ELSEVIER

Fuzzy Sets and Systems 103 (1999) 339–347

FUZZY sets and systems

# Pattern recognition with neural networks combined by genetic algorithm

Sung-Bae Cho[a,b,*]

[a] *Department of Computer Science, Yonsei University, 134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, South Korea*
[b] *ATR Human Information Processing Research Laboratories, 2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan*

Received May 1998

**Abstract**

Soft computing techniques have been recently exploited as a promising tool for achieving high performance in pattern recognition. This paper presents a hybrid method which combines neural network classifiers by genetic algorithm. Genetic algorithm gives us an effective vehicle to determine the optimal weight parameters that are multiplied by the network outputs as coefficients. The experimental results with the recognition problem of totally unconstrained handwritten numerals show that the genetic algorithm produces better results than the conventional methods such as averaging and Borda count.


*Keywords:* Soft computing; Combining neural networks; Genetic algorithm; Handwritten digit recognition

## 1. Introduction

Neural networks have been recognized as a powerful tool for pattern recognition problems. They have strong points in the discriminative power and the capability to learn and represent implicit knowledge. Recently, the concept of combining neural networks with other soft computing tools has been spotlighted as a new direction for the development of high performance systems in the area of pattern recognition [1,5]. While the usual scheme chooses one best network from amongst the set of candidate networks, this approach keeps multiple networks and runs them all with an appropriate collective decision strategy.

Several methods for combining evidence produced by multiple information sources have been widely applied in statistics, management sciences, and pattern recognition [1,25]. A general result from the previous works is that averaging separate networks improves generalization performance with respect to the mean squared error [19]. If we have networks of different accuracy, however, it is obviously not good to take their simple average or simple voting.

In our previous work we have proposed a series of methods based on fuzzy logic, which combines the outputs of separate networks with importance of each network. The importance is subjectively assigned [2–5]. This paper presents another method based on genetic algorithm, which also considers the difference of performance of each network in combining the networks. This method utilizes the weight parameters, which are determined by genetic algorithm, to obtain the combined output. We demonstrate the superiority

* Correspondence address: Department of Computer Science, Yonsei University, 134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, South Korea.

0165-0114/99/$ – see front matter 
PII: S0165-0114(98)00232-2

in performance of the proposed method and compare with conventional methods by thorough experiments in a real-world pattern recognition problem. In view of the simplicity of the approach, the reported performance is remarkable and can stand comparison with the best results reported in the literature.

The rest of this paper is organized as follows. Section 2 formulates the problem and considers possible methods for combining neural networks. In Section 3, we present the combining methods based on genetic algorithm, and the results with the recognition problem of totally unconstrained handwritten numerals are shown in Section 4.

## 2. Problem formulation

A feedforward neural network can be considered as a mapping device between an input set and an output set: It represents a function $f$ that maps $I$ into $O$; $f: I \to O$, or $y = f(x)$ where $y \in O$ and $x \in I$. Since the classification problem is a mapping from the feature space to some set of output classes, we can formalize the neural network, especially two-layer feedforward neural network trained with the generalized delta rule, as a classifier.

Consider a two-layer neural network classifier with $T$ neurons in the input layer, $H$ neurons in the hidden layer, and $c$ neurons in the output layer. Here, $T$ is the number of features, $c$ is the number of classes, and $H$ is an appropriately selected number. The network is fully connected between adjacent layers. The operation of this network can be thought of as a nonlinear decision-making process: Given an input $X = (x_1, x_2, \ldots, x_T)$ with unknown class and the class set $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$, each output neuron produces $y_i$ of belonging to this class by

$$P(\omega_i \mid X) \approx y_i = f\left\{\sum_{k=1}^{H} w_{ik}^{\mathrm{om}} f\left(\sum_{j=1}^{T} w_{kj}^{\mathrm{mi}} x_j\right)\right\}, \quad (1)$$

where $w_{kj}^{\mathrm{mi}}$ is a weight between the $j$th input neuron and the $k$th hidden neuron, $w_{ik}^{\mathrm{om}}$ is a weight from the $k$th hidden neuron to the $i$th class output, and $f$ is a sigmoid function such as $f(x) = 1/(1 + \mathrm{e}^{-x})$. The superscripts i, m and o in the weights are used to specify the input, middle (hidden) and output layers,

