# Recognition of Unconstrained Handwritten Numerals by a Radial Basis Function Neural Network Classifier

Hwang, Young-Sup and Bang, Sung-Yang

Department of Computer Science & Engineering

Pohang University of Science and Technology

Pohang, 790-784, Korea

**Abstract**

Among the neural network models RBF(Radial Basis Function) network seems to be quite effective for a pattern recognition task such as handwritten numeral recognition since it is extremely flexible to accommodate various and minute variations in data. Recently we obtained a good recognition rate for handwritten numerals by using an RBF network. In this paper we show how to design an RBF network classifier for a given problem in a well defined and easy-to-follow manner. We also report on the experiments to evaluate the performance of the RBF network classifier so designed.

*Keywords:* radial basis function network, handwritten numeral recognition, pattern classification, clustering.

## 1 Introduction

There are various methods for pattern classification problems like a handwritten numeral recognition and each of them has pros and cons. Table 1 presents a summary of the evaluation of the representative pattern classification methods based on training time, hardware requirement and classification time. We excluded the recognition rate from the criteria because it is difficult to evaluate. Here the training time means the time required to develop the classifier by training or statistical calculation after it is designed. And the hardware requirement generally means the size of the computer memory.

Table 1. Comparison of classifiers

| Classifier | Training time | Hardware requirement | Classification time |
|---|---|---|---|
| RBF[a] | middle | middle | middle |
| BP[b] | long | small | short |
| K-NN[c] | none | large | long |
| LVQ[d] | long | small | middle |
| MDC[e] | short | small | short |

[a] Radial basis function network
[b] Error back propagation network
[c] K nearest neighbor
[d] Learning vector quantization
[e] Minimum distance classifier

It depends on the given problem and the other requirements which classification method we choose to use among these various methods. For example, it has been reported that several methods obtained the same high recognition rate for a handwritten numeral classification problem(Lee, 1991; Ng and Lippmann, 1991). Therefore, in this case we may well choose a method which best satisfies the other given requirements. According to Table 1, the radial basis function(RBF) network ranks in the middle for all criteria. Therefore the RBF network could be a reasonable choice for those classification problems which do not have any particular requirements.

The architecture and the training methods of an RBF network are well-known(Moody and Darken, 1989; Poggio and Girosi, 1990; Musavi et al., 1992; Wettschereck and Dietterich, 1992; Vogt, 1993; Haykin, 1994; Monica Bianchini and Gori, 1995). However most of the training algorithms are for function approximation. Recently we proposed an effective training algorithm of an RBF network when the problem is a pattern classification(Hwang and Bang, 1997). Here we will apply the proposed method to a typical pattern classification problem of a handwritten numeral recognition and demonstrate its effectiveness.

## 2 Unconstrained Handwritten Numeral Data

A standard database is required to objectively evaluate the performance of a recognition system. The database we used in our experiment is the one which consists of numeral images extracted from the zip codes handwritten on U.S. mail envelopes (Suen et al., 1992). The database provides samples of totally unconstrained handwritten numerals from a large number of writers. There are some samples in the database

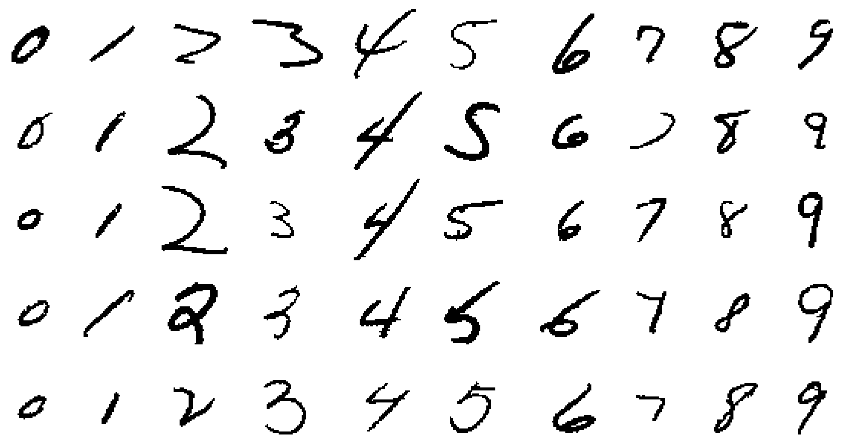


Fig. 1. The first 5 samples of each numeral in the training patterns

which are not easy even for humans to recognize. In an experiment to compare machine and human performance in the recognition of the confusing samples among the data, it was found that humans can recognize only about half of these samples (Suen et al., 1992).

Each numeral has 600 samples. Among them we used 400 samples for the training and 200 for the testing. Fig. 1 shows the first five samples of each numeral in the training patterns. As can be seen in the Fig. 1, the samples have different sizes. The width and the height of each sample are specified in the database, and all the data are binary. Since the size of a sample image varies, we first normalized each image into the size of $32 \times 32$ pixels.

