Training Three-layer Neural Network Classifiers by Solving Inequalities

Naoki Tsuchiya, Seiichi Ozawa, and 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 discuss training of three-layer neural network classifiers by solving inequalities. Namely, first we represent each class by the center of the training data belonging to the class, and determine the set of hyperplanes that separate each class into a single region. Then according to whether the center is on the positive or negative side of the hyperplane, we determine the target values of each class for the hidden neurons. Since the convergence condition of the neural network classifier is now represented by the two sets of inequalities, we solve the sets successively by the Ho-Kashyap algorithm. We demonstrate the advantage of our method over the BP using three benchmark data sets.

1 Introduction

Multi-layer neural network classifiers are widely used in many fields, but since training by the back-propagation algorithm (BP) is slow, a number of training methods have been proposed [1]. They are classified into acceleration within the framework of the BP, acceleration using the optimization methods, and acceleration based on the synthesis principle of the neural network classifier [2].

In [2], first a multilayer neural network classifier is trained by the BP. Then the separation hyperplanes are extracted from the weights between the input and hidden neurons, and using these weights the target values of the hidden neuron outputs are determined. Finally, the weights between the inputs and hidden neurons and the weights between the hidden and output neurons are tuned separately. In [3], using the Ho-Kashayp algorithm, two-layer neural networks are trained.

In this paper, we trained the network based on the synthesis principle, assuming that each class is separated into a single region by hyperplanes. First we represent each class by the center of the training data belonging to the class, and determine the set of hyperplanes that separate each class into a single region. Then according to whether the center is on the positive or negative side of the hyperplane, we determine the target values of each class for the hidden neurons. Since the convergence condition of the neural network classifier is now represented by the two sets of inequalities, we solve the sets successively by the Ho-Kashyap algorithm. In the following we discuss the details of the training method and compare the performance of our method with that of the BP.

2 Training by Solving Inequalities

Figure 1 shows the structure of a three-layer neural network. In the figure, the neurons between the input and hidden layers, and the hidden and output layers are completely connected by weights. Let $x_i(k)$ and $z_i(k)$ be the ith input and output variables of the kth layer, respectively. And $w_{ij}(k)$ is the weight between the ith neuron of the kth layer and the jth neuron of the $(k+1)$st layer. Then the ith input and output of the kth layer are given, respectively, by

$$x_i(k+1) = \mathbf{w}_i(k)^t \cdot \mathbf{z}(k), \tag{1}$$

$$z_i(k+1) = \frac{1}{1+\exp(-x_i(k+1))}, \tag{2}$$

where for $i = 1, \ldots, n(k+1)$, $\mathbf{x}(k) = [x_1(k), \ldots, x_{n(k)}(k)]^t$, $\mathbf{z}(k) = [z_1(k), \ldots, z_{n(k)+1}(k)]^t$, $\mathbf{w}_i(k) = [w_{1i}(k), \ldots, w_{n(k)+1,i}(k)]^t$ and $n(k)$ is the number of neurons for the kth layer.

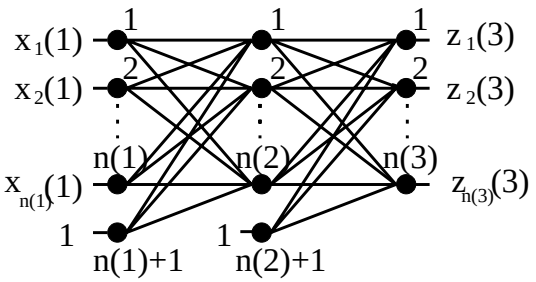

Figure 1 Structure of a three-layer neural network.

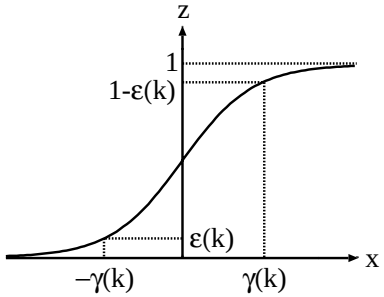

Figure 2 Sigmoid function.

