A Training Data Selection in On-Line Training for Multilayer Neural Networks

Kazuyuki Hara, Gunma Polytechnic Coll., 918 Yamana-cho Takasaki, Gunma 370-12 Japan.
Kenji Nakayama, Fac. of Eng., Kanazawa Univ. 2-40-20 Kodatsuno Kanazawa 920 Japan.
Ashraf A. M. Kharaf, Grad. School of Nat. Sci. & Tech. Kanazawa University.

Abstract

In this paper, a training data selection method for multilayer neural networks (MLNNs) in on-line training is proposed. Purpose of the reduction in training data is reducing the computation complexity of the training and saving the memory to store the data without loosing generalization performance. This method uses a pairing method, which selects the nearest neighbor data by finding the nearest data in the different classes. The network is trained by the selected data. Since the selected data located along data class boundary, the trained network can guarantee generalization performance. Efficiency of this method for the on-line training is evaluated by computer simulation.

1. Introduction

In pattern classification problems, a multilayer neural network (MLNN) trained by supervised learning algorithms [1] are capable of extracting common features or rules of training data through a training process. This is a benefit of using the MLNN for the classification.

A huge amount of the training data may guarantee generalization performance of the MLNN. On the other hand, it will require a very long training time and amount of computations. To avoid this difficulty, there are many papers related to reducing the computations for training. One of the solutions is simplify the network structure. Battiti[2] used the mutual information to evaluate the information content of input feature to limit the input dimensionality. The training time can be reduced by pruning the hidden units. Many papers are presented related to this approach [3, 5, 4]. Another approach is to reduce the number of training data. Cachin[6] proposed the error-dependent repetition (EDR). Presentation probability of a training data is proportional to the MLNN output error. Li [7] combined Fisher's linear discriminant analysis and the maximal signal-to-noise-ratio discriminant analysis to find the principle feature of the data. These approaches are performed in off-line training. For on-line training, above methods require to accumulate all the given data, so they are not suit for the on-line training.

In this paper, we propose a training data selection method for the on-line training. Purpose of the proposed method is to find the important data to guarantee the generalization performance. At the same time, this allow to reduce the number of the data, then the memories which store the data are reduced. The data selection is done by a method called the pairing method in this paper. This method selects the nearest neighbor data in the different classes. The data which are not selected will be eliminated, and will not be used in the future training. If the training with selected data will be converged, the generalization performance is guaranteed. At the same time, the number of selected data is relatively small.

In the on-line training, a small number of the training data are given in successively, and the network adjusts the connection weights to minimize the output error for the data. Moreover, it is possible that the class boundaries may continuously be changed. In this case, if the class boundaries are not overlapped, the network requires to keep the classification performance to the past data and also to adapt to the new data. The proposed method can solve these requirement by combining the previously selected data into the present training data.

Efficiency of the proposed method is investigated through computer simulations.

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

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

$$y_j = f_H(net_j) \quad (2)$$

As the activation function, one of these sigmoid functions can be used.

$$f_{H1}(net_j) = \frac{1 - e^{-net_j}}{1 + e^{-net_j}} \quad (3)$$

$$f_{H2}(net_j) = \frac{1}{1 + e^{-net_j}} \quad (4)$$

where, $f_H(\cdot)$ is an activation function in the hidden layer and θ_j is a bias of the unit. Eq. (3) is called Bipolar and Eq. (4) is called Unipolar in this paper. The hidden unit output y_j is transferred to the output layer. The same process, as in the hidden layer, is carried out in the output layer. The number of output units is equal to that of the classes. The MLNN is trained so that a single output unit responds to one of the classes.

3. On-Line Training Method with Data Selection

In the on-line training, a small number of training data are given for one training process, and this process is repeated many times. By train the network with given training data, the network will classify the training data well, however, there is no guarantee that the network classify the data of the other training processes correctly. To perform this kind of classification, the network must have generalization performance to the entire data. This can be solved by accumulates all the data of all the training processes. In this case, a huge memories to store a number of data are required, and the training with a huge number of data becomes difficult.

