IEICE TRANS. FUNDAMENTALS, VOL. E81–A, NO. 3 MARCH 1998

PAPER Special Section of Selected Papers from the 10th Karuizawa Workshop on Circuits and Systems

Training Data Selection Method for Generalization by Multilayer Neural Networks

Kazuyuki HARA† and Kenji NAKAYAMA††, *Members*

SUMMARY A training data selection method is proposed for multilayer neural networks (MLNNs). This method selects a small number of the training data, which guarantee both generalization and fast training of the MLNNs applied to pattern classification. The generalization will be satisfied using the data locate close to the boundary of the pattern classes. However, if these data are only used in the training, convergence is slow. This phenomenon is analyzed in this paper. Therefore, in the proposed method, the MLNN is first trained using some number of the data, which are randomly selected (Step 1). The data, for which the output error is relatively large, are selected. Furthermore, they are paired with the nearest data belong to the different class. The newly selected data are further paired with the nearest data. Finally, pairs of the data, which locate close to the boundary, can be found. Using these pairs of the data, the MLNNs are further trained (Step 2). Since, there are some variations to combine Steps 1 and 2, the proposed method can be applied to both off-line and on-line training. The proposed method can reduce the number of the training data, at the same time, can hasten the training. Usefulness is confirmed through computer simulation.

key words: *training data selection, generalization performance, multilayer neural networks*

1. Introduction

Recently, multilayer neural networks (MLNNs) [1] have been used in pattern classification and signal processing field [2]–[5]. In these applications, generalization is important. A huge amount of the training data may guarantee generalization of the MLNNs. On the other hand, it will require a long training time and a large data memory. Therefore, it is desirable to reduce the number of the training data while maintaining generalization.

This kind of approach has been discussed upto now. Battiti [6] used the mutual information to evaluate the input feature to limit the input dimensionality. This approach is to reduce the complexity of the network.

The training time can be reduced by reducing the number of hidden units. Many papers have been presented for this subject [7]–[9]. In statistical approaches, Fukumizu and Watanabe [10] proposed a training data selection method by using error analysis for the function approximation. Cachin [11] proposed the error-dependent repetition. Presentation probability of the training data is controlled to be proportional to the MLNN output error. However, the entire data are used in the training process.

In this paper, we propose a training data selection method for the MLNNs applied to pattern classification. The purpose of this method is to select a small number of the training data, with which both generalization and fast training are guaranteed. The selected data can locate around the decision boundary between the pattern (or data) classes. In this paper, we assume the following data distribution, that is the data distributions of the different classes are separated. This means they do not overlap to each other. This method can be applied to reduce in data memory and computations of off-line training, where a sufficient number of training data are obtained in advance. Furthermore, it will be useful for on-line training, where all training data cannot obtain at the beginning, rather they are gradually increased.

Efficiency of the proposed method is investigated through computer simulations. The back propagation algorithm [1] is used to train the MLNN. Two kinds of problems are employed as examples.

2. Multilayer Neural Network

In this paper, a two-layer MLNN is used to classify the data. N samples of a piece of data, that is the input vector $\boldsymbol{x} = \{x_i, i = 1 \sim N\}$, is applied to the input layer. The ith input unit receives x_i. The connection weight from the ith input to the jth hidden unit is denoted w_{ij}. The input potential net_j and the output y_j of the jth hidden unit are given by

$$net_j = \sum_{i=1}^{N} w_{ij} x_i + \theta_j \tag{1}$$

$$y_j = f_H(net_j) \tag{2}$$

$$f_H(net_j) = \frac{1 - e^{-net_j}}{1 + e^{-net_j}} \tag{3}$$

where, $f_H(\cdot)$ is the activation function in the hidden layer and θ_j is a bias. The input potential net_k and the output y_k of the kth output unit are given by

Manuscript received June 30, 1997.
Manuscript revised October 1, 1997.

†The author is with the Department of Computer Science, Gunma Polytechnic College, Takasaki-shi, 370–1213 Japan.
††The author is with the Department of Electrical and Computer Engineering, Faculty of Engineering, Kanazawa University, Kanazawa-shi, 920–0942 Japan.

$$net_k = \sum_{j=1}^{J} w_{jk} y_j + \theta_k \tag{4}$$

$$y_k = f_O(net_k) \tag{5}$$

$$f_O(net_k) = \frac{1}{1 + e^{-net_k}} \tag{6}$$

where $f_O(\cdot)$ is the activation function in the output layer.

The number of output units is equal to that of the pattern or data classes. The MLNN is trained so that a single output unit responds to one of the pattern or data classes.