If we can assume that each class is separated by hyperplanes, from other classes, into a single region, a three-layer (one-hidden layer) neural network classifier can be synthesized [1]. Thus if we can determine the hyperplanes that separate each class, we can set the target values of the hidden neuron outputs. Then we can determine the weights between the input and hidden layers and the weights between the hidden and output layers separately.

In the following, assuming that each class is separated into a single region, we determine the target values of hidden neuron outputs, derive the two sets of inequalities that the three-layer neural network satisfies, and solve the sets by the Ho-Kashyap algorithm.

2.1 Setting Target Values

We set the target values of hidden neuron outputs as follows. First, we calculate the center vector of each class using the training data included in the class, and we determine the separation hyperplane between two classes by the hyperplane that includes the middle point of the center vectors of the two classes and that is orthogonal to the line segment connecting these center vectors. Here, we define a positive δ and if the distance from the center vector other than the two center vectors to the hyperplane is larger than δ, we assume that the associated class is also separated by the hyperplane, and iterate the above procedure until all the classes are separated by the separation hyperplanes. In the following we show the algorithm of setting target values.

1. Calculate the center vector of class $i (i = 1, \ldots, n(3))$, $\mathbf{c}_i$, using the training data included in the class. Set $i = 1, j = 2$.
2. For two classes $i, j (i < j)$, let the vector connecting $\mathbf{c}_j$ and $\mathbf{c}_i$ be $\mathbf{n}_{ij} = \mathbf{c}_j - \mathbf{c}_i$, and determine the separation hyperplane p_{ij} that is parallel to $\mathbf{n}_{ij}$ and that includes the middle point of the two center vectors by
$$\left(\mathbf{x} - \frac{1}{2}(\mathbf{c}_j + \mathbf{c}_i)\right)^t \mathbf{n}_{ij} = 0, \tag{3}$$
where the regions in which the classes i and j exist are on the negative and positive sides of the hyperplane, respectively.
3. If the distance between the center vector $\mathbf{c}_l (l \neq i, j, l = 1, \ldots, n(3))$ for class l and the kth separation hyperplane p_{ij}, d, is less than δ:
$$d = \frac{\left|(\mathbf{c}_l - \frac{1}{2}(\mathbf{c}_j + \mathbf{c}_i))^t \mathbf{n}_{ij}\right|}{|\mathbf{n}_{ij}|} \leq \delta, \tag{4}$$
we assume that class l is not separable by the hyperplane p_{ij} and set the target value of class l for the kth hyperplane p_{ij}, i.e., that of the kth hidden neuron output, $s_k(2)$, is dc where dc means either 1 or 0. When the distance is larger than or equal to δ, $s_k(2)$ is set to 1 when class l is on the positive side of the hyperplane and 0 otherwise.
4. If the number of center vectors in the region separated by the hyperplanes is equal to or less than 1, go to step 5. If there are more than two center vectors, select two center vectors, let the associated classes be i and $j (i < j)$, and go to step 3.
5. Consider that dc takes on both 0 and 1 and check if different classes have the same target vector. If there are the same target vectors, generate the separation hyperplane p_{ij} for the classes associated with the target vectors in the similar way discussed in steps 2 and 3. If there are no such classes, terminate the algorithm.
6. Calculate the distances between the generated hyperplane and the center vectors and generate the target values. If the target vectors for all the classes are different considering dc takes on both 1 and 0, terminate the algorithm. If not, go to step 5.

The target value of the output for the ith hidden neuron is set to 1 for the training data belonging to class i and 0 for the remaining training data. Namely,
$$s_j(3) = \begin{cases} 1 & \text{for } j = i, \\ 0 & \text{for } j \neq i. \end{cases} \tag{5}$$

2.2 Defining Inequalities

Since we have determine the target values of the hidden neuron outputs for each class, we can describe the conditions that the weights between the $(k-1)$st $(k=2,3)$ layer neurons and kth layer neurons must satisfy.

For the weights connecting from the $(k-1)$st layer neurons to the ith neuron of the kth layer, the following inequalities must hold for all the training data:

$$|z_i(k) - s_i(k)| \leq \varepsilon(k) \qquad \text{for} \quad s_i(k) = 0,\ 1, \tag{6}$$

where $\varepsilon(k)$ denotes the tolerance of convergence for the kth layer neuron output. When $k=2$ and the value of $s_i(2)$ is dc, the above inequalities need not hold.