## 3 Design of an RBF network

### 3.1 Architecture of an RBF network

An RBF network is a three layer feed-forward network that consists of one input layer, one hidden layer and one output layer as shown in Fig. 2. Each input neuron corresponds to a component of an input vector $\mathbf{x}$. The hidden layer consists of $n$ neurons and one bias neuron. Each input neuron is fully connected to the hidden layer neurons except the bias one. Each hidden layer neuron computes a kernel function (also called an activation function) which is usually the following Gaussian function:

$$y_i = \begin{cases} \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_i\|}{2\sigma_i^2}\right) & i = 1,2,\cdots,n \\ 1 & i = 0 \qquad \text{(bias neuron)} \end{cases} \tag{1}$$

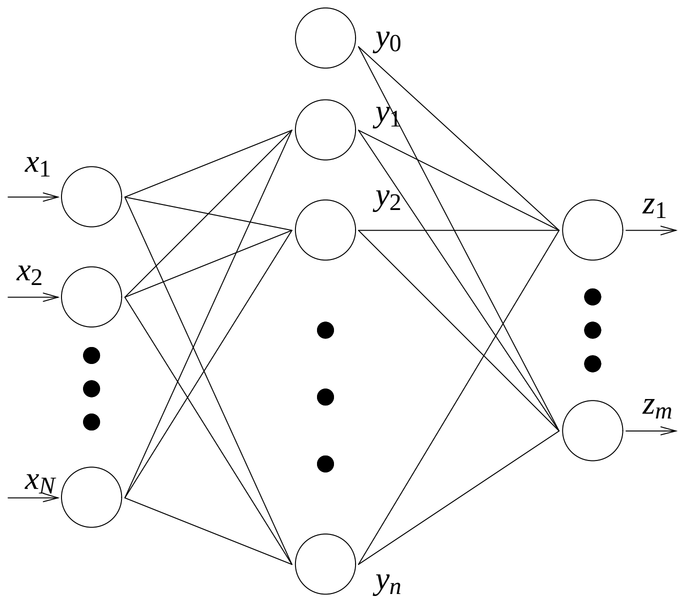


Fig. 2. Architecture of an RBF network

where $\mathbf{c}_i$ and $\sigma_i$ represent the center and the width of the neuron, respectively. $\|\cdot\|$ denotes the Euclidean distance. The weight vector between the input layer and the $i$-th hidden layer neuron corresponds to the center $\mathbf{c}_i$ in (1). And the net input to the $i$-th hidden layer neuron in an RBF network is $||\mathbf{x}-\mathbf{c}_i||$ rather than usual $\mathbf{x}\cdot\mathbf{c}_i$. The kernel function decreases rapidly if the width $\sigma_i$ is small, and slowly if it is large.

The output layer consists of $m$ neurons which correspond to the possible classes of the problem. Each output layer neuron is fully connected to the hidden layer and computes a linear weighted sum of the outputs of the hidden neurons as follows:

$$z_j = \sum_{i=0}^{n} y_i w_{ij}, \qquad j = 1, 2, \cdots, m \tag{2}$$

where $w_{ij}$ is the weight between the $i$-th hidden layer neuron and the $j$-th output layer neuron.

## 3.2 Construction of the input and output layer

Once the feature to be used is decided, the number of the neurons in the input layer is fixed. The number of the neurons in the output layer is also fixed when a problem is given. For the numeral recognition problem at hand, the number of the neurons in the output layer is 10 which is the number of numerals, while that in the input layer is the same as the dimension of the feature.

The feature we used in this experiment is *peripheral directional contributivity(PDC)* which was applied to Chinese character recognition and gave a high recognition rate(Hagita et al., 1983; Hildebrandt and Liu, 1993). The feature seems to well reflect the complexity, the directions, the connectivity, and the relative positional relationship of the strokes in a character.

The directional contributivity(DC) of a point inside a character is represented by a 4-dimensional vector. Each component of the vector represents the maximum line length through the point in one of the four directions, i.e. vertical, horizontal and two diagonal directions. The value of each component is normalized by: $d_i = l_i / \sqrt{\sum_i l_i^2}$ where $l_i$ is the original length in the direction $i$.

On the other hand if we move straight in one of the four directions from a point on the border of the rectangle image frame, we usually come across a point where 0 changes to 1. We call such a point as a peripheral point of the character. If we move further, we may hit another such peripheral point. We used, as the feature, those DCs for all first and second peripheral points (also called the peripheral points of depth 1 and 2) which we hit when we move in the four directions. In order to reduce the dimension and the shift variance, we divided the sequence of peripheral points in each direction into eight segments, and used the average of DC values of four peripheral points in each segment. Therefore the final dimension of the feature is $4 \times 4 \times 2 \times 8 = 256$ (no. of DCs $\times$ no. of move directions $\times$ no. of depths $\times$ no. of segments). Since the PDC feature used in our experiment has the dimension of 256, the number of neurons in the input layer is 256.

## 3.3 Construction of the hidden layer

In order to specify the hidden layer of an RBF network, we have to decide the number of neurons of the layer and their kernel functions which are usually Gaussian functions as mentioned above. In this paper we also use a Gaussian function as the kernel function. A Gaussian function is specified by its center and width. The simplest and most general way to decide the hidden layer neurons is to create a neuron for each training pattern. However this method is usually not practical since in most applications there are a large number of training patterns and the dimension of the input space is fairly large. Therefore it is usual and practical to first cluster the training patterns to a reasonable number of groups by using a clustering algorithm such as K-means and SOFM, and then to assign a neuron to each cluster. A simple way, though not usually optimal, is to choose a relatively small number of patterns randomly among the training patterns and create the neurons for these patterns.