3. Geometrical Relation between Input and Output

The input space can be separated into two regions by a hyperplane formed by $net_j = 0$ in Eq. (1). A distance between this hyperplane and the input vector $\boldsymbol{x}$ is given by

$$d_j = \frac{\left| \sum_{i=1}^{N} w_{ij} x_i + \theta_j \right|}{\|\boldsymbol{w}_j\|} = \frac{|\boldsymbol{net}_j|}{\|\boldsymbol{w}_j\|}, \tag{7}$$

$$\boldsymbol{w}_j = \{w_{ij}, i = 1 \sim N\}. \tag{8}$$

where $\|\boldsymbol{w}_j\|$ is an L_2 norm of the weight vector $\boldsymbol{w}_j$. Since $\|\boldsymbol{w}_j\|$ is independent of the input $\boldsymbol{x}$, the input potential net_j is proportional to the distance d_j. The activation function Eq. (3) is a monotonically increasing function, then the hidden unit output y_j is also monotonically increasing with respect to the distance d_j. However, y_j is not a linear function of the distance.

The output of the output unit y_k is separated by the ranges of $y_k > 0.5$ and $y_k < 0.5$. The input potential $net_k = 0$ provides a decision boundary. This is called "a network boundary" in this paper. On the other hand, in order to express the boundary between the data belong to the different classes, a term "class boundary" is used. If the training converges, the network boundary will agree with the class boundary. Then a distance from the class boundary to the input data is related to $|y_k - 0.5|$. In this case, net_k is also related to the distance.

In conclusion, $|y_k - 0.5|$ and $|net_k|$ are monotone functions with respect to the distance between the data boundary and the input data.

4. Pairing Method for Training Data Selection

4.1 Pairing Method

In this paper, two classes $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are taken into account for convenience. However, the proposed method can be applied to more than two classes.

In this section, a pairing method is first described. In the pairing process, pairs data of the different classes, whose Euclidean distance is minimum, are selected. Let $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ be sets of two data classes, and $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ be elements of them. $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are paired with each other through the following steps.

Step 1: Select $\boldsymbol{x}_1$ (or $\boldsymbol{x}_2$) from $\boldsymbol{X}_1$ (or $\boldsymbol{X}_2$) randomly.

Step 2: Select $\boldsymbol{x}_2^p$ (or $\boldsymbol{x}_1^p$) from $\boldsymbol{X}_2$ (or $\boldsymbol{X}_1$), which has the shortest distance to the $\boldsymbol{x}_1$ (or $\boldsymbol{x}_2$), selected in Step 1.

Step 3: Select $\boldsymbol{x}_1^{p'}$ (or $\boldsymbol{x}_2^{p'}$) from $\boldsymbol{X}_1$ (or $\boldsymbol{X}_2$), which has the shortest distance to $\boldsymbol{x}_2^p$ (or $\boldsymbol{x}_1^p$), selected in Step 2.

When all the data are selected from $\boldsymbol{X}_1$ (or $\boldsymbol{X}_2$) in Step 1, the pairing process is completed. Otherwise, return to Step 1, and repeat the above process. In this process, the same data will not be selected. Finally, the data x_1^p (or $x_2^{p'}$) and x_2^p (or $x_1^{p'}$), selected based on the distance, are included in the reduced data set.

In this paper, it is assumed that the data distributions of the different classes do not overlap. Therefore, the class boundary locates between the nearest data of the different classes, facing each other. Thus, the data locate close to the boundary can be selected by this method.

4.2 Training Using Data Selected by Pairing Method

The data selected by the pairing method are gathers of the pairs of the neighbor data. The training using these selected data takes a long time for the following two reasons.

First, to classify the similar data into the different classes, the network output must be sensitive to the change of the input. In this case, the slope of the sigmoid function must be steep. To do so, the absolute value of the connection weights must be large. If the initial connection weights are small and net_j distributed within a narrow range in a linear part of the sigmoid function, this requires many iterations.

Second, the connection weights are not properly modified to the selected data. Consider $\boldsymbol{x}_{1m}^p \in \boldsymbol{X}_1$ and $\boldsymbol{x}_{2m'}^p \in \boldsymbol{X}_2$ as the selected data.

(1) When $\|\boldsymbol{x}_{1m}^p - \boldsymbol{x}_{2m'}^p\| \ll 1$ is held.

If the network boundary is far from $\boldsymbol{x}_{1m}^p$ and $\boldsymbol{x}_{2m'}^p$, the output of the output unit for these data satisfy $y_{1m} \cong y_{2m'} \cong 1$ or 0. In this case, the connection weights are adjusted properly. So, the network boundary approaches to $\boldsymbol{x}_{1m}^p$ and $\boldsymbol{x}_{2m'}^p$. At the same time, y_{1m} and $y_{2m'}$ approach to 0.5. In this case, as shown in the next paragraph, the amount of the correction decreases. Especially, when the network boundary exceeds $\boldsymbol{x}_{1m}^p$ (or $\boldsymbol{x}_{2m'}^p$) and enters between them, the amount of the correction weight becomes very small.