In the next subsections, we introduce a pairing method to select the data near the class boundary. The selected data will be used for training. Training algorithm is explained in subsection 3.3.

In each training process except for the initial training process, previously selected data are combined to new training data. In this case, only the selected data are path to the next training process, then computation will not be increased so much. By adding the neighbor data in the previous process, the network will be keep the performance both for the past data and for the new data.

3.1. Pairing Method for Data Selection

In this paper, two classes are taken into account. However, this can be applied to more than two classes.

The proposed method is called pairing method in this paper. This method selects the nearest data of different classes using the Euclidean distance. The data selection is carried out through the following steps.

Step 1: Select some number of $\boldsymbol{x}_1$ (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 in Step 1, this selection is completed. Otherwise, return to Step 1, and repeat the above process.

Finally, the data $\boldsymbol{x}_1^p, \boldsymbol{x}_2^p, \boldsymbol{x}_1^{p'}$ and $\boldsymbol{x}_2^{'p}$ selected based on the Euclidean distance are included in the selected data set $\boldsymbol{X}_1^p$ and $\boldsymbol{X}_2^p$.

In this paper, we assumed that the class distributions are not overlapped. Then, the data are classified by the linear separator. The MLNN has a similar property to the linear separator in the classification, so if the class boundaries in the data space are based on the distance and pairs of the shortest data between the classes are selected, these data are located close to the class boundaries.

3.2. Selecting Activation Function for Training Using Paired Data

The distance between the paired data is short. This means the paired data are similar to each other. However, the output should be entirely different. This kind of classification is a difficult problem for the MLNNs. In this subsection, we will analyze convergence behavior of the back propagation algorithm using the paired data.

3.2.1. Amount of update of connection weight and Hidden Unit Output

The connection weights are updated respect to the output error of the network. The change of the connection weight $\Delta_p w_{jk}$ is defined as follow.

$$\Delta_p w_{jk} = -\eta \frac{\partial E_p}{\partial w_{jk}} \quad (5)$$

Here, $|\partial E_p / \partial w_{jk}|$ is the magnitude of the gradient for the pth data in the weight space. w_{jk} is the connection weight from the jth hidden unit to the kth output unit. η is a learning rate. $\partial E_p / \partial w_{jk}$ of Unipolar is as,

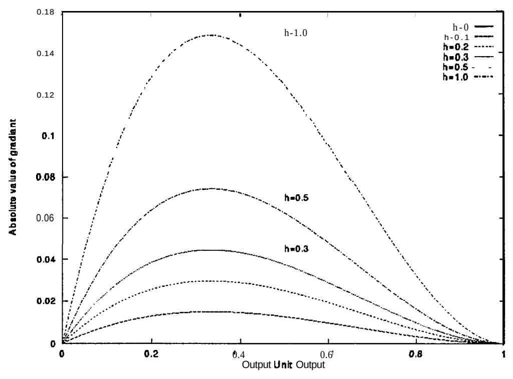

Figure 1: **Gradient value related to output unit output.**

$$-\frac{\partial E_p}{\partial w_{jk}} = (t_{pk} - y_{pk})(1 - y_{pk})y_{pk}y_{pj} \qquad (6)$$

Where, t_{pk} is the target of the pth data, and y_{pk} is the output of the output layer. y_{pj} is the output of the ith hidden layer. In Fig. 1, by using Eq. (6), the absolute value of the gradient, $|\partial E_p/\partial w_{jk}|$, for several hidden unit outputs are plotted along the output unit outputs. In this figure, the target $t_{pk} =$ 1.0 is used. So, the output error becomes larger in the left side of the output axis, and it becomes smaller in the right side. h = 0 means the hidden unit output is zero. And h=0.1 means the hidden unit output is 0.1, and so on.

From this figure, when the hidden unit output is small, the gradient is almost equal for any output error. When the hidden unit output is small, the connection weights are not properly modified respect to the output error.