Which method to be used to construct the hidden layer of an RBF network should be decided by considering the overall performance of the network. Since there is no known best method, here we use the widely used K-means clustering algorithm. It will be the subject of another research to develop a method to construct an optimal hidden layer.

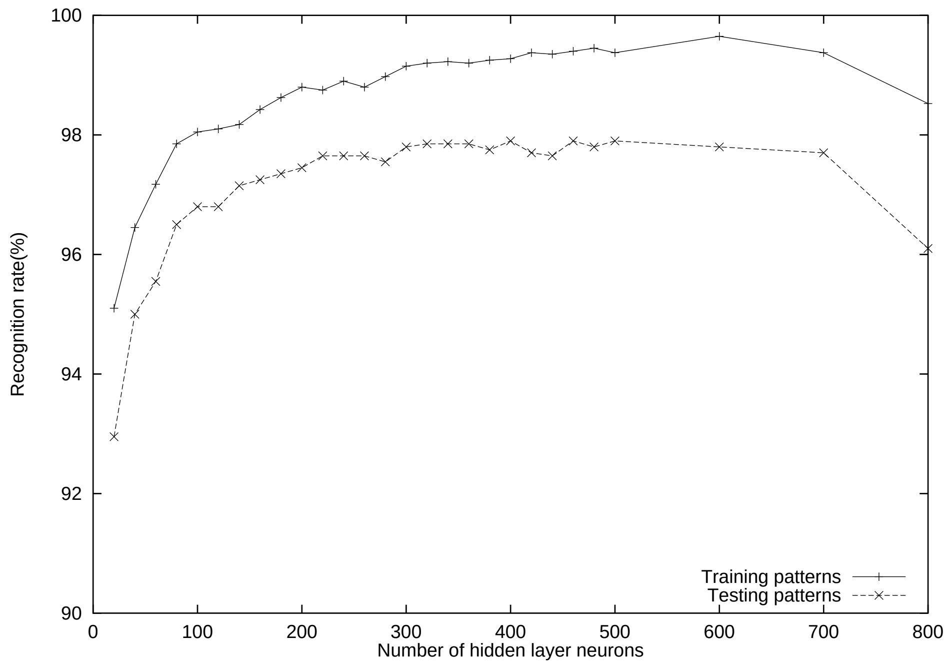


Fig. 3. Recognition rate by the number of hidden layer neurons

Originally a clustering algorithm is a kind of unsupervised learning algorithm and is used when the class of each training pattern is not known. But for the problem at hand the class of each training pattern is known. This fact alone, however, does not solve the problem of the clustering. But we may take advantage of the information of these class memberships in order to obtain a better clustering result or to do the job more efficiently(Moody and Darken, 1989; Musavi et al., 1992). Here we cluster the training patterns class by class instead of clustering the entire patterns at the same time. In this way we can reduce at least the total computation time required to cluster the entire training patterns since the number of patterns of each class is usually far less than that of the entire patterns. For the current problem we cluster the training patterns of each numeral. Then we create a hidden neuron for each cluster and set its center to that of the cluster.

In case of an RBF network, the number of hidden layer neurons affects its overall performance significantly. Usually the more in the hidden layer neurons, the higher the recognition rate up to a certain point. Fig. 3 shows the recognition rates when the number of hidden layer neurons changes. It should be noted that it does not go higher after the number is over 400 for the testing patterns. This phenomena may be related to the generalization of the network.

We have to decide the width of each kernel function in addition to its center as seen in (1). One can either set it to a pre-determined value, calculate it by a complex procedure (Musavi et al., 1992), or train it

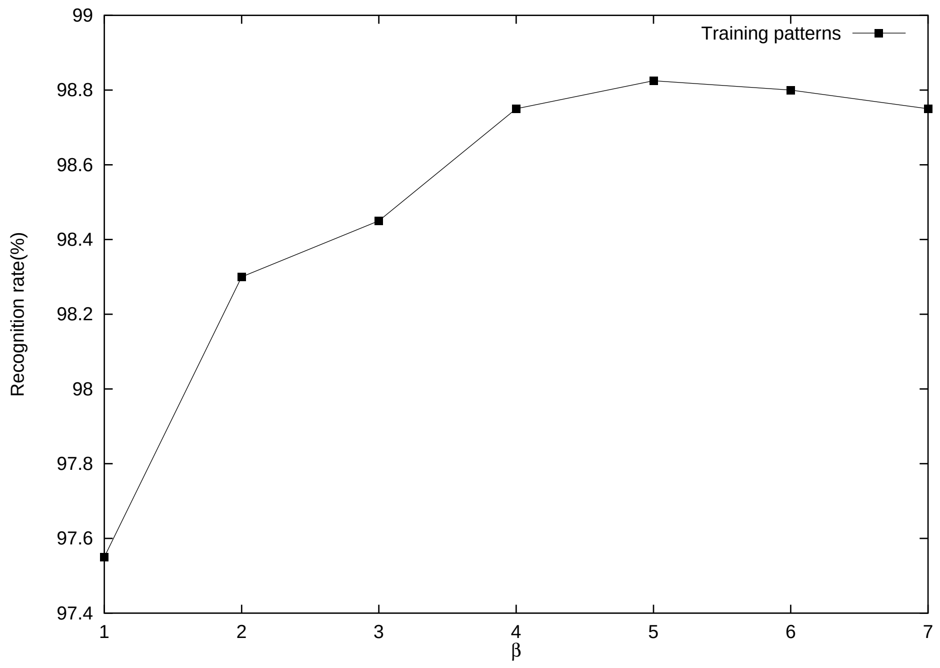


Fig. 4. Recognition rate by the value of β

by a gradient descent approach (Wettschereck and Dietterich, 1992). But the following heuristic is usually used (Moody and Darken, 1989):

> Find the distance between the center of a neuron and the center of the nearest other neuron and assign this value multiplied by β to the width of the neuron.