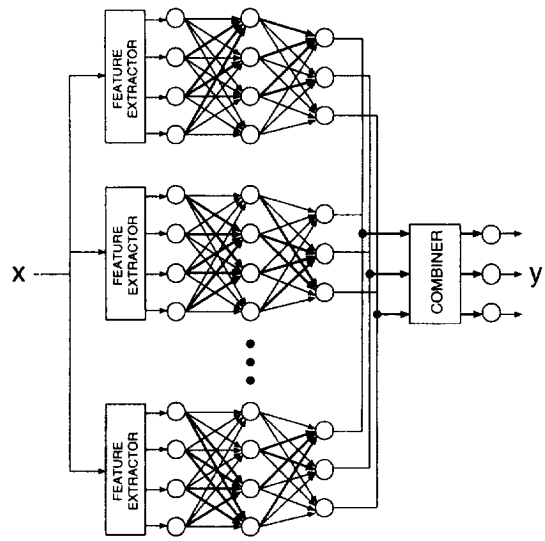



Fig. 1. The multiple neural networks combined by fusion method.

respectively. The neuron having the maximum value is selected as the corresponding class.

The network presented above trains on a set of example patterns and discovers relationships that distinguish the patterns. A network of a finite size, however, does not often load a particular mapping completely or it generalizes poorly. Increasing the size and number of hidden layers most often does not lead to any improvements. Furthermore, in complex problems such as character recognition, both the number of available features and the number of classes are large. The features are neither statistically independent nor unimodally distributed.

The basic idea of the multiple network scheme is to develop $n$ independently trained neural networks with relevant features, and to classify a given input pattern by utilizing combination methods to decide the collective classification [7,22]. Fig. 1 shows the schematic diagram of the multiple network scheme. The combiner aggregates the results from the neural networks trained with different features. Then it naturally raises the question of how to obtain a consensus from the results of each individual network or expert.

There might be two general approaches for combining the multiple neural networks: One is based on fusion technique and the other on voting technique. In the methods based on the fusion technique, the classification of an input $X$ is actually based on a set of

real value measurements:

$$P(\omega_i \mid X), \quad 1 \leqslant i \leqslant c. \tag{2}$$

They estimate the probabilities[1] that $X$ come from each of the $c$ classes. In the combined network scheme consisting of $n$ networks, each network $k$ estimates by itself a set approximations of those true values as follows:

$$P_k(\omega_i \mid X), \quad 1 \leqslant i \leqslant c, \quad 1 \leqslant k \leqslant n. \tag{3}$$

One simple approach to combine the results on the same $X$ by all $n$ networks is to use the following average value as a new estimation of combined network:

$$P(\omega_i \mid X) = \frac{1}{n} \sum_{k=1}^{n} P_k(\omega_i \mid X), \quad 1 \leqslant i \leqslant c. \tag{4}$$

We can think of such a combined value as an averaged Bayes classification. This estimation will be improved if we give the judge the ability to bias the outputs based on a priori knowledge about the reliability of the networks:

$$P(\omega_i \mid X) = \sum_{k=1}^{n} r_k^i P_k(\omega_i \mid X), \quad 1 \leqslant i \leqslant c, \tag{5}$$

$$\text{where } \sum_{k=1}^{n} r_k^i = 1. \tag{6}$$

The other method based on voting techniques considers the result of each network as an expert judgement. A variety of voting procedures can be adopted from group decision making theory: unanimity, majority, plurality, Borda count [8], and so on. In particular, we shall introduce the Borda count which has been reported as the best method among voting techniques.

For any particular class $i$, the Borda count is the sum of the number of classes ranked below class $i$. Unlike the usual voting techniques, it produces the value based on ranks. For instance, consider a three-class classifier for a particular input $x$ producing $y_1 = 0.3$, $y_2 = 0.5$, and $y_3 = 0.1$. Then the Borda counts for the three classes are 1, 2 and 0 because the class 2 ranks the top. As can be seen, the larger the value is the higher the rank is.