This can be proven as follows: We assume $y_{1m} \cong y_{2m'} \cong 0.5$. The correction of the connection weight for

the mth pattern Δ_m is defined as follows:

$$\Delta_m = \eta \delta_m o_{mj} \tag{9}$$

$$\delta_m = (t_m - y_m) f'(net_m) \tag{10}$$

here o_{mj} is the jth hidden output. Since the selected two data are similar, if $o_{1mj} = o_{2m'j}$ and $f'_O(net_{1m}) = f'_O(net_{2m'})$ are satisfied, the correction of the connection weight is

$$\Delta_{1m} + \Delta_{2m'} = \eta o_{1mj}\{(t_{1m} - y_{1m}) + (t_{2m'} - y_{2m'})\} \tag{11}$$

Suppose the targets are $t_{1m} = 1, t_{2m'} = 0$. For $y_{1m} = y_{2m'} = 0.5$, $(t_{1m} - y_{1m}) + (t_{2m'} - y_{2m'}) = (1 - 0.5) + (0 - 0.5) = 0$. The total weight correction for both $\boldsymbol{x}^p_{1m}$ and $\boldsymbol{x}^p_{2m'}$ becomes very small. Thus, the convergence of the training becomes very slow.

(2) When $\|\boldsymbol{x}_{1m} - \boldsymbol{x}_{2m'}\| \ll 1$ is not held.

In this case, if the network boundary approaches to the data, $y_{1m} \cong 0.5$ and $y_{2m'} \cong 0.5$ are not held at the same time. Therefore, the total weight correction is not small.

In the next section, we will propose the training and pairing method to avoid the problem of the slow training as shown in Eq. (1).

5. Data Selection Method Combining Training and Pairing Processes

5.1 Data Selection and Training Algorithm

This method combines the training and the pairing processes as follows:

Step 1: Some data are selected from the training data sets $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ at random. The selected data sets are denoted $\boldsymbol{X}^r_1$ and $\boldsymbol{X}^r_2$, respectively.

Step 2: The MLNN is trained by using the data of $\boldsymbol{X}^r_1$ and $\boldsymbol{X}^r_2$ until the mean squared error (MSE) E satisfies,

$$E \leqq \varepsilon_1 \tag{12}$$

Step 3: The data denoted $\boldsymbol{x}^e_1$ and $\boldsymbol{x}^e_2$, for which the output error is greater than δ, are selected as they are located near the network boundary. These data sets are denoted $\boldsymbol{X}^e_1$ and $\boldsymbol{X}^e_2$, respectively.

Step 4: The data $\boldsymbol{x}^p_1$ and $\boldsymbol{x}^p_2$ are selected from $\boldsymbol{X}^r_1$ and $\boldsymbol{X}^r_2$, with which the distances $\|\boldsymbol{x}^p_1 - \boldsymbol{x}^e_2\|$ and $\|\boldsymbol{x}^p_2 - \boldsymbol{x}^e_1\|$ are the minimum. The set of $\boldsymbol{x}^p_1$ and $\boldsymbol{x}^p_2$ are also denoted $\boldsymbol{X}^p_1$ and $\boldsymbol{X}^p_2$, respectively.

Step 5: The data $\boldsymbol{x}^{pe}_1$ and $\boldsymbol{x}^{pe}_2$ are selected from $\boldsymbol{X}^r_1$ and $\boldsymbol{X}^r_2$, with which $\|\boldsymbol{x}^{pe}_1 - \boldsymbol{x}^p_2\|$ and $\|\boldsymbol{x}^{pe}_2 - \boldsymbol{x}^p_1\|$ are the minimum. The set of $\boldsymbol{x}^{pe}_1$ and $\boldsymbol{x}^{pe}_2$ are denoted $\boldsymbol{X}^{pe}_1$ and $\boldsymbol{X}^{pe}_2$, respectively.

Step 6: The MLNN is further trained by using the data of $\boldsymbol{X}^p_1$, $\boldsymbol{X}^p_2$, $\boldsymbol{X}^{pe}_1$ and $\boldsymbol{X}^{pe}_2$ until E satisfies,

$$E \leqq \varepsilon_2 \tag{13}$$

Step 7: Evaluation of the trained network. The classification performance of the trained network is evaluated by the validation data set. This data set is dependent on the application. This will be discussed in Sect. 7.2.2 more.

A general rule how to determine δ, ε_1 and ε_2 is not provided in this paper. It should be investigated more. At the present, we determine them by experience.

5.2 Off-Line and On-Line Training

The proposed data selection methods can be applied to both off-line training and on-line training [13]. In the off-line training, all the data are given at the beginning of the training.

