating method in 2-dimensional space using Fig. 6, which shows the initial hyperplane. Since there are data that violate the CARVE criterion, we rotate the hyperplane as shown in Fig. 7. In this case, $\mathbf{r}_1$ is given by $\mathbf{r}_1 = \mathbf{c}_1 - \mathbf{c}_0$ (Fig. 8). We calculate the weight vector $\mathbf{w}_1$ that is orthogonal to $\mathbf{r}_1$:

$$\mathbf{w}_1 = \mathbf{w}'_0 - \left(\mathbf{w}'_0 \cdot \mathbf{p}'_1\right) \mathbf{p}'_1, \tag{6}$$

where $\mathbf{p}_1$ is the orthogonal system of $\{\mathbf{r}_1\}$ in the 2-dimensional space, and is expressed as $\mathbf{p}_1 = \mathbf{r}_1$, because the orthogonal system $\{\mathbf{r}_1\}$ has only one element.

3.2. Learning algorithm

Step 1 To determine the initial support vectors $(\mathbf{x}_0^+, \mathbf{x}_0^-)$, we find the data pair of opposite targets with the nearest distance among all data pairs of opposite targets. The center $\mathbf{c}_0$ and the weight $\mathbf{w}_0$ are given by

$$\mathbf{c}_0 = \frac{1}{2}(\mathbf{x}_0^+ + \mathbf{x}_0^-), \tag{7}$$

$$\mathbf{w}_0 = \mathbf{x}_0^+ - \mathbf{x}_0^-. \tag{8}$$

Step 2 We determine the decision function:

$$f(\mathbf{x}_i) = \mathbf{w}_m \cdot \mathbf{x}_i - \mathbf{w}_m \cdot \mathbf{c}_0 \ \text{ for } i = 1, ..., l, \tag{9}$$

where $\mathbf{w}_m \cdot \mathbf{c}_0$ is the bias term, and l is the number of remaining data.

Step 3 If all the data with negative targets satisfy

$$y_n f(\mathbf{x}_n^-) \geq C_h f(\mathbf{x}_0^+) \ \text{ for } n = 1, ..., M, \tag{10}$$

we consider that the hyperplane that satisfies the CARVE condition is found, and finish updating. Here, $\mathbf{x}_n^-$ is the data with negative target, M is the number of data with negative targets, and C_h accomplishes the role of soft-margin. Usually, C_h is set to $C_h < 1$. If C_h is negative, the data with negative target may exist in the positive side of the hyperplane. Thus for negative C_h, the CARVE condition is violated.

Step 4 In the m-th updating, we include the most violated data $\mathbf{x}_m^-$ in the support vectors. Then, we calculate $\mathbf{c}_m = \frac{1}{2}(\mathbf{x}_0^+ + \mathbf{x}_m^-)$, and calculate $\mathbf{r}_m = \mathbf{c}_m - \mathbf{c}_0$. From $\{\mathbf{r}_1, ..., \mathbf{r}_m\}$, the m-th component of the orthogonal system, $\mathbf{p}_m$, is

$$\mathbf{p}_m = \mathbf{r}'_m - \sum_{k=1}^{m-1} (\mathbf{r}'_m \cdot \mathbf{p}'_k)\mathbf{p}'_k. \tag{11}$$

The weight vector $\mathbf{w}_m$ is written as follows:

$$\mathbf{w}_m = \mathbf{w}'_{m-1} - \left(\mathbf{w}'_{m-1} \cdot \mathbf{p}'_m\right) \mathbf{p}'_m. \tag{12}$$

We can obtain the orthogonal vector $\mathbf{w}_m$ for hyperplane that pass through $\mathbf{c}_0, \mathbf{c}_1, ..., \mathbf{c}_m$ by updating in this way.

Table 1: Feature of benchmark data

Data	Inputs	Classes	Train.	Test
Blood cell	13	12	3097	3100
Thyroid	21	3	3772	3428
Hiragana-50	50	39	4610	4610

Table 2: Performance for blood cell training data by MM-NN

C_h	Hidden Units	Time [s]	Train. [%]	Test [%]
-1.5	65	66	96.90	92.32
-1.0	95	122	97.64	92.48
-0.5	151	242	98.48	93.39
0.0	320	1776	99.23	92.55

Step 5 If updating was repeated n or M times return to Step 1 to renew the initial hyperplane. Otherwise, return to Step 2.

4. DETERMINATION OF OUTPUT LAYER HYPERPLANES

We determine the output layer hyperplanes by using the same technique that trains SVMs. The hyperplane is obtained by solving the following dual problem.

$$\text{maximize } Q(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \mathbf{z}_i^t \mathbf{z}_j \quad (13)$$

$$\text{subject to } \sum_{i=1}^{k} y_i \alpha_i = 0, \quad 0 \le \alpha_i \le C_o \quad (14)$$

where $\alpha = (\alpha_1, ..., \alpha_l)^t$ is the Lagrange multiplier, C_o is the margin parameter for the output layer, and $\mathbf{z}_i$ is the output vector of the hidden units.

5. PERFORMANCE EVALUATION

We evaluated the performance improvement of the proposed method over the support vector machine [6] and the three-layer neural network using the blood cell data [4], thyroid data,[1] and the hiragana data [5] listed in Table 1. We used a Pentium III 1GHz PC to measure the recognition rates of the training and test data.

[1] ftp://ftp.ics.uci.edu/pub/machine-learning-databases/

Table 3: Performance comparison for blood cell data

Classifier	Parm	Train. [%]	Test [%]
MM-NN	$C_h = -0.5$	98.48	93.39
SVM	$d = 3$	98.22	93.26
BP	Epochs = 15000	95.61	91.42

Blood cell data. The blood cell classification involves classifying optically screened white blood cells into 12 classes using 13 features. This is a very difficult problem; class boundaries for some classes are ambiguous since the classes are defined according to the growth stages of white blood cells.