This heuristic would make the width of a hidden neuron small in a dense area of the clusters and large in a sparse area. It is desirable that the activation function of a hidden neuron does not overlap much with those of the neurons in the other classes although we do not care how closely it overlaps with those in the same class. In order to reflect this point we modified the above heuristic to the following:

> Find the distance between the center of a neuron and the center of the nearest neuron of the other classes and assign this value multiplied by β to the width of the neuron.

Fig. 4 shows the recognition rate by the value of β when the number of hidden layer neurons is 200. In our experiment we chose 5 as the value of β at which the recognition rate was the best.

## 3.4 Calculation of the Weights

The output of an output layer neuron is given by

$$z_i = \mathbf{w}_i^T \mathbf{y}, \qquad i = 1, 2, \cdots, m, \tag{3}$$

where $\mathbf{y}$ is an output of the hidden layer and $\mathbf{w}_i$ is the weight vector between the hidden layer and the $i$-th output layer neuron.

If the RBF network is used for classification, then the condition for the class $\omega_i$ to be chosen is

$$z_i > z_j, \qquad \forall j, j \neq i. \tag{4}$$

The LMS(least mean square) procedure (Devijver and Kittler, 1982) tries to find the discriminant functions which best satisfy the conditions:

$$z_i = \begin{cases} 1 & \text{for } \mathbf{x} \in \omega_i, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

For an $m$-class problem, let $\mathbf{V}_i$ designate the $i$-th column vector of an $m \times m$ identity matrix and $\mathbf{W}$ be the $(n+1) \times m$ matrix of weights :

$$W = [\mathbf{w}_1, \cdots, \mathbf{w}_m].$$

Then the criterion function to be minimized is

$$J(W) = \sum_{i=1}^{m} P_i E_i \left\{ \|W^T \mathbf{y} - V_i\|^2 \right\} \tag{6}$$

where $P_i$ and $E_i\{\}$ are the *a priori* probability and the expected value of class $\omega_i$, respectively.

Devijver and Kittler (1982) showed that the discriminant functions which minimize (6) are

$$z_i = \mathbf{w}_i^T \mathbf{y}, \qquad i = 1, 2, \cdots, m, \tag{7}$$

$$\mathbf{w}_i = K^{-1} M_i, \tag{8}$$

where

$$K = \sum_{i=1}^{m} K_i, \tag{9}$$

Table 2. Comparison with the recognition rates of other research teams using the same database

| Researcher(year) | Recognition rate(%) |
|---|---|
| Nadal('88) | 86.05 |
| Lam('88) | 93.10 |
| Legault('89) | 93.90 |
| Mai('90) | 92.95 |
| Kryzak('90) | 94.85 |
| Lee('95) | 97.80 |
| Proposed method | 97.9 |

$$K_i = E_i\{\mathbf{y}\mathbf{y}^T\}. \tag{10}$$

In the above, $M_i$ is the mean vector of class $\omega_i$.

In the above, one have to obtain the inverse of the covariance matrix to find the weights. We calculated it by using the program of the LU decomposition method found in the book: *Numerical Recipes in C*(Press et al., 1992).

## 4 Performance Test

We evaluated the performance of the RBF network so designed by following the procedure described in the above. We obtained the best recognition rate of 97.9% when the number of hidden layer neurons is 400 as can be seen in Fig. 3. For comparison, Table 2 gives our result along with those obtained by the other research teams (Suen et al., 1992; Lee, 1995) who used the same database as ours. These results are all for the testing patterns.

It may be questioned that the high recognition rate may be due to the feature we used. In order to clarify this point, we performed another experiment in which we instead used a simple MDC with the same input feature. The result was the recognition rate of 89.1% for the training patterns and 87.9% for the testing. This implies that the feature we used was good but not a definitive factor for the good result.

We also conducted an experiment in order to compare the proposed RBF network classifier with K-NN and LVQ classifier. using the same input feature and the same clusters. As seen in Fig. 5, the proposed RBF network classier was consistently better than the others.

Table 3 shows the confusion matrix for our testing data when the recognition rate is best, and Fig. 6 shows all the misclassified data among the testing patterns. As seen in the figure some of them are really hard to correctly recognize. For example, the 12th image of Fig. 5 is actually the numeral 2 but was misread as 5. The reason for the misclassification of the 22nd image is that the image has two numerals 2 and 3.