3.2.2. Amount of update of connection weight and activation function

We assumed that the initial connection weights are given as small Gaussian random number with zero mean. For example, the range of the random number is [-0.2,0.2]. In this case, if the number of the inputs is large enough, the input of the hidden units are almost zero. If Unipolar is used as the activation function in the hidden layer, the hidden unit outputs 0.5 for the input of zero. So, the gradient is different with respect to the output error. However, if Bipolar is used, the hidden unit outputs zero for input of zero.

From Fig. 1, for h=0, the gradient is zero for any output error, so the connection weights are not updated respect to the output error. Therefore, at the beginning of the training, the amount of update of the connection weight is larger when using Unipolar than using Bipolar in hidden layer. Another word, for faster convergence, Unipolar is more suitable than Bipolar.

For randomly selected data, above discussion is not always true. These data are located randomly, and the distances between the data are not so small. In this case, the connection weights are updated properly for the sigmoid functions of Bipolar and Unipolar.

3.3. Training Data Selection in On-Line Training

In on-line training, training data are given in successively. The proposed algorithm uses the network trained in the previous training, which will be slightly changed by trained using new data. This method combines the data pairing method and the training as follows:

Step 1: Some number of the training data included in $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are given. Let these data be $\boldsymbol{X}_1^s$ and $\boldsymbol{X}_2^s$. $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are the entire data.

Step 2: Select the data $\boldsymbol{x}_1^p$ and $\boldsymbol{x}_2^p$ from $\boldsymbol{X}_1^s$ and $\boldsymbol{X}_2^s$ by using the pairing method. The set of the selected data be X_1^P and X_2^P, respectively.

Step 3: Train the MLNN by using $\boldsymbol{X}_1^p$ and $\boldsymbol{X}_2^p$ until Mean Squared Error (MSE) at the output layer becomes less than desired value.

At the initial of the training, only new training data are included into $\boldsymbol{X}_1^s$ and $\boldsymbol{X}_2^s$. After that, a set of new training data $\boldsymbol{X}_1^s$ and $\boldsymbol{X}_2^s$, and the selected data in the previous training process $\boldsymbol{X}_1^p$ and $\boldsymbol{X}_2^p$ will be used as the training data in Step 1. Until the last data are processed, these steps will be repeated. From Step 1 through Step 3 is called training process in this paper.

The proposed method uses the network and the selected data in the previous training process in the new training process. This allows to achieve the high classification performance for the previous data and the new data.

For the off-line training, if the number of the new data is large, some number of the data should be selected, and are included in $\boldsymbol{X}_1^s$ and $\boldsymbol{X}_2^s$. After that, return to Step 3.

4. Computer Simulation

Two-dimensional two-class classification is employed for computer simulations. Figure 2 shows a concept of the problems. One of the classes is shown as hatched region, and the other is dotted region. Dark dotted region between the classes shows a gap, so there is no overlap. The gap is not shown in (b). Fig. 2 (a) is called Problem 1, and Fig. 2 (b) is called Problem 2, respectively.

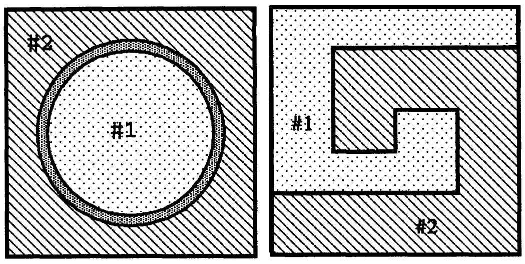

Figure 2: Concept of problem.(a)Circle in square(left), (b)Spiral(right).

In the network, two input units and two output units are used. Then, the data are X = $\{\boldsymbol{X}_1, \boldsymbol{X}_2\}$ and the input data is $\boldsymbol{x} = \{x_n, n = 1, 2\}$.

For the training, learning-rate η is 0.1, and momentum constant α is 0.8. These are decided by experience. In Problem 1, two classes are defined as follows:

$$\boldsymbol{X}_1 = \{\boldsymbol{x} \,|x_{11} + x_{12} \le (r - \gamma)^2\} \qquad (7)$$

$$\boldsymbol{X}_2 = \{\boldsymbol{x} \,|x_{21} + x_{22} > (r + \gamma)^2\} \qquad (8)$$

here, r is the radius of the circle and is 0.39. γ is the width of the gap, and is 0.02 for problem 1 and is 0.04 for problem 2.

