Integrated Computer-Aided Engineering 9 (2002) 363–372
IOS Press

Fusion of neural networks with fuzzy logic and genetic algorithm

Sung-Bae Cho
Department of Computer Science, Yonsei University, 134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, South Korea
Tel.: +82 2 2123 2720; Fax: +82 2 365 2579; E-mail: sbcho@csai.yonsei.ac.kr

Abstract. Combining multiple estimators has appeared as one of hot research topics in several areas including artificial neural networks. This paper presents three methods to work out the problem based on so-called softcomputing techniques, toward a unified framework of hybrid softcomputing techniques. The first method based on fuzzy logic nonlinearly combines objective evidences, in the form of network outputs, with subjective evaluation of the reliability of the individual neural networks. The second method based on genetic algorithm gives us an effective vehicle to determine the optimal weight parameters that are multiplied by the network outputs. Finally, we have proposed a hybrid synergistic method of fuzzy logic and genetic algorithm to optimally combine neural networks. The experimental results with the recognition problem of totally unconstrained handwritten digits show that the performance could be improved significantly with the proposed softcomputing techniques.

1. Introduction

When trying to solve a real-world problem by neural networks, we are faced with a large variety of learning algorithms and a vast selection of possible network architectures. After all the training, we choose the best network with a winner-take-all cross-validatory model selection. Recent theoretical and experimental studies, however, indicate that we can improve the performance by considering methods for combining neural networks [2,3,7,9,20,22,32]. One of the key issues of this approach is how to combine the results of the various networks to give the best estimate of the optimal result. There are a number of possible schemes for automatically optimizing the choice of individual networks and/or combining architectures.

It is possible to identify two main approaches to combining neural networks: modular and ensemble-based approaches [23]. In modular approach, a problem is decomposed into a number of subtasks that are treated by specialist modules. Hampshire and Waibel [6] have described a system of this kind that can be used when the decomposition into subtasks is known prior to training, and Jacobs et al. [9] have also proposed a supervised learning procedure for systems composed of many separate networks, each of which learns to handle a subset of the complete set of training instances. The subnetworks are local in the sense that the weights in one expert are decoupled from the weights in other subnetworks. However, there is still some indirect coupling because it may cause the gating network to alter the responsibilities that get assigned to the subnetworks if some other network changes its weights.

On the contrary, ensemble-based approach facilitates a set of networks trained on what is essentially the same task, and then the outputs of the networks are combined. This aims to obtain a more reliable ensemble output than would be obtained by selecting the best network. While a 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 applied in statistics, management sciences, and pattern recognition [1, 33]. A general result from the previous works is that averaging separate networks improves generalization performance for the mean squared error [20]. If we have networks of different accuracy, however, it is ob-

ISSN 1069-2509/02/$8.00

viously not good to take their simple average or simple voting.

To give a solution to the problem, this paper presents three methods based on so-called softcomputing techniques such as fuzzy logic, genetic algorithm, and the hybrid of them. These methods tend to exploit the difference of each network to combine the networks. The first method is to combine the network outputs with the importance of each network, where the importance can be subjectively assigned from the spirit of fuzzy logic. The second method utilizes the weight parameters, which are determined by genetic algorithm, to obtain the combined output. The third method is to hybridize the two methods for achieving the optimal solution to combine neural networks.

There have been a lot of literature that integrate NN/Fuzzy, Fuzzy/GA, GA/NN, and NN/Fuzzy/GA [5, 30]. There are even some attempts to use fuzzy logic and/or genetic algorithms to search for ensemble members. However, there is no systematic work to propose the ensemble framework for multiple neural networks just like ours. Our novelty lies in proposing the unified framework to combine three softcomputing techniques to optimize the way of how to combine the networks, even though the idea itself is not that complicated.

The rest of this paper is organized as follows. Section 2 formulates the problem and considers possible methods for combining neural networks. In Sections 3 to 5, we present the combining methods based on fuzzy logic, genetic algorithm, and the hybrid of them, respectively. In Section 6, we demonstrate the superior performance of the presented methods and compare with conventional methods by thorough experiments in a real-world pattern recognition problem of totally unconstrained handwritten digits.

2. Combining neural networks

A neural network can be considered as a mapping device between input and output sets. 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-layered feedforward neural network trained with the generalized delta rule, as a classifier.