Using (2) (see Fig. 2), we can convert (6) that specifies the outputs of the kth layer neurons into the specifications of inputs of the kth layer neurons:

$$\begin{aligned} \mathbf{z}^t(k-1)\mathbf{w}_i(k-1) \leq \quad & -\gamma(k) \qquad \text{for} \quad s_i(k) = 0, \\ \mathbf{z}^t(k-1)\mathbf{w}_i(k-1) > \quad & \gamma(k) \qquad \text{for} \quad s_i(k) = 1, \end{aligned} \tag{7}$$

where $\gamma(k)$ is given by

$$\gamma(k) = \log\left(\frac{1}{\varepsilon(k)} - 1\right). \tag{8}$$

2.3 Solving Inequalities

Now we can determine the weights between the input and hidden neurons and the weights between the hidden and output neurons, solving (7) with $k=2$ first, and then (7) with $k=3$.

To solve the inequalities, we apply the Ho-Kashyap algorithm [3, 4]. In this algorithm to solve $Y\mathbf{a} > 0$ for $\mathbf{a}$, where Y is an arbitrary matrix, a positive margin vector $\mathbf{b}$ is introduced to convert the original inequality to the equation $Y\mathbf{a} = \mathbf{b}$. Then $\mathbf{a}$ and $\mathbf{b}$ are corrected iteratively so that the squared error $(\mathbf{b} - Y\mathbf{a})^t(\mathbf{b} - Y\mathbf{a})$ is minimized. By correcting $\mathbf{b}$ so that the elements of $\mathbf{b}$ are non-decreasing, the obtained $\mathbf{a}$ is guaranteed to satisfy $Y\mathbf{a} > \mathbf{0}$. Because of the space limitation, we skip the details of the algorithm.

To obtain the weights $\mathbf{w}_i(k)$ $(k=1,2)$ that satisfy (7), we do the following.

For the lth $(l=1,\ldots,M)$ training data, we generate $Y_{li}(i=1,\ldots,n(k)+1)$ by

$$Y_{li} = \begin{cases} z_i(k) & \text{for } s_i(k+1) = 1, \\ -z_i(k) & \text{for } s_i(k+1) = 0, \end{cases} \tag{9}$$

and the initial vector of $\mathbf{b}$ by

$$b_l = \gamma(k+1). \tag{10}$$

If the target value of the hidden neuron output is dc, we do not apply the above procedure. Using the solution $\mathbf{a}$ obtained by applying Y and $\mathbf{b}$ to the Ho-Kashyap algorithm, we obtain

$$w_{ji}(k) = a_j \qquad \text{for} \ \ j = 1,\ldots,n(k)+1. \tag{11}$$

By applying the above procedure to all the neurons in the hidden and output neurons, we can determine the weights. To accelerate solving (7) with $k=3$, we delete the training data that do not satisfy (7) with $k=2$. Notice that when we obtain the weights between the input and hidden layers, $\mathbf{w}_i(1)$, we calculate $z_i(2)$ using $\mathbf{w}_i(1)$ and then generate Y_{li}. By this, the exact values of the targets that are determined roughly as discussed in **2.1** are determined.

3 Performance Evaluation

We compared the training time and the recognition rate of our method with those of the BP using blood cell data [5], thyroid data [6], and hiragana data [7]. Table 1 lists the numbers of inputs, classes, training data, and test data for the three data sets. We used IBM Think Pad 600E 3JJ (PentiumII 300MHZ, 64MB memory).

Table 2 lists the numbers of hidden neurons for the BP and the proposed method, and the maximum number of epochs for the BP. The number of hidden neurons were set so that the recognition rates for the test data were maximized.

We trained the network with the BP three times changing the initial weights with $\varepsilon = 0.01$ and the learning rate being 1. In training by the BP, if the output values were within the tolerance of convergence or if the number of epochs exceeded the maximum number, we stopped training.

In the proposed method, we set $\varepsilon(3) = 0.01$ which is the same as that of the BP and changed $\varepsilon(2)$ to evaluate the performance.

Table 1: Benchmark data specifications.