4.1. Comparison of Convergence Speed

In subsection 3.2.2, it is claimed that the sigmoid function of Unipolar converge faster than the one of Bipolar. This analytical result is verified by the computer simulation.

The convergence speed of two networks trained using the selected data by the pairing method are compared. One network is using Bipolar as the activation function in the hidden layer, and the other is using Unipolar .

For both networks, six hidden units are used in the hidden layer. Problem 1 is used. 130 training data are selected by the pairing method. Training is stopped when MSE is less than 0.01.

Table 1 shows the comparison result. In this table, Bipolar uses Eq. (3) in hidden layer and Unipolar uses Eq. (4). The network having Unipolar is converged faster than the one of Bipolar. This result will support the analytical results.

In Fig 3, the convergence curves of two sigmoid functions are shown. From this figure, as we analyzed in Sec. 3.2.2, Unipolar iterates more than Bipolar to reduce the MSE at the beginning of the training.

Table 1: Comparison of Convergence speed of Bipolar and Unipolar sigmoid functions.

Kind of Sigmoid Function	Iteration
Bipolar	22347
Unipolar	8783

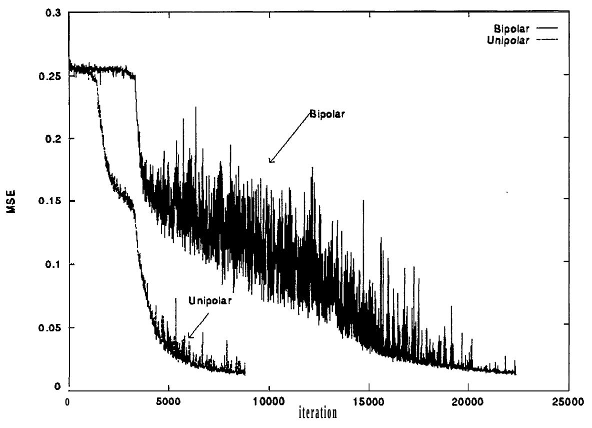

Figure 3: Convergence curve of two sigmoid functions.

When the entire data are used, 176 iterations for f_{H2}, and 183 iterations for f_{H1} are needed to converge on the proper MSE. Therefore, for the training with many data, the convergence speed of two sigmoid functions is not remarkably different.

4.2. On-Line Training

In this simulation, 25 data for each class are given to the network in one training process, and the training process is repeated 14 times. Then totally, 25 x 14 = 350 data for each class are used. Training in Step 3 is stopped at the MSE of $\varepsilon<0.01$. The classification performance is evaluated for the entire data. Six hidden units are used for Problem 1, and eight hidden units are used for Problem 2. In the hidden layer and the output layer, Unipolar is used.

4.2.1. Simulation results

Table 2 and 3 shows the results. For both problems, training is converged properly, and the classification rates are sufficiently high in every training process. From this result, by using the proposed method, high accuracy can be achieved in the on-line training.

Except for the initial training, the training is converged with a small number of iterations. This result shows that the connection weights are adjusted properly at the initial training process. Even if the data class boundaries formed by the training data will be slightly changed in each process, the network easily adapts to the class boundary changing.

The number of selected data is increasing process by process, however, it is saturated after the sixth or seventh training processes. From this result, the efficient data are selected in the on-line training. Figures 4 and 5 show the selected data in the initial and the 14th training processes for Problem 1 and 2, respectively. The regions formed by the network is shown in Fig. 6. From these figures, the class boundaries are properly formed by selected data.

Table 2: Simulation result for problem 1.