If a large amount of training data is available, the data selection is desirable to reduce the training time and memory capacity. In the on-line training, the training data are not given all together, but are given successively. Furthermore, they may change continuously. If the data successively received are all accumulated, then the number of the data will be extremely large. Therefore, in this application, the training data selection is also important.

Off-line Training: In this case, we suppose that all the data are known in advance. After processing Steps 1 through 6, another $\boldsymbol{X}^{r*}_1$ and $\boldsymbol{X}^{r*}_2$ are selected from $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, and are added to the previous $\boldsymbol{X}^p_1$, $\boldsymbol{X}^p_2$, $\boldsymbol{X}^{pe}_1$ and $\boldsymbol{X}^{pe}_2$. Thus, the new data sets become

$$\boldsymbol{X}^r_1(new) = \boldsymbol{X}^{r*}_1 \cup \boldsymbol{X}^p_1 \cup \boldsymbol{X}^{pe}_1 \tag{14}$$

$$\boldsymbol{X}^r_2(new) = \boldsymbol{X}^{r*}_2 \cup \boldsymbol{X}^p_2 \cup \boldsymbol{X}^{pe}_2 \tag{15}$$

The processes Steps 1 through 6 are repeated using the new $\boldsymbol{X}^r_1$ and $\boldsymbol{X}^r_2$. Furthermore, the above processes are repeated until the error for all the data $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ satisfies the requirement.

On-line Training: In this case, we cannot get all the data in advance. The data will be observed sequentially. In Eqs. (14) and (15), $\boldsymbol{X}^{r*}_1$ and $\boldsymbol{X}^{r*}_2$ become the sequentially observed data sets. The remaining parts are the same as in the off-line training except for the validation data set, which will depend on stationary or nonstationary cases. Some volume of the recent data should be held to validate the trained network.

When the number of the data, used in the training and pairing method, is small, the meaningful data cannot find its partner, and will be removed. In order to avoid this problem, some lower bound for the number of the data is necessary. In the on-line training, it is better to select the training data after accumulating some number of the input data.

5.3 Data Distribution

The purpose of the training in Step 2 is to find the data, which locate close to the class boundary, with fewer computations. Therefore, the training is stopped at the middle stage using the criterion ε_1. In Sect. 6.2.2, this criterion for the off-line training is described.

Even though the training is not completely converged, the data, which locate close to the class boundary can be detected using the output error. The boundary formed through the initial training in Step 2 may be a little different from the class boundary. By the paring in Steps 4 and 5, the data locate near the class boundary can be selected.

On the other hand, the paring using all the data require a huge number of combinations. Furthermore, the training using the data selected by only the pairing converges very slow as discussed in Sect. 4.2. The details of selecting the data are described in the following.

For convenience, a two-dimensional pattern classification given by Fig. 1 (a) is employed. Class 1(#1) region is inside the circle and Class 2(#2) region is the outside.

In Step 2, the network is trained by using the data sets $\boldsymbol{X}_1^r$ and $\boldsymbol{X}_2^r$ randomly selected from all the data in Step 1. These data cannot cover exactly the class regions. Therefore, we suppose, for instance, a triangle boundary is formed after Step 2 as shown in Fig. 1 (b). It will be explained in the following that the data close to the class boundary can remain in ether class or both classes in Step 3. Furthermore, using these data as $\boldsymbol{x}_1^e$ and $\boldsymbol{x}_2^e$, the data close to the class boundary can be selected in both classes.

The regions in Fig. 1 (b) are categorized into four parts, A, B, C, and D. The network classifies the data in the triangle into Class 1 and the outside Class 2. The data in the regions B and D are correctly classified, and the corresponding output error are small. However, the data in B close to the boundary between A and B have relatively large error. The same situation occurs for the data in D close to the boundary between C and D. On the other hand, the data in the regions A and C are misclassified, and have large output errors. These data, $\boldsymbol{x}_1^e$ and $\boldsymbol{x}_2^e$, are selected in Step 3 due to their large output error.

In Step 3, however, the data far from the class boundary are included, and the data close to the boundary are missed in ether class. In Steps 4 and 5, the redundant data are removed and the necessary data are further extracted from $\boldsymbol{X}_1^r$ and $\boldsymbol{X}_2^r$. In Step 4, the data in $\boldsymbol{X}_1^e$ and $\boldsymbol{X}_2^e$ find the nearest data from $\boldsymbol{X}_1^r$ and $\boldsymbol{X}_2^r$, which are $\boldsymbol{x}_2^p$ and $\boldsymbol{x}_1^p$, respectively. Further, $\boldsymbol{x}_1^p$ and $\boldsymbol{x}_2^p$ find the nearest data from $\boldsymbol{X}_1^r$ and $\boldsymbol{X}_2^r$, which is $\boldsymbol{x}_2^{pe}$ and $\boldsymbol{x}_1^{pe}$, respectively. The shortest distance between two data in both classes means they face across the class boundary.