[1] The outputs of neural networks are not just likelihoods or binary logical values near zero or one. Instead, they are estimates of Bayesian a posteriori probabilities of a classifier [21].

Let $B_k(i)$ be the number of classes ranked below the class $i$ by the $k$th network. Then, the Borda count for class $i$ becomes

$$F(\omega_i \mid X) = \sum_{k=1}^{n} B_k(i), \quad 1 \leqslant i \leqslant c. \tag{7}$$

Notice that we cannot consider it as a sort of probability. Like the average case, we can extend it to the weighted Borda count by considering the reliability of the neural networks in the combination:

$$F(\omega_i \mid X) = \sum_{k=1}^{n} r_k^i B_k(i), \quad 1 \leqslant i \leqslant c, \tag{8}$$

$$\text{where } \sum_{k=1}^{n} r_k^i = 1. \tag{9}$$

The final decision is given by selecting the class label of which the Borda count is the largest.

## 3. Genetic algorithm based method

This section describes some of the basic mechanisms of the evolutionary computation with the genetic algorithm (GA) and the method based on the GA to combine neural networks.

### 3.1. Overview of genetic algorithm

Evolution is a remarkable problem-solving machine [23]. First proposed by John Holland in 1975, GA is an attractive class of computational models that mimic natural evolution to solve problems in a wide variety of domains. A GA emulates some of the processes of natural genetic systems to solve optimization problems. In common with all other search algorithms, a GA performs a search on a multidimensional space containing a hypersurface known as fitness surface.

The basis of a GA is that a population of problem solutions is maintained in the form of chromosomes, which are strings encoding problem solutions. Strings can be binary or have many possible alternatives (genes) at each position. The strings are converted into problem solutions, which are then evaluated according to an objective scoring function. It is important that this is a good estimate, otherwise the

selective pressure that favors truly high scoring chromosomes can be lost in the noise caused by poor fitness estimates.

Following fitness evaluation, a new population of chromosomes is generated by applying a set of genetic operators to the original population. The basic genetic operations are selection, crossover and mutation. The selection process copies parent chromosomes into a tentative new population. The number of copies reproduced for the next generation by an individual is expected to be directly proportional to its fitness value. The crossover recombines genetic material of two parent chromosomes to produce offspring on the basis of crossover probability. Provided that two chromosomes were $a=(1\,1\,1\,1)$ and $b=(0\,0\,0\,0)$, one-point crossover at the third point produces two new chromosomes, $a'=(1\,1\,0\,0)$ and $b'=(0\,0\,1\,1)$. Finally, the mutation selects a random position of a random string and negates the bit value. For instance, if mutation is applied to the fourth bit of string $a'$, the transformed string becomes $(1\,1\,0\,1)$. This process continues until an acceptable solution is found.

In summary, a GA comprises a set of individual elements (the population) and a set of biologically inspired operators defined over the population itself. According to evolutionary theories, only the most suited elements in a population are likely to survive and generate offspring, thus transmitting their biological features to new generations.

### 3.2. Genetic algorithm for combining networks

In computing terminologies, a genetic algorithm maps a problem onto a set of strings, each string representing a potential solution. In our problem, a string must encode $n \times c$ real-valued parameters, $r_k^i$, in Eq. (5), thereby optimal combination coefficients for combining neural networks can be obtained. Each coefficient is encoded by 8 bits and scaled between $[0 \sim 1]$. The GA then manipulates the most promising strings in its search for improved solutions. A GA operates through a simple cycle of stages:

1. creation of a population of real-valued strings,
2. evaluation of each string with recognition rate on training data,
3. selection of good strings, and
4. genetic manipulation with crossover and mutation to create the new population of strings.

The cycle stops when the recognition rate gets better no longer. Notice that we replace all the members of old population with the new ones, and preserve the best possible solution obtained so far by elitist strategy. Fig. 2a shows these four stages using the biologically inspired terminology. The GA approach to our problem takes pieces of weighting coefficients to combine neural networks as such strings as shown in Fig. 2b.