In this simulation, we set the output layer margin parameter $C_o = 100$. We determined the slope of the sigmoid function T so that the output of the hidden (output) unit is $\varepsilon(2) = 0.6$ ($\varepsilon(3) = 0.8$) for the support vector with the target value of 1. Here, ε is given by

$$\varepsilon = \frac{1}{1 + \exp(-x^+/T)}, \quad (15)$$

where x^+ is the neuron input associated with the support vector with the target value of 1.

Table 2 shows the results of the blood cell data for the proposed method (MM-NN). Here, we show the number of hidden units, training time, the recognition rates of training data, and that of test data when we change the value of the hidden layer margin parameter C_h. Negative C_h means that hidden layer hyperplanes might not satisfy the CARVE condition. The best recognition rate of the test data was achieved for $C_h = -0.5$. For $C_h = 0.0$, the recognition rate of the training data is less than 100%. This was caused by low value of $\varepsilon(2)$.

Table 3 shows the comparison among the proposed method, conventional SVM, and the three-layer neural network trained by BP. Parm shows C_h in MM-NN, the kernel parameter in SVM and the number of epochs in BP. For SVM, we use the following polynomial kernel function:

$$H(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^t \mathbf{x}' + 1)^d.$$

For BP, we show the maximum recognition rate for 10 experiments changing the initial weights. From the table, MM-NN and SVM show the comparable recognition rates of the test data but BP shows a lower recognition rate.

Thyroid data. The thyroid data include 15 digital features and more than 92% of the data belong to one class. Thus the recognition rate smaller than 92% is useless. Table 4 shows the results for the thyroid data for the proposed method. In the simulation, we set the output layer margin parameter $C_o = 1000$, the support vector output of hidden

Table 4: Performance for thyroid data by MM-NN

C_h	Hidden Units	Time [s]	Train. [%]	Test [%]
-1.0	22	19	99.10	98.10
-0.5	49	28	99.07	98.42
0.0	95	84	100	98.13

Table 5: Performance comparison for thyroid data

Classifier	Parm	Train. [%]	Test [%]
MM-NN	$C_h = -0.5$	99.07	98.42
SVM	$d = 3$	99.26	97.55
BP	Epochs= 4000	99.15	97.93

layer units $\varepsilon(2) = 0.7$, and the support vector output of output layer units $\varepsilon(3) = 0.8$. For $C_h = -0.5$ the maximum recognition of the test data was obtained.

Table 5 shows performance comparison for the thyroid data. The recognition rates of the test data for the three classifiers are comparable, but MM-NN shows the best recognition rate.

Hiragana-50 data. Hiragana-50 data were gathered from Japanese license plates. The original gray-scale images of hiragana characters were transformed into 5×10-pixel with the gray-scale range being from 0 to 255. Then by performing gray-scale shift, position shift, random noise addition to the images, the training and test data were generated. Table 6 shows the results for the hiragana data for the proposed method. We showed the number of hidden units, training time, the recognition rates of training data, and those of test data when we change the value of hidden layer margin parameter C_h. Also in this simulation, we set the output layer margin parameter $C_o = 1000$, the support vector output of hidden layer units $\varepsilon(2) = 0.7$, and the support vector output of output layer units $\varepsilon(3) = 0.8$. For $C_h = 0.1, -0.1$ the maximum recognition of the test data was obtained.

Table 7 shows performance comparison for the hiragana-50 data. Both MM-NN and SVM show comparable recognition rates of the test data, while the BP shows inferior results, which is caused by the fact that BP does not have a mechanism to maximize margins.

6. CONCLUSIONS

In this paper we proposed a new method for training neural networks based on the CARVE algorithm. To determine the hidden layer hyperplane we extended the DirectSVM so that the trained hyperplane satisfies the condition that the

Table 6: Performance for hiragana-50 data by MM-NN

C_h	Hidden Units	Time [s]	Train. [%]	Test [%]
0.2	356	945	100	99.15
0.1	341	889	100	99.07
0.0	325	848	100	98.96
-0.1	308	756	100	99.15
-0.2	280	629	100	99.00

Table 7: Performance comparison for hiragana-50 data

Classifier	Parm	Train. [%]	Test [%]
MM-NN	$C_h = -0.1$	100	99.15
SVM	$d = 5$	100	99.46
BP	Epochs= 10000	98.92	95.77

CARVE algorithm is applicable. We determine the output layer hyperplane by the quadratic programming technique. Computer simulations using some benchmark data sets showed that the generalization ability of the proposed method is comparable with that of the conventional support vector machines and superior to that of three-layer neural networks trained by BP.

7. REFERENCES

[1] S. Young and T. Downs, "CARVE - A constructive algorithm for real-valued examples," *IEEE Trans. Neural Networks*, Vol. 9, No. 6, pp. 1180–1190, 1992.

[2] S. Abe, *Pattern Classification: Neuro-fuzzy Methods and Their Comparison*, Springer-Verlag, 2001.

[3] D. Roobaert, "DirectSVM - A fast and simple support vector machine perceptron," *Proc. 2000 IEEE Signal Processing Society Workshop*, pp. 356–365, 2000.

[4] Hashizume, J. Motoike and R. Yabe, "Fully automated blood cell differential system and its application," *Proc. IUPAC 3rd International Congress on Automation and New Technology in the Clinical Laboratory*, pp. 297–302, Kobe, Japan, 1988.

[5] H. Takenaga et al., "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, 1991.

[6] T. Inoue and S. Abe, "Fuzzy support vector machines for pattern classification," *Proc. IJCNN'01*, pp. 1449–1454, 2001.