Figure 2 shows another example, where the data locate in the shaded parts are only satisfy the conditions' $MSE > \delta$ in Step 3, and are detected. Some data in the regions marked by oblique line will be missed.

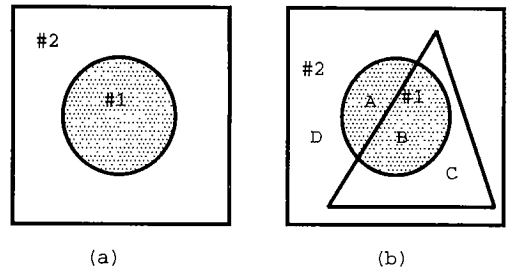


Fig. 1 (a) Class distribution, (b) Classification result by not well achieved network.

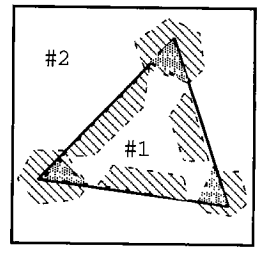


Fig. 2 Example of two-class classification.

By using the pairing method proposed in Sect. 4, the data in the different classes locate close to $\boldsymbol{x}_1^e$ and $\boldsymbol{x}_2^e$ can be found. They are denoted $\boldsymbol{x}_2^p$ and $\boldsymbol{x}_1^p$ as shown in Step 4, respectively. Therefore, the data near the circumference in B in Fig. 1 will be detected.

6. Computer Simulation

6.1 Computer Simulation Conditions

Two-dimensional two-class classification is employed for the computer simulations. The number of the input unit N is 2, and the number of the output unit K is 2. Then, the data is $\boldsymbol{X} = \{\boldsymbol{X}_1, \boldsymbol{X}_2\}$ and the input data is $\boldsymbol{x} = \{x_1, x_2\}$. Six hidden units are used.

Figure 3 shows a concept of the problems. One of the classes is shown as shaded region, and the other is dotted region. Whited region between the classes shows a gap, so there is no overlap.

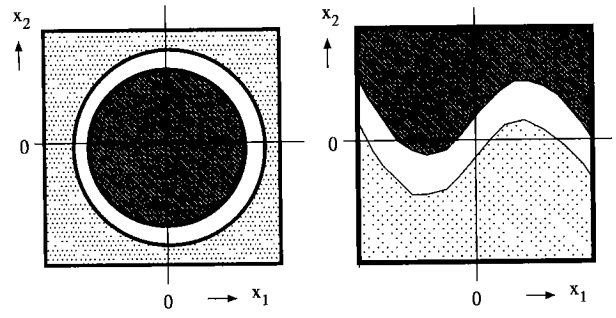
For the training, a learning-rate parameter η is 0.1, and a momentum coefficient α is 0.8. They are decided by experience. The circle in square is called Problem 1, and the sinusoidal in square is called Problem 2 in the following sections.

In Problem 1, two classes are defined as follows:

$$\boldsymbol{X}_1 = \{\boldsymbol{x} \,|\, x_1 + x_2 \leqq (r - \gamma)^2\} \tag{16}$$

$$\boldsymbol{X}_2 = \{\boldsymbol{x} \,|\, x_2 + x_2 > (r + \gamma)^2\} \tag{17}$$

here, r is the radius of the circle and is 0.39. γ is width of the gap, and is 0.02.



(a) Circle in Square (b) Sinusoidal in Square

Fig. 3 Concept of problems. (a) Circle in square, (b) Sinusoidal in square.

In Problem 2, two classes are defined as follows:

$$\boldsymbol{X}_1 = \{\boldsymbol{x} \,|\, A\sin(2\pi x_1) \leq x_2 - \gamma\} \tag{18}$$

$$\boldsymbol{X}_2 = \{\boldsymbol{x} \,|\, A\sin(2\pi x_2) > x_2 + \gamma\} \tag{19}$$

where, A is 0.22.

A total number of the data in each class is 1000. Two hundreds of the data are selected randomly from 1000 data in each class, and are used as the validation data.

6.2 Off-Line Training

6.2.1 Pairing Method

For the off-line training, the pairing method is first used. Figure 4 shows the entire data of Problem 1, and Fig. 5 shows the data found by the pairing method. From Fig. 5, the class boundary is formed by the selected data properly. Sixty-five data are selected for each class.

The MLNN is trained with the selected data. The stopping criterion is $\varepsilon_2 = 0.001$ in Eq. (13). Iteration of 23763 is needed for convergence.

6.2.2 Training and Pairing Method

Next, the training and pairing method is employed. The initial training is stopped at MSE $< \varepsilon_1 = 0.05$. All the data are used in the initial training. Then, the data are selected by using the criterion $\delta = 0.073$ in Step 3. In Problem 1, 207 data are selected from Class 1, and 164 from Class 2. In Problem 2, 116 and 150 data are selected from Class 1 and 2, respectively. The pairing is done using these data in Steps 4 and 5. As a result, 114 and 62 data are selected in Problems 1 and 2, as shown in Table 1. ε_2 in Step 6 is 0.001.