Consider a two-layered 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 unknown input $X = (x_1, x_2, \ldots, x_T)$ and the class set $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$, each output neuron produces y_i of belonging to this class by

$$\begin{aligned} P(\omega_i|X) &\approx y_i \\ &= f\left\{\sum_{k=1}^{H} w_{ik}^{om} f\left(\sum_{j=1}^{T} w_{kj}^{mi} x_j\right)\right\}, \end{aligned} \tag{1}$$

where w_{kj}^{mi} is a weight between the jth input neuron and the kth hidden neuron, w_{ik}^{om} is a weight from the kth hidden neuron to the ith class output, and f is a sigmoid function such as $f(x) = 1/(1 + e^{-x})$. 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 combining neural networks 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 [7,24]. Then it naturally raises the question of obtaining a consensus on the results of each individual network or expert.

Two general approaches to combining multiple networks can be identified; One is based on a fusion technique and the other on a 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:

$$P(\omega_i|X), \quad 1 \leqslant i \leqslant c.$$

They represent the probabilities that X comes from each of the c classes under the condition X. In the combined network scheme, each network k estimates by itself a set approximations of those true values as follows:

$$P_k(\omega_i|X), \quad 1 \leqslant i \leqslant c, \quad 1 \leqslant k \leqslant n.$$

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:

$$P(\omega_i|X) = \frac{1}{n}\sum_{k=1}^{n} P_k(\omega_i|X), \quad 1 \leqslant i \leqslant c. \tag{2}$$

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

$$P(\omega_i|X) = \sum_{k=1}^{n} r_k^i P_k(\omega_i|X), \quad 1 \leqslant i \leqslant c, \quad \text{where } \sum_{k=1}^{n} r_k^i = 1. \tag{3}$$

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, and so on. In particular, we shall introduce the Borda count which has been reported as the best method among voting techniques [8].

For any particular class i, the Borda count is the sum of the number of classes ranked below i by each network. Let $B_k(i)$ be the number of classes ranked below the class i by the kth network. Then, the Borda count for class i becomes:

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

Like the average case, we can extend it to the weighted Borda count by considering the reliability of the neural networks in the combination:

$$P(\omega_i|X) = \sum_{k=1}^{n} r_k^i B_k(i), \quad 1 \leqslant i \leqslant c, \quad \text{where} \sum_{k=1}^{n} r_k^i = 1. \tag{5}$$

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

3. Fuzzy logic based method

In this section, we describe the method that utilizes the fuzzy integral for combining neural networks. This method might produce better performance with the subjectively assigned importances of individual networks.

3.1. Overview of fuzzy integral

The fuzzy integral is a nonlinear functional that is defined with respect to a fuzzy measure, especially g_λ-fuzzy measure introduced by Sugeno [27]. The ability of the fuzzy integral to combine the results of multiple sources of information has been established in several previous works [15,28,34].

Definition 1. A set function $g : 2^X \to [0, 1]$ is called a fuzzy measure if

1) $g(\emptyset) = 0$, $g(X) = 1$,
2) $g(A) \leqslant g(B)$ if $A \subset B$,
3) If $\{A_i\}_{i=1}^{\infty}$ is an increasing sequence of measurable sets, then

$$\lim_{i\to\infty} g(A_i) = g(\lim_{i\to\infty} A_i).$$

Note that g is not necessarily additive. This property of monotonicity is substituted for the additivity property of the conventional measure.

From the definition of a fuzzy measure g, Sugeno introduced the so-called g_λ-fuzzy measures satisfying the following additional property: For all $A, B \subset X$ and $A \cap B = \emptyset$,

$$g(A \cup B) = g(A) + g(B) + \lambda g(A)g(B), \quad \text{for some } \lambda > -1.$$

It affords that the measure of the union of two disjoint subsets can be directly computed from the component measures.

Using the notion of fuzzy measures, Sugeno developed the concept of the fuzzy integral, which is a nonlinear functional that is defined with respect to a fuzzy measure, especially g_λ-fuzzy measure [27,15,28].

Definition 2. Let X be a finite set, and $h : X \to [0, 1]$ be a fuzzy subset of X. The fuzzy integral over X of the function h with respect to a fuzzy measure g is defined by

$$
\begin{aligned}
& h\,(x)\ \circ g(\cdot) \\
&= \max_{E\subseteq X}\left[\min\left(\min_{x\in E} h(x), g(E)\right)\right] \qquad (6)\\
&= \sup_{\alpha\in[0,1]}\left[\min\left(\alpha, g(h_\alpha)\right)\right]
\end{aligned}
$$

where h_α is the α level set of h,

$$h_\alpha = \{x \mid h(x) \geqslant \alpha\}. \qquad (7)$$

To get some intuition for the fuzzy integral we consider the following interpretation. $h(y)$ measures the degree to which the concept h is satisfied by y. The term $\min_{y\in E} h(y)$ measures the degree to which the concept h is satisfied by all the elements in E. Moreover, the value $g(E)$ is a measure of the degree to which the subset of objects E satisfies the concept measured by g. Then, the value obtained from comparing these two quantities in terms of the min operator indicates the degree to which E satisfies both the criteria of the measure g and $\min_{y\in E} h(y)$. Finally, the max operation takes the biggest of these terms. One can interpret the fuzzy integral as finding the maximal grade of agreement between the objective evidence and expectation.