Proc. No.	No. data	Iteration	Class. rate		
			#1	#2	ave
1	18	8184	95.7	91.4	93.6
2	27	1722	98.6	91.2	94.9
3	33	1466	99.0	96.9	96.5
4	33	252	98.6	96.7	97.7
5	36	453	99.7	96.7	98.2
6	40	3	99.9	95.7	97.8 ,
7	41	2684	99.8	99.1	99.5
8	42	1	99.8	99.1	99.5
9	44	72	99.9	99.8	99.9
10	44	4	99.9	99.8	99.9
11	45	40	99.9	99.8	99.9
12	45	40	99.8	99.8	99.8
13	45	7	100	99.8	99.9
14	48	13	100	99.7	99.9

Table 3: Simulation result for problem 2.

Proc. No.	No. data	Iteration	Class. rate		
			#1	#2	ave
1	16	9248	86.1	93.5	89.8
2	23	5544	96.4	90.7	93.6
3	28	2101	98.7	91.2	95.0
4	33	485	98.8	93.9	96.4
5	30	697	98.2	91.2	94.7
6	36	82	95.3	95.3	95.3
7	40	650	95.3	97.5	96.4
8	38	164	94.1	98.7	96.4
9	37	56	93.8	98.3	96.1
10	42	18	92.9	99.1	96.0
11	46	22	93.9	99.0	96.5
12	44	58	92.9	98.0	95.5
13	41	32	92.4	98.8	95.6
14	43	123	98.7	98.0	98.4

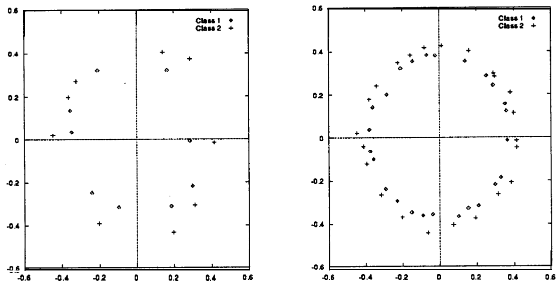

Figure 4: Selected data at the first (left) and the 14th(right) training process. Problem 1.

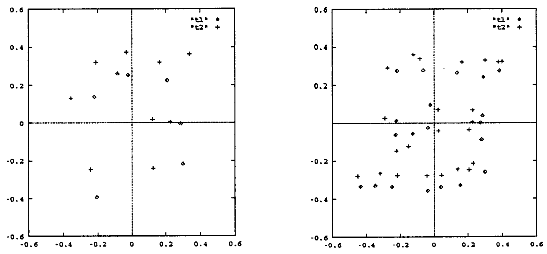

Figure 5: Selected data at the first(left) and the 14th(right) training process. Problem 2.

4.2.2. Amount of memory required in conventional method and proposed method

The on-line training can be performed by accumulating the training data of each training process. This will be treated as a conventional on-line training in this paper. In the conventional method, all the data given in the past processes must be kept in the memory. Then a huge memory is required to store all the data in every training process.

In this simulation, Problem 2 is used. Table 4 shows the number of training data and the classification rate of the proposed method and the conventional method. The number of the training process is 14. These two methods achieved the same classification performance.

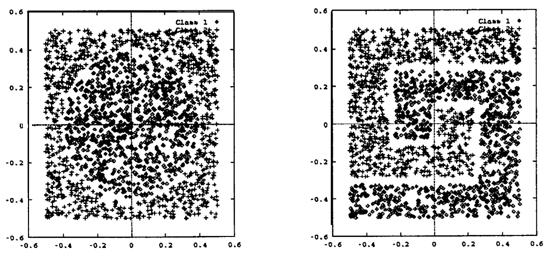

Figure 6: Regions formed by the network. Problem 1(left) and Problem 2(right). 14th training process.

Table 4: **Number of training data and classification rate of proposed method and conventional method.**

Method	Number of training data	Classification rate
Proposed	43	98.4
Conventional	700	98.6

Table 5: **Transition behavior to class boundary changes**

Process No.	Iteration	Class. rate	
		X^B	X^A
14th	–	99.9	86.8
15th	15	87.2	99.0
16th	1	86.3	98.6

This means that the class boundary is properly selected by the proposed method with a small number of training data.