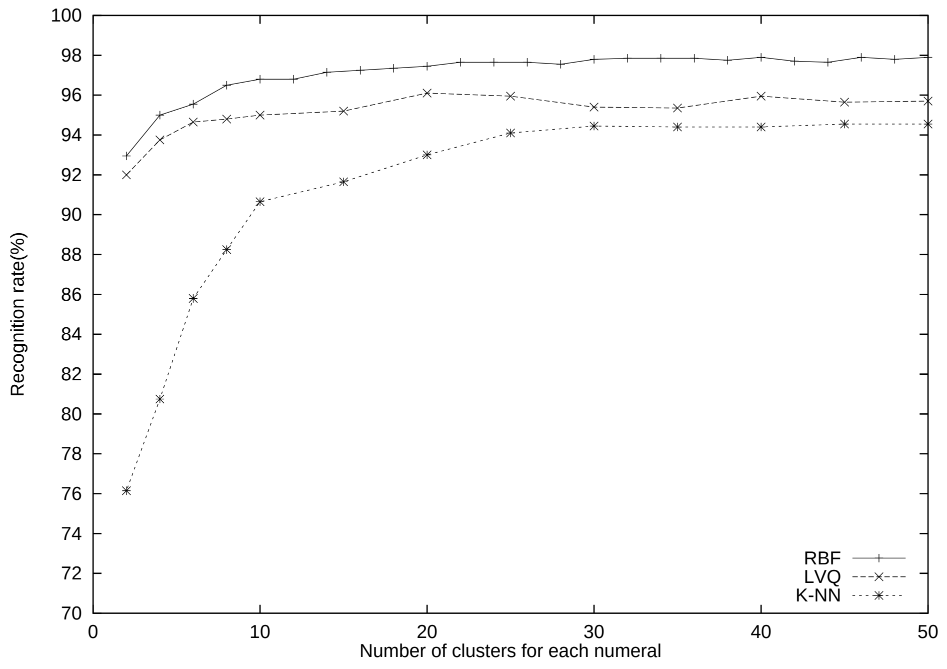


Fig. 5. Comparison with K-NN and LVQ classifiers

Table 3. Confusion matrix for the testing patterns

| Digit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Recognition rate(%) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 198 | | 1 | | | | 1 | | | | 99.0 |
| 1 | | 200 | | | | | | | | | 100.0 |
| 2 | 1 | | 190 | 3 | 1 | 1 | | 1 | 3 | | 95.0 |
| 3 | | | 5 | 188 | | 2 | | | 4 | 1 | 94.0 |
| 4 | | | | | 196 | | 3 | | | 1 | 98.0 |
| 5 | 1 | 1 | | 4 | | 193 | | | 1 | | 96.5 |
| 6 | 1 | | | | | | 199 | | | | 99.5 |
| 7 | | | | | 3 | | | 196 | | 1 | 98.0 |
| 8 | | | 1 | | | | | | 199 | | 99.5 |
| 9 | | | | | | | | | 1 | 199 | 99.5 |

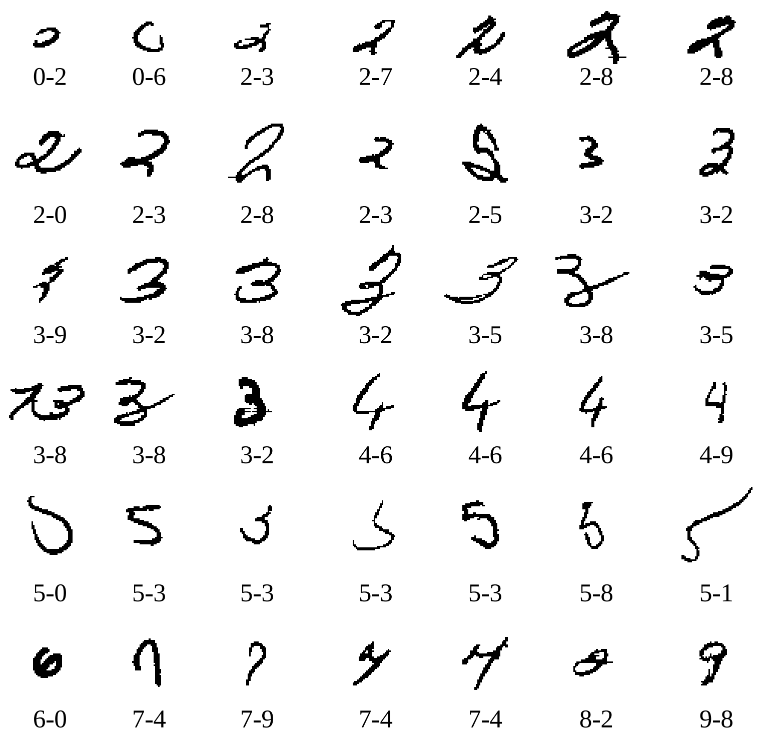


Fig. 6. List of all the misclassified testing patterns

## 5 Conclusion

While there have been proposed many character recognition methods using neural network, there are few using an RBF network which actually has a powerful pattern recognition ability. This fact seems to be due to the situation that an easy but effective procedure to design a good RBF network to solve a given problem has not been known. Earlier we proposed such a procedure. In this paper we actually applied the procedure and designed an RBF network to solve the problem of unconstrained handwritten numeral recognition.

By using a real database of unconstrained handwritten numerals we described how we designed an RBF network step by step. Then we evaluated the performance of the RBF network so designed. The result showed that the recognition rate of the RBF network developed by our method is better than those by the other recognition systems reported so far which had used the same database. We also found that our RBF network classifier performed better than K-NN and LVQ classifier when they used the same input feature and the same clusters.

The contribution of the current paper is twofold. First, we showed how to design an RBF network for a handwritten numeral recognition which resulted in a very competitive performance. Second, as seen in the description of the procedure, the proposed method is general enough to be applied to any other classification problems. So this paper will help make it possible for an RBF network to be more widely and easily used for pattern recognition problems.