3.2. Fuzzy integral for combining networks

The calculation of the fuzzy integral when Y is a finite set is easily given. Let $Y = \{y_1, y_2, \ldots, y_n\}$ be a finite set and let $h : Y \to [0,1]$ be a function. Suppose $h(y_1) \geqslant h(y_2) \geqslant \ldots \geqslant h(y_n)$, (if not, Y is rearranged so that this relation holds). Then a fuzzy integral, e, with respect to a fuzzy measure g over Y can be computed by

$$e = \max_{i=1}^{n}\left[\min\left(h(y_i),\ g(A_i)\right)\right] \qquad (8)$$

where $A_i = \{y_1, y_2, \ldots, y_i\}$.

Note that when g is a g_λ-fuzzy measure, the values of $g(A_i)$ can be determined recursively as

$$
\begin{aligned}
& g(A_1) = g(\{y_1\}) = g^1 \\
& g(A_i) = g^i + g(A_{i-1}) + \lambda g^i g(A_{i-1}), \\
& \qquad\qquad \text{for } 1 < i \leqslant n.
\end{aligned}
$$

λ is given by solving the equation

$$\lambda + 1 = \prod_{i=1}^{n}(1 + \lambda g^i) \qquad (9)$$

where $\lambda \in (-1, +\infty)$, and $\lambda \neq 0$. This can be easily calculated by solving an $(n-1)$st degree polynomial and finding the unique root greater than -1. Thus the calculation of the fuzzy integral with respect to a g_λ-fuzzy measure would only require the knowledge of the density function, where ith density, g^i, is interpreted as the degree of importance of the source y_i towards the final evaluation.

Let $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$ be a set of classes of interest. Note that each ω_i may, in fact, be a set of classes by itself. Let $Y = \{y_1, y_2, \ldots, y_n\}$ be a set of neural networks, and A be the object under consideration for recognition. Let $h_k : Y \to [0,1]$ be the partial evaluation of the object A for class ω_k, that is, $h_k(y_i)$ is an indication of how certain we are in the classification of object A to be in class ω_k using the network y_i, where a 1 indicates absolute certainty that the object A is really in class ω_k and 0 implies absolute certainty that the object A is not in ω_k.

Corresponding to each y_i the degree of importance, g^i, of how important y_i is in the recognition of the class ω_k must be given. These densities can be subjectively assigned by an expert, or can be induced from data set. The g^i's define the fuzzy density mapping. Hence λ is calculated using Eq. (9) and thereby the g_λ-fuzzy measure, g, is constructed. Now, using Eqs (8) to (9), the fuzzy integral can be calculated. Finally, the class ω_k with the largest integral value is chosen as the output class.

4. 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.

4.1. Overview of genetic algorithm

Evolution is a remarkable problem-solving machine [25]. First proposed by John Holland in 1975, GA as one of computational implementations is an attractive class of computational models that mimic natural evolution to solve problems in a wide variety of domains. A GA emulates biological evolutionary theories to solve optimization problems.

In common with all other search algorithms, a GA performs a search of a multidimensional space containing a hypersurface known as the fitness surface. A particular parameter set defines a point on a hyperplane on to which the surface is projected, with the height of the surface above the hyperplane reflecting the rel-

ative merit of the problem solution represented by the parameter set.

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. Often it is not possible to exhaustively test all aspects of a solution, and noise may be present on the objective function, so the assigned fitness is an estimate of the true fitness of a chromosome. 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 heredity to new generations.

4.2. Genetic algorithm for combining networks

Opitz and Shavlik [19] present an algorithm that uses genetic algorithms to search actively for ensemble members which generalize well, but which disagree as much as possible. The GA operators are used to create new individuals from an initial set. Unlike this method, we just want to optimize the weights of the ensemble networks with GA. In our problem, a string must encode $n \times c$ real-valued parameters, r_k^i, in Eq. (3), 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 to create the new population of strings.

   (a) one-point crossover with probability 0.6.
   (b) standard mutation with probability 0.01.

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. Figure 1 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.

5. Genetic fuzzy hybrid method

Fuzzy logic and genetic algorithm are probably useful techniques that have been proposed for achieving some aspect of intelligent system [29,5]. Their differences, however, have prompted a number of researchers to try combining them to produce more powerful systems. In this section we present a hybrid method of fuzzy logic and genetic algorithm to give an optimal solution to combine neural networks.