Data	Input	Class	Train.	Test
Blood Cell	13	12	3097	3100
Thyroid	21	3	3772	3428
Hiragana	50	39	4610	4610

Table 2: Training conditions.

Data	BP	Proposed	Epochs
Blood Cell	18	12	15000
Thyroid	3	3	10000
Hiragana	25	35	10000

Table 3 lists the results for the proposed method. Numerals in brackets in the "Rate" column show the recognition rates for the training data.

In general when we set a larger value to $\varepsilon(2)$, the solution space becomes larger. Thus the inequalities are more easily solved. Seen from the table, the recognition rates for the blood cell data and the hiragana data show this tendency but those for the thyroid data show the opposite.

Table 3: Performance for three data sets by the proposed method.

(a) Blood cell data

$\varepsilon(2)$	Rate [%]	Time [m]
0.70	91.39(93.83)	4
0.75	91.65(93.80)	4
0.80	91.06(93.54)	4
0.85	91.22(92.57)	4
0.90	90.38(91.96)	4
0.95	89.54(91.24)	4

(b) Thyroid data

$\varepsilon(2)$	Rate [%]	Time [m]
0.70	96.39(97.27)	1
0.75	96.73(97.77)	2
0.80	97.17(98.17)	2
0.85	97.40(98.32)	2
0.90	97.37(98.49)	3
0.95	97.43(98.44)	3

(c) Hiragana data

$\varepsilon(2)$	Rate [%]	Time [m]
0.70	95.12(98.22)	18
0.75	94.81(98.37)	18
0.80	94.07(98.35)	18
0.85	93.93(98.37)	18
0.90	93.62(98.00)	18
0.95	93.30(97.80)	18

Table 4 shows recognition performance and the training times for the BP and the proposed method. The recognition rates and training times for the BP are the averages of three runs. From the table it is evident that our method is faster than the BP and shows better or comparable recognition performance with that of the BP.

4 Conclusion

In this paper, we proposed to train neural network classifiers by solving inequalities. First, we represent each class region by the center of the training data belonging to the class, and determine the set of hyperplanes that separate each class into a single region. By these hyperplanes we set the target values of the hidden neuron outputs and formulate the two sets of inequalities that the weights must satisfy. By solving the sets by the Ho-Kashyap algorithm, training was speeded-up and recognition performance was comparable or better than that by the BP.

Table 4: Performance comparison.

Data	BP		Proposed		Speed-up
	Rate [%]	Time [m]	Rate [%]	Time [m]	
Blood Cell	88.88(94.22)	85	91.65(93.80)	4	21
Thyroid	97.66(98.75)	7	97.43(98.44)	3	2.3
Hiragana	95.07(98.22)	206	95.12(98.22)	18	32

Acknowledgments

We are grateful to Professor N. Matsuda of Kawasaki Medical School for providing the blood cell data and to Mr. P. M. Murphy and Mr. D. W. Aha of the University of California at Irvine for organizing the data bases including the thyroid data (ics.uci.edu: pub/machine-learning-databases).

References

[1] S. Abe, *Neural Networks and Fuzzy Systems: Theory and Applications*, Kluwer Academic, Boston, 1996.

[2] S. Abe and M. Kayama and H. Takenaga, "Acceleration of Learning and Improvement of Generalization Ability for Pattern Classification Networks," *Trans. IEICE of Japan D-II*, Vol. J76-DII, No. 3, pp. 647–652, 1993 (in Japanese).

[3] M. H. Hassoun and J. Song, Adaptive Ho-Kashyap Rules for Perceptron Training, *IEEE Transactions on Neural Networks*, Vol. 3, No. 1, pp. 51–61, 1992.

[4] R. O. Duda and P. E. Hart, "The Ho-Kashyap Procedures," in *Pattern Classification and Scene Analysis*, pp. 159–166, John Wiley & Sons, 1973.

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

[6] S. M. Weiss and I. Kapouleas, "An Empirical Comparison of Pattern Recognition Neural Nets and Machine Learning Classification Methods, *Proc. IJCAI–89*, pp. 781–787, 1989.

[7] H. Takenaga, S. Abe, M. Takatoo, M. Kayama, T. Kitamura, and Y. Okuyama, "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.