## References

Devijver, P. and Kittler, J. (1982). *Pattern Recognition A Statistical Approach*. Prentice-Hall.

Hagita, N., Naito, S., and Masuda, I. (1983). Handprinted chinese characters recognition by peripheral direction contributivity feature. *The Transactions of the IEICE D*, 66(10):1185–1192. (in Japanese).

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Macmillan.

Hildebrandt, T. H. and Liu, W. (1993). Optical recognition of handwritten Chinese characters: Advances since 1980. *Pattern Recognition*, 26(2):205–225.

Hwang, Y.-S. and Bang, S.-Y. (1997). An efficient method to construct a radial basis function neural network classifier. *Neural Networks*. to appear.

Lee, S.-W. (1995). Multilayer cluster neural network for totally unconstrained handwritten numeral recognition. *Neural Networks*, 8(5):783–792.

Lee, Y. (1991). Handwritten digit recognition using K nearest-neighbor, radial-basis function, and backpropagation neural networks. *Neural Computation*, 3:440–449.

Monica Bianchini, P. F. and Gori, M. (1995). Learning without local minima in radial basis function networks. *IEEE Transactions on Neural Networks*, 6(3):749–756.

Moody, J. and Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–294.

Musavi, M., Ahmed, W., Chan, K., Faris, K., and Hummels, D. (1992). On the training of radial basis function classifiers. *Neural Networks*, 5:595–603.

Ng, K. and Lippmann, R. P. (1991). A comparative study of the practical characteristics of neural network and conventional pattern classifiers. In Lipmann, R. P. and Moody, J., editors, *Advances in Neural Information Processing Systems*, volume 3, pages 970–976. Morgan Kaufman.

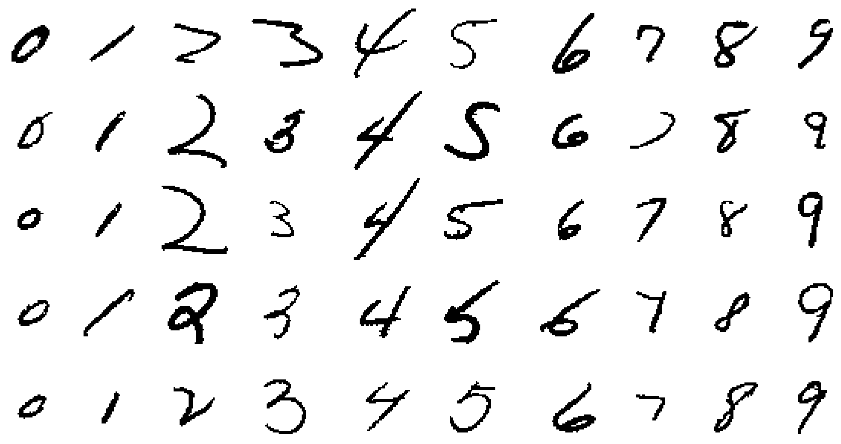
Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, 78:1481–1497.

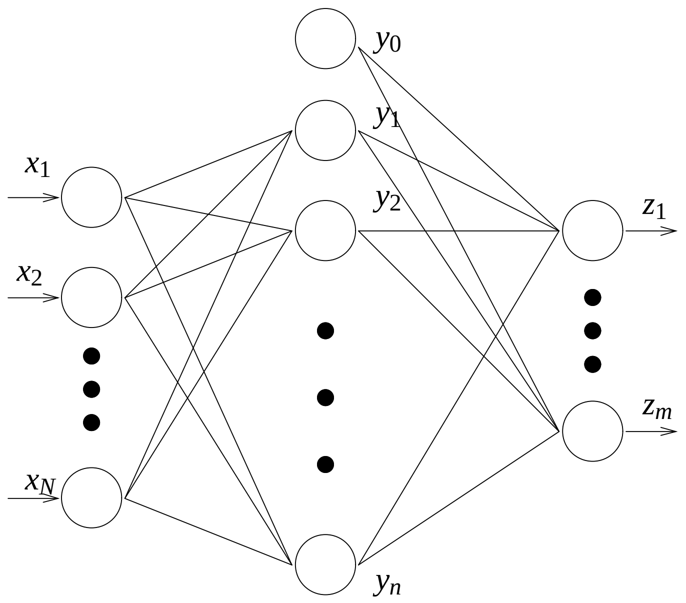
Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C: The Art of Scientific Computing*, chapter 2. Second ed. Cambridge University Press.