## 4. Experimental results

### 4.1. Environments

In the experiments, we have used the handwritten digit database of Concordia University of Canada, which consists of 6000 unconstrained digits originally collected from dead letter envelopes by the US Postal Services at different locations in the US. The digits of this database were digitized in bilevel on a $64 \times 224$ grid of 0.153 mm square elements, giving a resolution of approximately 166 PPI [24]. Among the data, 4000 digits were used for training and 2000 digits for testing. Fig. 3 shows some representative samples taken from the database. We can see that many different writing styles are apparent, as well as digits of different sizes and stroke widths.

Digits, whether handwritten or typed, are essentially line drawings, i.e., one-dimensional structures in a two-dimensional space. Thus, local detection of line segments seems to be an adequate feature extraction method. For each location in the image, information about the presence of a line segment of a given direction is stored in a feature vector [10]. Especially, in this paper Kirsch masks [20] have been used for extracting directional features.

Kirsch defined a nonlinear edge enhancement algorithm as follows:

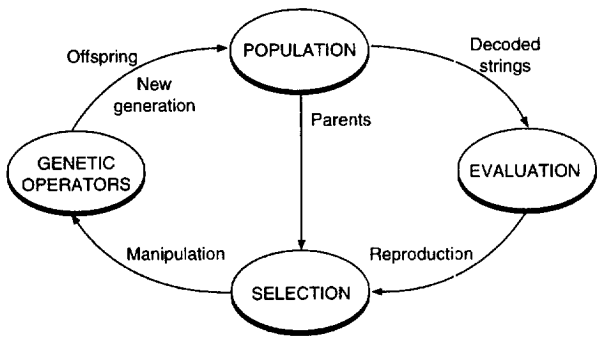
$$G(i,j) = \max\left\{1,\ \max_{k=0}^{7}[|5S_k - 3T_k|]\right\}, \tag{10}$$

where

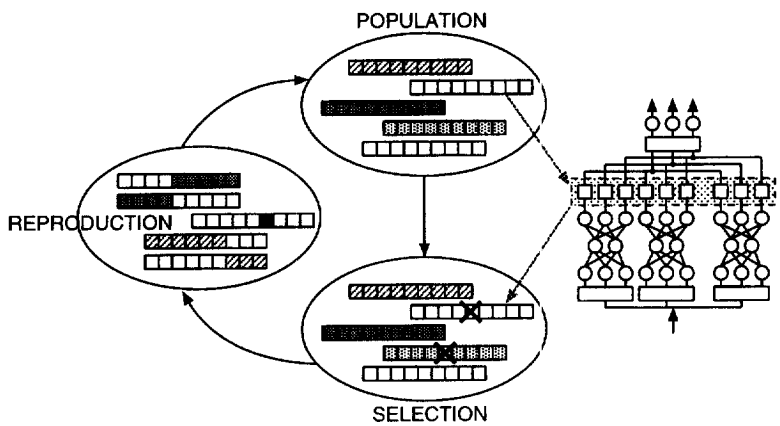
$$S_k = A_k + A_{k+1} + A_{k+2}, \tag{11}$$

$$T_k = A_{k+3} + A_{k+4} + A_{k+5} + A_{k+6} + A_{k+7}. \tag{12}$$

Here, $G(i,j)$ is the gradient of pixel $(i,j)$, the subscripts of $A$ are evaluated modulo 8, and $A_k$




(a)




(b)

Fig. 2. (a) A typical procedure for genetic algorithm; (b) overall scheme of the procedure for the proposed method.

$(k=0,1,\ldots,7)$ is eight neighbors of pixel $(i,j)$ as shown in Fig. 4.