Figure 6 shows the distribution of the data selected above. From these figures, the boundaries are detected properly.

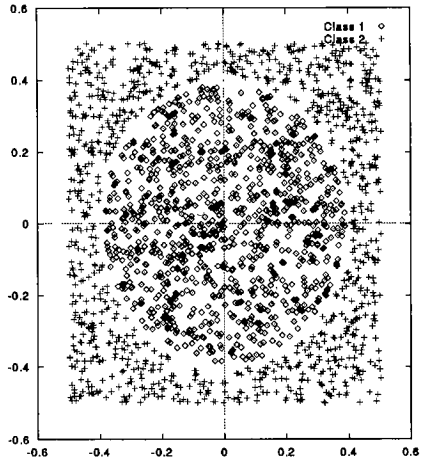


Fig. 4 Entire data of Problem 1.

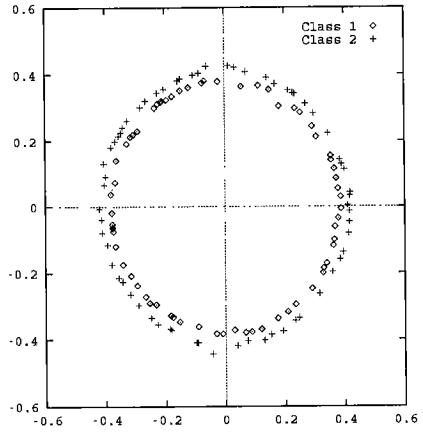


Fig. 5 Selected data by pairing method.

Table 1 Comparison of computational complexity between conventional and proposed training methods.

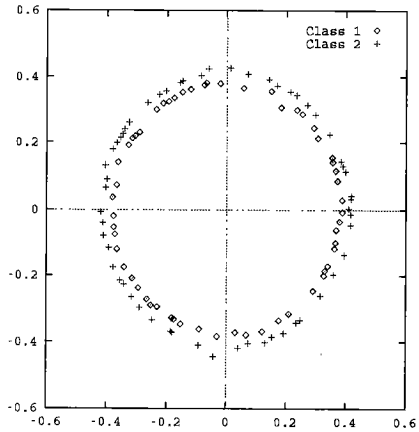
	Prob. 1			Prob. 2	
	Conv1	Conv2	Prop	Conv1	Prop
Init.	0	112	134	0	18
Epoch	2444	1701	4394	89	390
N of data	2000	1800	114	2000	62
Comp.	1	0.63	0.10	1	0.14

Init.: Epoch of initial training. N of data: Number of data.
Conv: Conventional method. Comp. Computation
Prop: Proposed method

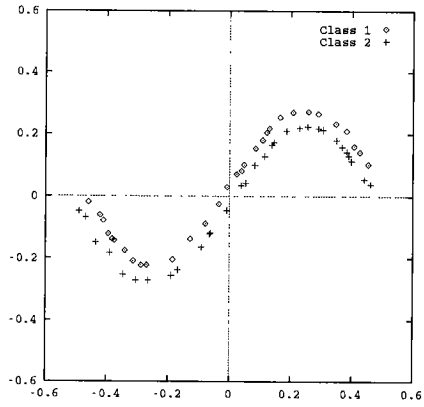
6.2.3 Comparison with Conventional Methods

Training with all the data is called the Conventional Method 1 in this paper. Furthermore, the following method is called Conventional Method 2. Some number of the data are selected from the entire data at random, and are used in the initial training. All the data are used after this training [15]. In this simulation, 100 data for each class is used in the initial training.

Table 1 shows the results. Computation complexity is defined as the number of the data multiplied by



(a) Circle in square.



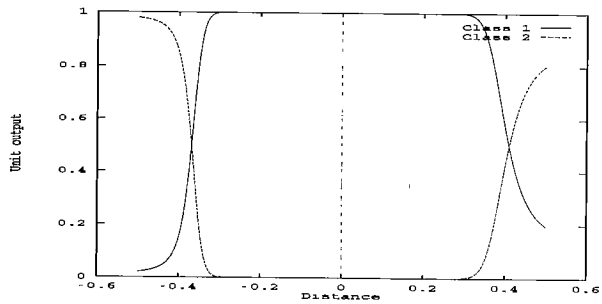
(b) Sinusoidal in square.

Fig. 6 Selected data by training and pairing method.