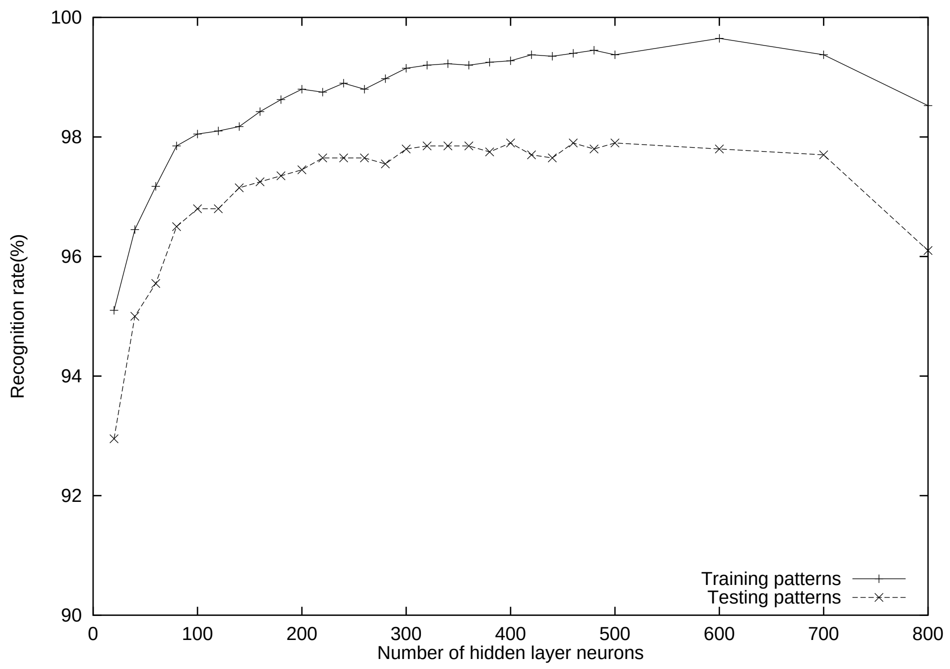
Suen, C., Nadal, C., Mai, T., Legault, R., and Lam, L. (1992). Computer recognition of unconstrained handwritten numerals. *Proceedings of the IEEE*, 80(7):1162–1189.

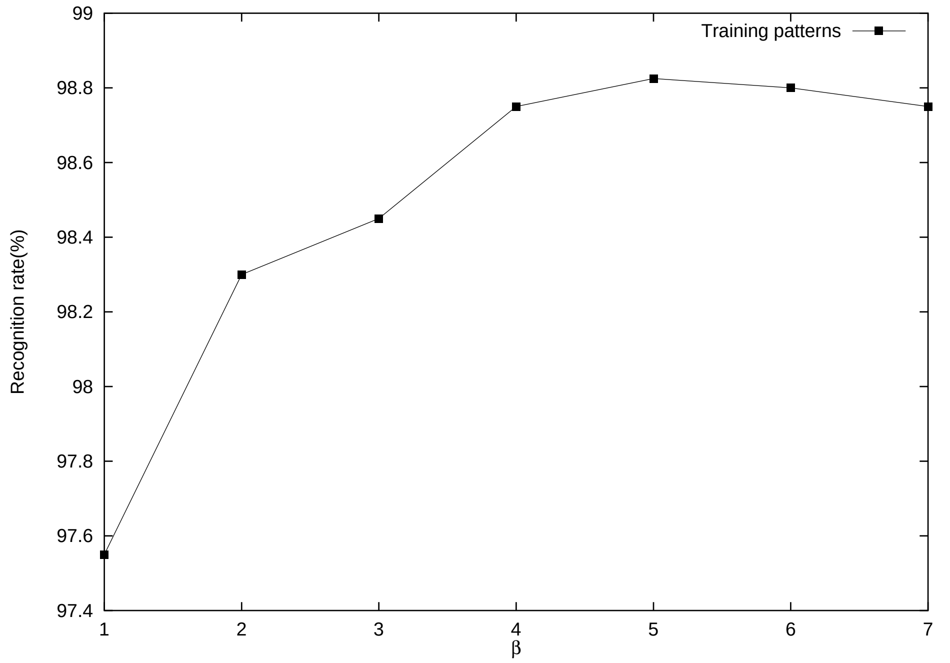
Vogt, M. (1993). Combination of radial basis function neural networks with optimized learning vector quantization. In *ICNN'93*, pages 1841–1846.

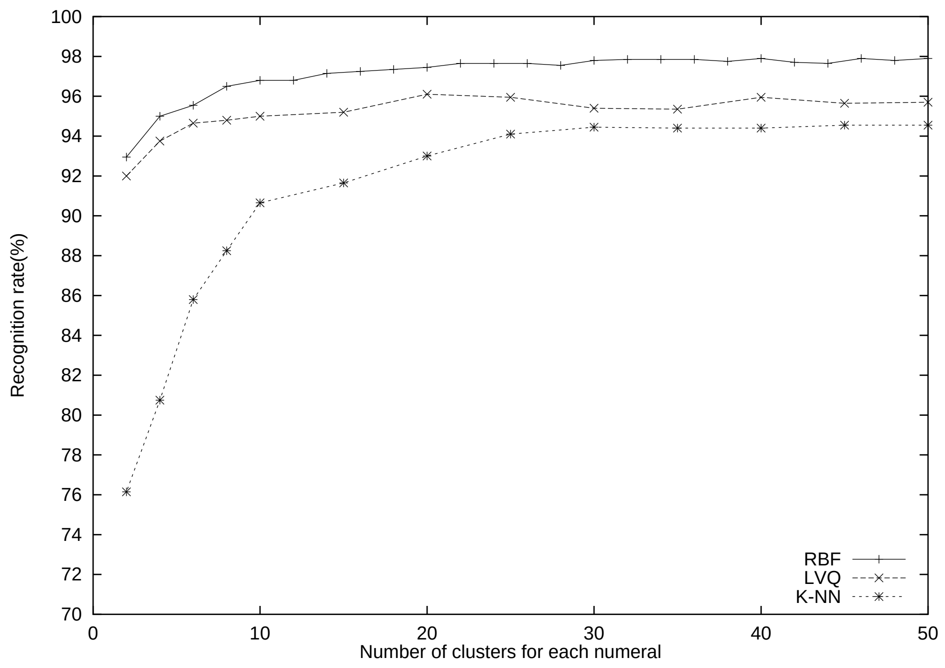
Wettschereck, D. and Dietterich, T. (1992). Improving the performance of radial basis function networks by learning center locations. In *Advances in Neural Information Processing Systems*, volume 4, pages 1133 –1140. Morgan Kaufman.