Since the data set was prepared by thorough preprocessing, in this paper, each digit is scaled to fit in a $16 \times 16$ bounding box such that the aspect ratio of the image is preserved. Then, feature vectors for horizontal, vertical, right-diagonal, and left-diagonal directions are obtained from the scaled image as proposed by [9].

$$G_V(i,j) = \max(|5S_2 - 3T_2|, |5S_6 - 3T_6|),$$

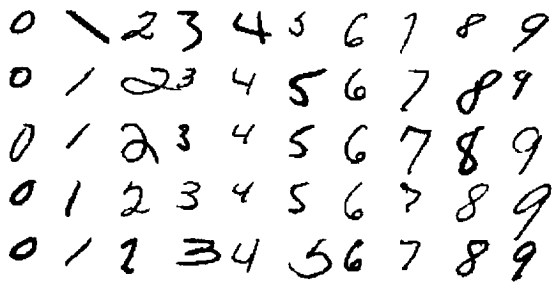
$$G_H(i,j) = \max(|5S_0 - 3T_0|, |5S_4 - 3T_4|),$$

$$G_L(i,j) = \max(|5S_3 - 3T_3|, |5S_7 - 3T_7|),$$

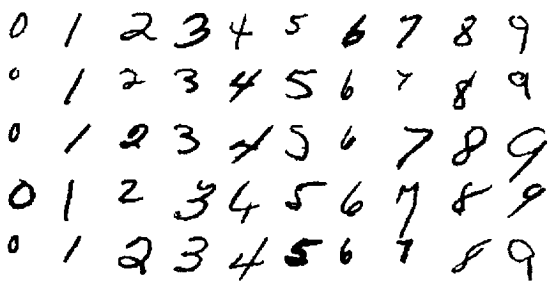
$$G_R(i,j) = \max(|5S_1 - 3T_1|, |5S_5 - 3T_5|).$$

The final step in extracting the features compresses each $16 \times 16$ directional vector into $4 \times 4$ vector with averaging operator, which produces a value for $2 \times 2$ pixels with dividing by 4 the value obtained by summing the four values. Moreover, $4 \times 4$ compressed image can be considered as a good candidate for global features.

In addition to those two features, we have also used a contour feature which is one of the most effective to represent the information of pattern boundary. After extracting contour from a size-normalized image, eight directional chain codes are obtained. 128 histograms for chain codes in $4 \times 4$ subregions are finally generated for a digit pattern. As a result, available features include four $4 \times 4$ local features, one $4 \times 4$ global features, and structural features extracted from the contours of the digits.



(a)



(b)

Fig. 3. Sample data for (a) training; (b) test.

| | | |
|---|---|---|
| $A_0$ | $A_1$ | $A_2$ |
| $A_7$ | $(i, j)$ | $A_3$ |
| $A_6$ | $A_5$ | $A_4$ |

Fig. 4. Definition of eight neighbors $A_k$ $(k = 0, 1, \ldots, 7)$ of pixel $(i, j)$.

### 4.2. Results

To evaluate the performance of the combining methods, we have implemented three different networks, each of which is a two-layer neural network using different features. $NN_1$, $NN_2$ and $NN_3$ are the networks using normalized image, Kirsch features, and sequence of contour features, respectively. In this way each network makes the decision through its own criterion. Each of the three networks was trained with 4000 samples, and tested on 2000 samples from the Concordia database.

The error backpropagation algorithm was used for the training and the iterative estimation process was stopped when an average squared error of 0.9 over the training set was obtained, or when the number of iteration reaches 1000, which was adopted mainly for preventing networks from overtraining. The parameter values used for training were: learning rate is 0.4 and momentum parameter is 0.6. An input vector is classified as belonging to the output class associated with the highest output activation.

After training all the three neural networks with different features, GA is used to obtain the optimal parameters to combine them. Initial population is 100 individuals, each of which consists of 240 bits $(3 \times 10 \times 8)$. The evolution parameters used in this experiment are as follows: 0.6 one-point crossover rate and differing mutation rates. A fitness value is assigned to a string by testing the recognition rate with the trained digits. The cycle of evolutionary process stops when the best fitness of the population does not improve any more for a long while.

Fig. 5 shows the fitness changes as generation goes with respect to two different mutation rates.[2] As the figure dictates, it is not appropriate to choose the mutation rate as larger than 1%, and we used the genetic algorithm with 1% mutation rate. Fig. 6 shows the best and average fitness changes in 1% mutation rate. As the figure depicts, it is clear that the performance increases as the generation goes and after the initial radical improvements the overall fitness settles down soon.