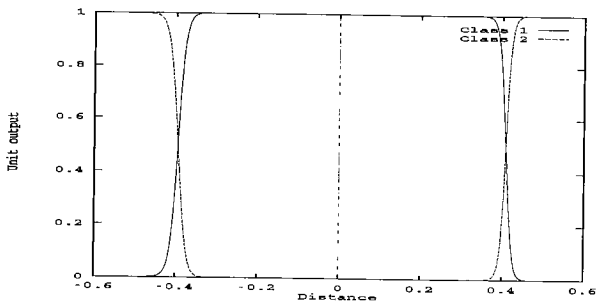
the number of iterations of the training. In the last row (Comp.), the computational complexity of Conventional Method 1 is normalized be 1. In both Problems 1 and 2, the computational complexities of the proposed method are the minimum.

6.2.4 Relation between Distance and Network Output

Figure 7 shows relation between the distance, between the input data and the boundary, and the network output. The MLNN in Step 2 of Problem 1 is used. The input data of (a) is $x_2 = 0$ and (b) is $x_1 = x_2$. This means, the data locate on the vertical line and the line with 45 degree slope. The horizontal axis is the distance from the origin of the data space to a data. The vertical axis is the output of the output unit to the input data of the horizontal axis. The slope of the curves in (b) are much steeper than those of (a) around the class boundary $\pm$ 0.4 on the horizontal axis. These figures demonstrate the relation between the distance and the network output is not always the same. We cannot directly estimate the distance from the output, as mentioned in Sect. 5.3.



(a) Input data satisfy $x_2 = 0$



(b) Input data satisfy $x_1 = x_2$

Fig. 7 Network output and distance in data space.

6.3 On-Line Training

The on-line training is simulated using the partial data of Problem 1. The entire data $\boldsymbol{X}$ is separated into three sets as described below.

$$\boldsymbol{X}_{up} = \{\boldsymbol{x} \,|x_2 \geqq 0.167\} \tag{20}$$

$$\boldsymbol{X}_{mid} = \{\boldsymbol{x} \,|-0.167 < x_2 < 0.167\} \tag{21}$$

$$\boldsymbol{X}_{down} = \{\boldsymbol{x} \,|x_2 \leqq -0.167\} \tag{22}$$

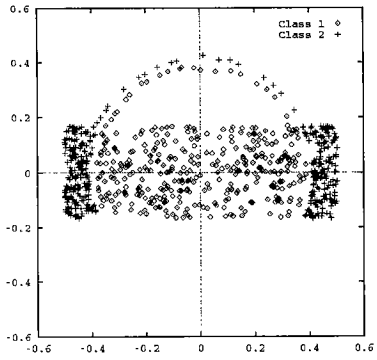
Each data set includes 333 data.

$\boldsymbol{X}_{up}$ is used as $\boldsymbol{X}_1^r$ and $\boldsymbol{X}_2^r$ in the first training process. $\boldsymbol{X}_{mid}$ and $\boldsymbol{X}_{down}$ are used as the remained training data in Step 1 of Sect. 5. The stopping criterion ε_1 in Step 1 is 0.05, and ε_2 in Step 6 is 0.01, respectively. δ for all training process is 0.073.

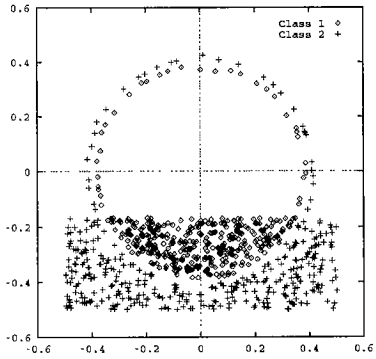
Figure 8 shows the result of the on-line training using the selected data. The training is converged and their percentage of correctly classified are 100% for the entire data. The boundary is also detected correctly.

Two conventional methods and the proposed method are compared in computational complexity. In Conventional Method 1, the data given until the present are accumulated and are used as the present training data. In Conventional Method 2, as discussed in Sect. 7.2.2, the above accumulated data are treated as all the data at the present. The initial training and the overall training are done using these data.

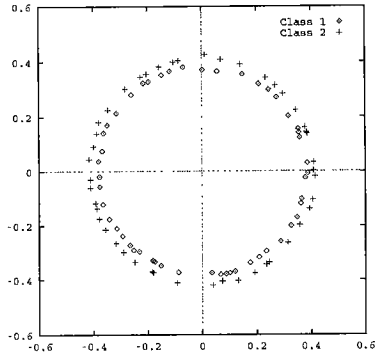
To compare three methods, the computational complexity of Conventional Method 1 is normalized be 1. The computational complexity of Conventional Method



(a) Training data in Step 2 in 2nd training process. Upper one third region is selected data in 1st training process and middle region is newly given data.



(b) Training data of Step 2 in 3rd training process. Upper two third regions are selected data in 2nd training process. MLNN is trained with data of (a). Rest of the region is newly given data.