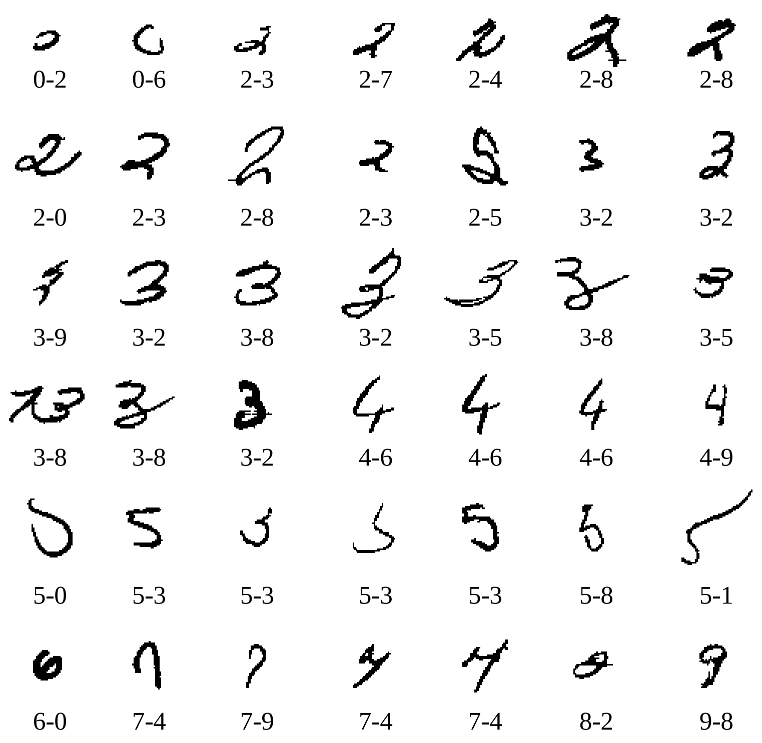



$y_0$
$y_1$
$y_2$
$y_n$
$x_1$
$x_2$
$x_N$
$z_1$
$z_m$


Recognition rate(%)
Number of hidden layer neurons
Training patterns
Testing patterns
100
98
96
94
92
90
0
100
200
300
400
500
600
700
800


Recognition rate(%)
β
Training patterns
99
98.8
98.6
98.4
98.2
98
97.8
97.6
97.4
1
2
3
4
5
6
7


Recognition rate(%)
Number of clusters for each numeral
RBF
LVQ
K-NN
100
98
96
94
92
90
88
86
84
82
80
78
76
74
72
70
0
10
20
30
40
50


0-2
0-6
2-3
2-7
2-4
2-8
2-8
2-0
2-3
2-8
2-3
2-5
3-2
3-2
3-9
3-2
3-8
3-2
3-5
3-8
3-5
3-8
3-8
3-2
4-6
4-6
4-6
4-9
5-0
5-3
5-3
5-3
5-3
5-8
5-1
6-0
7-4
7-9
7-4
7-4
8-2
9-8

| Researcher(year) | Recognition rate(%) |
|---|---|
| Nadal('88) | 86.05 |
| Lam('88) | 93.10 |
| Legault('89) | 93.90 |
| Mai('90) | 92.95 |
| Kryzak('90) | 94.85 |
| Lee('95) | 97.80 |
| Proposed method | 97.9 |

| Classifier | Training time | Hardware requirement | Classification time |
|---|---|---|---|
| RBF[a] | middle | middle | middle |
| BP[b] | long | small | short |
| K-NN[c] | none | large | long |
| LVQ[d] | long | small | middle |
| MDC[e] | short | small | short |

[a] Radial basis function network
[b] Error back propagation network
[c] K nearest neighbor
[d] Learning vector quantization
[e] Minimum distance classifier

| Digit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Recognition rate(%) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 198 |  | 1 |  |  |  | 1 |  |  |  | 99.0 |
| 1 |  | 200 |  |  |  |  |  |  |  |  | 100.0 |
| 2 | 1 |  | 190 | 3 | 1 | 1 |  | 1 | 3 |  | 95.0 |
| 3 |  |  | 5 | 188 |  | 2 |  |  | 4 | 1 | 94.0 |
| 4 |  |  |  |  | 196 |  | 3 |  |  | 1 | 98.0 |
| 5 | 1 | 1 |  | 4 |  | 193 |  |  | 1 |  | 96.5 |
| 6 | 1 |  |  |  |  |  | 199 |  |  |  | 99.5 |
| 7 |  |  |  |  | 3 |  |  | 196 |  | 1 | 98.0 |
| 8 |  |  | 1 |  |  |  |  |  | 199 |  | 99.5 |
| 9 |  |  |  |  |  |  |  |  | 1 | 199 | 99.5 |