Table 1 reports the recognition rates with respect to the three different networks and their combinations by utilizing combining methods such as (weighted) average, (weighted) Borda count, and the proposed method. $NN_{all}$ here means the network trained with all the features that used to train the three networks.

The reliability in the table is computed as the following equation:

$$\text{Reliability} = \frac{\text{Correct}}{\text{Correct} + \text{Error}}. \tag{13}$$

As can be seen, any method of combining neural networks produces better results than individual networks, and the overall classification rate for the

[2] An extensive work on different parameters was performed, and in the text only the typical result has been given.

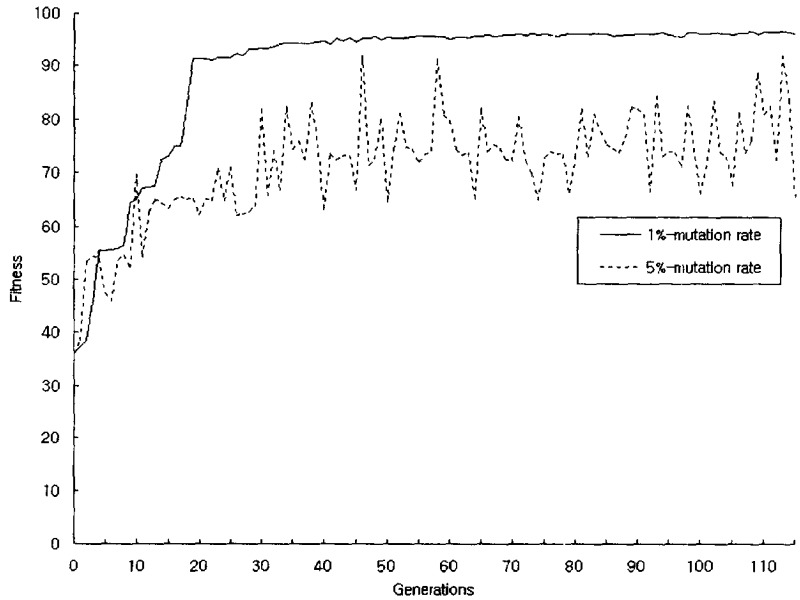



Fig. 5. Fitness changes with respect to different mutation rates.

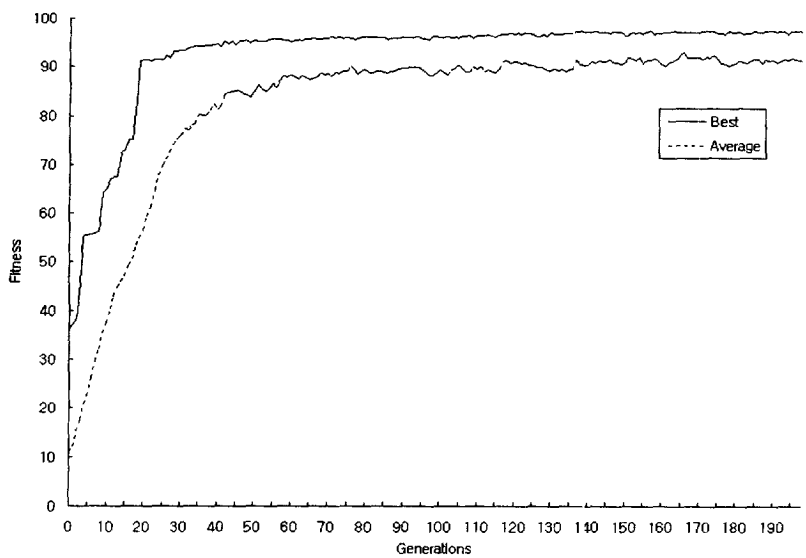



Fig. 6. Best and average fitness changes as generation goes.