(c) Selected data of Step 5 in 3rd training process. MLNN is trained with data of (b).

Fig. 8 Selected data in on-line training: Problem 1.

2 is 0.56 and that of the proposed method is 0.29. From this result, the proposed method is also superior to the conventionals.

7. Conclusion

The training data selection method for the multilayer neural networks (MLNNs) has been proposed. The selected small number of the data can guarantee generalization with low computational complexity. The proposed method combines a training and pairing processes in this order. The training process provides the semi-optimum network, with which the next training can be hastened. The pairing process can select the nearest data of the different classes, which locate close to the class boundary. The proposed method can be applied to the off-line and the on-line training.

Two examples of two-dimensional two-class pattern classification were simulated. In the off-line case, the proposed method can reduce the number of the training data and the training time. The classification performance is the same as the conventional method using all the data. In the on-line training, the same properties described above are obtained using the successively received data.

References

[1] D.E. Rumelhart and J.L. McCelland, et al., "Parallel Distributed Processing," the MIT Press, 1986.
[2] R. Vanderbeek and A. Garper, "A back-propagation network for analog signal separation in high environment," Proc. IJCNN'92, Boltimore MD, pp.664–669, 1992.
[3] Z.H. Michalopoulou, L.W. Nolte, and D. Alexandrou, "Performance evaluation of multilayer perceptrons in signal detection and classification," IEEE Trans. Neural Network, vol.6, no.2, pp.381–386, 1995.
[4] G. Veciana and A. Zakhor, "Neural net-base continuous phase modulation receivers," IEEE Trans. Commun., vol.40, no.8, 1992.
[5] K. Hara and K. Nakayama, "Multi-frequency signal classification by multilayer neural network and linear signal processing methods," IEICE Trans. Fundamentals., vol.E80-A, no.5, pp.894–902, May 1997.
[6] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," IEEE Trans. Neural Networks, vol.5, no.4, pp.537–550, July 1994.
[7] M. Hagiwara, "Back-propagation with artificial selection—Reduction of the number of learning times and that of hidden units," IEICE Trans., vol.J74-D-II, no.6, pp.812–818, June 1991.
[8] N. Nakayama and Y. Kimura, "Optimization of activation functions in multilayer neural network," Proc. ICNN'94, Orland, Florida, pp.431–436, June 1994.
[9] T. Ueda, K. Takahashi, and S. Mori, "A structural learning for multi-layered neural networks by using fuzzy set—A pruning weights and units," IEICE Trans. vol.J78-D-II, no.10, pp.1479–1490, 1995.
[10] K. Fukumizu and S. Watanabe, "Error estimation and learning data arrangement for neural networks," Proc. ICNN'94, Orland, Florida, pp.777–780, 1994.
[11] C. Cachin, "Pedagogical pattern selection strategies," Neural Networks, vol.7, no.1, pp.175–181, 1994.
[12] Qi Li and D.W. Tufts, "Principal feature classification," IEEE Trans. Neural Networks, vol.8, no.1, pp.155–160, 1997.
[13] S. Haykin, "Neural Networks—A comprehensive foundation," pp.57–59, Macmillan College Publishing Company, 1994.
[14] K. Hara and K. Nakayama, "Comparison of activation functions in multilayer neural network for pattern classification," Proc. ICNN'94, Orland, Florida, pp.2997–3002,

1994.

[15] Y. Mori, "Hand-written KANJI character recognition by a PDP model," IEICE Trans., vol.J73-D-II, no.8, pp.1268–1274, 1990.

[16] K. Hara and K. Nakayama, "Selection of minimum training data for generalization and on-line training by multilayer neural networks," Proc. ICNN '96, Washington D.C., pp.436–441, July 1996.

Kazuyuki Hara received the B.E. and M.E. degree in electrical engineering from Nihon University, Tokyo, Japan, in 1979 and 1981, respectively. He received the Dr.degree in computer science from Kanazawa University in 1997. From 1979 to 1987 he was involved in NEC Home Electronics Corporation. He joined the Toyama Polytechnic College in 1987, where he was a Lecturer. He is currently a Lecturer at Gunma Polytechnic College. His current research interests include neural networks and pattern classifications. Dr.Hara is a member of IEEE, INNS and IPSJ.

Kenji Nakayama received the B.E. and Dr.degree in electronics engineering from Tokyo Institute of Technology (TIT), Tokyo, Japan, in 1971 and 1983, respectively. From 1971 to 1972 he was engaged in the research on classical network theory in TIT. He was involved in NEC Corporation from 1972 to 1988, where his research subjects were filter design methodology and signal processing algorithms. He joined the Department of Electrical and Computer Engineering at Kanazawa University, in Aug. 1988, where he is currently a Professor. His current research interests include neural networks, adaptive signal processing and signal theory. Dr.Nakayama is a senior member of IEEE and a member of INNS.