4.3. Adaptability of Class Boundary Changes

In the on-line training, there is a possibility that the class boundary changes during the training processes. Therefore, the network is required to adapt to the change in the class boundary. In this subsection, the transition behavior of the proposal method to the change in the class boundary is verified. Problem 1 is used.

The change in the region assumes the center of the circle is moved from (0,0) to (0.1,0). For convenient, the entire data set before center is moved is denoted $\boldsymbol{X}^B$ and the entire data data after center is moved is denoted $\boldsymbol{X}^A$, respectively. The selected data at nth training process is $\boldsymbol{X}^{Sn}$. The data $\boldsymbol{X}^B$ and $\boldsymbol{X}^A$ in nth training process is denoted $\boldsymbol{X}^{Bn}$ and $\boldsymbol{X}^{An}$. Verification is done by the following procedure.

1. Train the network until 14th training processes. This is shown in Table 2. At 14th process, $\boldsymbol{X}^{B14}$ is used.
2. At 15th training process, $\boldsymbol{X}^{B14}$ and $\boldsymbol{X}^{A15}$ are combined, and $\boldsymbol{X}^{S15}$ are selected by the pairing method. The network is trained with $\boldsymbol{X}^{S15}$.
3. At 16th training process, $\boldsymbol{X}^{S15}$ and new data $\boldsymbol{X}^{A16}$ are combined. $\boldsymbol{X}^{S16}$ are selected by the pairing method. The network is trained with $\boldsymbol{X}^{S16}$.

The table 5 shows the classification rate for $\boldsymbol{X}^B$ and $\boldsymbol{X}^A$. From the table, the proposed method is capable of adapting the change of the region.

5. Conclusion

The training data selection method used in the on-line training have been proposed. The pairing method has extracted the data locate along the class boundary. To do this, the Euclidean dist ance has been employed. For the training, only the selected data are used, and the remained data are not used in the training.

Proposed method has been evaluated by the computer simulation. The training has converged by using selected data. The proposed method achieved the same classification performance as conventional method. Therefore, the proposed method can select the important training data to guarantee the generalization performance in the on-line training. The number of stored data can be drastically reduced.

In this paper, we have assumed the class data are not overlapped. However, for real world applications, this assumption will not be satisfied. To solve this problem, the weighted decay of the learning parameters along the training iteration for the previous data will be required. This is our future work.

References

[1] D. E. Rumelhart and J. L. McCelland, *Parallel* ***Distributed*** Processing. The MIT Press, 1993.

[2] R. Battiti, ***Using Mutual Information for Selecting Features in Supervised Neural Net Learning,*** IEEE Trains. Neural Networks, 5, 4, 537-550, 1994.

[3] M. Hagiwara, ***Back-propagation with*** *Artificial* ***Selection - Reduction*** *of* ***the Number*** *of* ***Learning times and that*** *of* ***hidden units- (in Japanese),*** IEICE Trans. J74-D-II, 6, 812-818, June, 1991.

[4] R. Reed, ***Pruning algorithms - A survey,*** IEEE Trans. Neural Networks, 4, 740-747, 1993.

[5] N. Nakayama and Y. Kimura, *Optimization of* ***Activation Functions in*** *Multilayer* ***Neural Network,*** Proc. ICNN'94, Orlando, Florida, 431-436, 1994.

[6] C. Cachin, ***Pedagogical pattern selection strategies.*** Neural Networks 7, 1, 175-181, 1994.

[7] Qi Li and D. W. Tufts, *Principal* ***Feature*** *Classification.* IEEE Trans. Neural Networks, 8, 1, 155-160, 1997.

[8] S. Haykin, *Neural* ***Networks - A comprehensive foundation.*** IEEE Press 57-59 1994.

[9] G. J. Gibson, ***C.*** F. ***N.*** Cowan, ***On the decision regions*** *of multilayer perceptrons.* IEEE Proc. 78, 10, 1590-1594 1990.

[10] K. Hara and K. Nakayama, ***Selection*** *of* ***Minimum Training Data*** *for Generalization* ***and Online Training by*** *Multilayer* ***Neural Networks.*** Proc. ICNN'96, Washington D.C., 436-441, July 1996.