Table 1
The result of recognition rates (%). Here "W-" means "weighted"

| Methods | Correct | Error | Reject | Reliability |
|---|---|---|---|---|
| $NN_1$ | 89.05 | 7.00 | 3.95 | 92.71 |
| $NN_2$ | 95.40 | 3.75 | 0.85 | 96.22 |
| $NN_3$ | 93.95 | 4.10 | 1.95 | 95.82 |
| $NN_{all}$ | 95.85 | 4.15 | 0.00 | 95.85 |
| Average | 97.15 | 2.35 | 0.50 | 97.64 |
| W-average | 97.35 | 2.30 | 0.35 | 97.70 |
| Borda | 96.70 | 3.05 | 0.25 | 96.94 |
| W-Borda | 97.35 | 2.50 | 0.15 | 97.50 |
| Genetic | 97.90 | 2.10 | 0.00 | 97.90 |

Table 2
Comparisons of the presented method with the related (%)

| Methods | Correct | Error | Reject |
|---|---|---|---|
| Lam et al. [12] | 93.10 | 2.95 | 3.95 |
| Nadal et al. [18] | 86.05 | 2.25 | 11.70 |
| Legault et al. [14] | 93.90 | 1.60 | 4.50 |
| Krzyzak et al. [11] | 94.85 | 5.15 | 0.00 |
| Mai et al. [16] | 92.95 | 2.15 | 4.90 |
| Suen et al. [24] | 93.05 | 0.00 | 6.95 |
| Le Cun et al. [13] | 96.40 | 3.40 | 0.20 |
| Martin et al. [17] | 96.00 | 4.00 | 0.00 |
| Cohen et al. [6] | 97.10 | 2.90 | 0.00 |
| Knerr et al. [10] | 90.30 | 9.70 | 0.00 |
| Lemarie [15] | 97.97 | 2.03 | 0.00 |
| Kim et al. [9] | 95.85 | 4.15 | 0.00 |
| The proposed method | 97.90 | 2.10 | 0.00 |

proposed method is higher than those for other combining methods. Although the network learned the training set almost perfectly in all three cases, the performances on the test sets are quite different. Furthermore, we can see that the performance did not improve by training a large network with considering all the features used by each network. This is a strong evidence that multiple neural network might produce better result than conventional single network approach. Actually, the proposed method has a small, but statistically significant ($p > 0.995$ [3]), advantage in recognition rates obtained by the conventional methods.

[3] This range has been obtained by the paired $t$-test. In this comparison, the degree of freedom is $(n-1)=9$.

To give a fairer view of the performance in this field of handwritten digit recognition, Table 2 shows the performances of the presented hybrid method along with the results reported by some previous methods in the literature. The error rate of the proposed method is 2.10%, which is a big improvement compared with those of the previous methods. Actually, this performance is remarkable and can stand comparison with the best results reported in the literature.

## 5. Concluding remarks

This paper has presented genetic algorithm based method of combining neural networks for producing an improved performance on real-world recognition problem, especially handwritten digit recognition. The experimental results for classifying a large set of handwritten digits show that it improves the generalization capability significantly. This indicates that even these straightforward, computationally tractable approach can significantly enhance pattern recognition. The primary contribution of this paper lies in showing the potential of the hybrid soft computing technique for pattern recognition problem. Of course, further work seems necessary to conclude that the proposed method gives significant improvements from theoretical point of view.

## Acknowledgements

This work was supported in part by a grant from Brain Science Research Center with brain research program of the Ministry of Science and Technology in Korea.

## References

[1] J.A. Benediktsson, P.H. Swain, Consensus theoretic classification methods, IEEE Trans. Syst. Man Cybernet. 22 (1992) 418–435.
[2] S.-B. Cho, Cooperation of modularized neural networks by fuzzy integral with OWA operators, Proc. 3rd Internat. Conf. Fuzzy Logic, Neural Nets and Soft Computing, 1994, pp. 95–96.
[3] S.-B. Cho, Neural network ensemble aggregated by fuzzy logic, Proc. IEEE/Nagoya Univ. WWW on Fuzzy Logic and Neural Networks/Genetic Algorithms, 1994, pp. 46–52.

[4] S.-B. Cho, J.H. Kim, Two design strategies of neural network for complex classification problems, Proc. 2nd Internat. Conf. Fuzzy Logic and Neural Net, 1992, pp. 759–762.
[5] S.-B. Cho, J.H. Kim, Combining multiple neural networks by fuzzy integral for robust classification, IEEE Trans. Syst. Man Cybernet. 25 (1995) 380–384.
[6] E. Cohen, J.J. Hull, S.N. Srihari, Understanding handwritten text in a structured environment: Determining ZIP codes from addresses, Int. J. Pattern Recog. Artificial Intell. 5(1&2) (1991) 221–264.
[7] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Mach. Intell. 12 (1990) 993–1001.
[8] T.K. Ho, A theory of multiple classifier systems and its application to visual word recognition, Ph.D. Thesis, University of Buffalo, 1992.
[9] Y.J. Kim, S.-W. Lee, Off-line recognition of unconstrained handwritten digits using multilayer backpropagation neural network combined with genetic algorithm, Proc. 6th Workshop on Image Processing and Understanding, 1994, pp. 186–193 (in Korean).
[10] S. Knerr, L. Personnaz, G. Dreyfus, Handwritten digit recognition by neural networks with single-layer training, IEEE Trans. Neural Networks 3(6) (1992) 962–968.
[11] A. Krzyzak, W. Dai, C.Y. Suen, Unconstrained handwritten character classification using modified backpropagation model, Proc. 1st Internat. Workshop on Frontiers in Handwriting Recognition, Montreal, Canada, 1990, pp. 155–166.
[12] L. Lam, C.Y. Suen, Structural classification and relaxation matching of totally unconstrained handwritten ZIP-code numbers, Pattern Recog. 21(1) (1988) 19–31.
[13] Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Handwritten digit recognition with a back-propagation network, Adv. Neural Inform. Process. Systems 2 (1990) 396–404.
[14] R. Legault, C.Y. Suen, Contour tracing and parametric approximations for digitized patterns, in: A. Krzyzak, T. Kasvand, C.Y. Suen (Eds.), Computer Vision and Shape Recognition, World Scientific Publishing, Singapore, 1989, pp. 225–240.
[15] B. Lemarie, Practical implementation of a radial basis function network for handwritten digit recognition, Proc. 2nd Internat. Conf. Document Analysis and Recognition, Tsukuba, Japan, 1993, pp. 412–415.
[16] T. Mai, C.Y. Suen, A generalized knowledge-based system for the recognition of unconstrained handwritten numerals, IEEE Trans. Syst. Man Cybernet. 20(4) (1990) 835–848.
[17] G.L. Martin, J.A. Pittman, Recognizing hand-printed letters and digits, Adv. Neural Inform. Process. Systems 2 (1990) 405–414.
[18] C. Nadal, C.Y. Suen, Recognition of totally unconstrained handwritten numeral by decomposition and vectorization, Technical Report, Concordia University, Montreal, Canada, 1988.
[19] M.P. Perrone, L.N. Cooper, When networks disagree: ensemble methods for hybrid neural networks, in: R.J. Mammone (Ed.), Neural Net. for Speech and Image Proc., Chapman & Hall, London, 1993.
[20] W.K. Pratt, Digital Image Processing, Wiley, New York, 1978.
[21] M.D. Richard, R.P. Lippmann, Neural network classifiers estimate Bayesian a posteriori probabilities, Neural Comput. 3 (1991) 461–483.
[22] S. Shlien, Multiple binary decision tree classifiers, Pattern Recog. 23 (1990) 757–763.
[23] M. Srinivas, L.M. Patnaik, Genetic algorithms: a survey, IEEE Comput. (1994) 17–26.
[24] C.Y. Suen, C. Nadal, T. Mai, R. Legault, L. Lam, Recognition of handwritten numerals based on the concept of multiple experts, Proc. 1st Internat. Workshop Frontiers in Handwriting Recognition, Montreal, Canada, 1990, pp. 131–144.
[25] L. Xu, A. Krzyzak, C.Y. Suen, Methods of combining multiple classifiers and their applications to handwriting recognition, IEEE Trans. Syst. Man Cybernet. 22 (1992) 688–704.