As described in the previous sections, each method has its own pros and cons. To produce more powerful system, several integration and synthesis techniques of them have been proposed. It is natural to think of a framework for high-performance system based on them. Neural networks can be used as a baseline system, because they are well recognized as a powerful input–output mapper. However, human operators cannot easily incorporated some knowledge about the problem into the neural networks. In this case, fuzzy logic might be useful.

Fuzzy logic gives a possibility to utilize top-down knowledge from designer. Human operators can

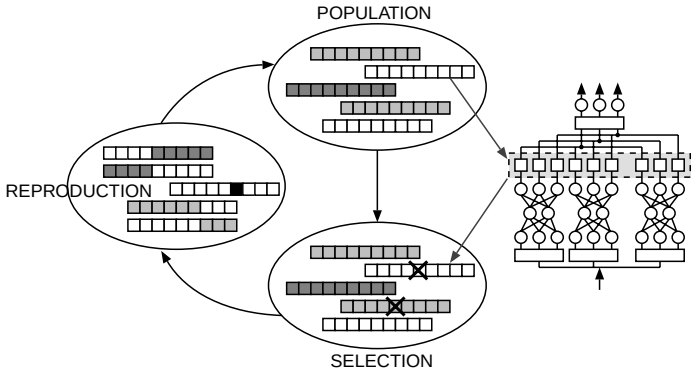

Fig. 1. Overall procedure of the genetic algorithm based method.

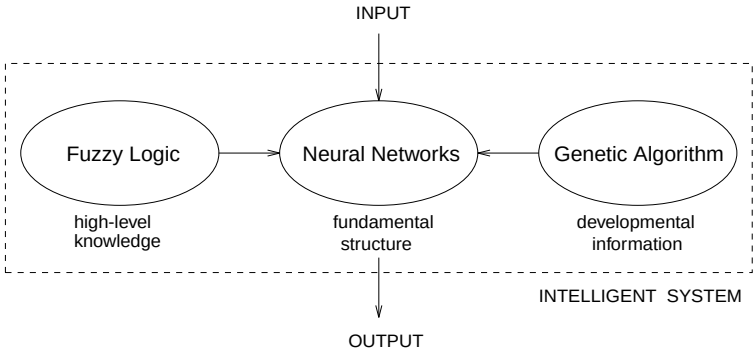

Fig. 2. Schematic diagram of the hybrid framework based on neural networks, fuzzy logic and genetic algorithm.

enhance the neural networks by incorporating their knowledge with fuzzy membership functions, which are modified through learning process as fine tuning. After the learning, the human operators may be able to understand the acquired rules. On the other hand, genetic algorithm is a powerful tool for structure optimization of fuzzy logic and neural networks which provide evaluation functions for genetic algorithm. Figure 2 shows a schematic diagram of the general framework based on the hybridization of them.

To give an idea of how such hybrid technique yields better system, we have developed the hybrid method that utilizes the fuzzy integral to combine the outputs of separate networks with importance of each network, which is assigned by genetic algorithm. In the following, $\hat{g}_\lambda(A)$ and $\hat{g}_i$ denote the human-provided values, and $g_\lambda(A)$ and g_i denote the identified values.

In this method, chromosomes encode the fuzzy density values g_i^j by a vector $C_j = (g_1^j, g_2^j, \ldots, g_k^j; \lambda_j)$. The fitness function $f(C_j)$ for chromosome C_j is the sum of the differences between human-provided fuzzy measure value $\hat{g}_\lambda(A)$ and fuzzy measure value obtained by g_i^j and λ_j.

$$f(C_j) = \sum_{A \in B(X)} \left| \hat{g}_\lambda(A) - \frac{1}{\lambda_j} \left[\prod_{x_i \in A} (1 + \lambda_j g_i^j) - 1 \right] \right|$$

With these the genetic operators yield an optimal set of parameters to combine neural networks. When adopting GA to solve a problem at hand, we have to consider about the computational cost and convergence problem: It may not converge if the problem is structured poorly. However, in the hybrid system, they are not so serious because the solution space is not that large, and furthermore, the GA replaces a moderate amount of trial-and-error overhead we have to put in the course of deciding the optimal parameters.

6. Experiments and analysis

6.1. Environments

In the experiments, we have used the handwritten digit database of Concordia University of Canada,

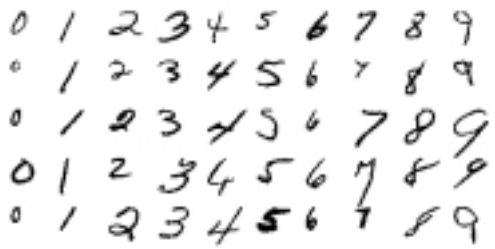

Fig. 3. Sample data.

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 64×224 grid of 0.153 mm square elements, giving a resolution of approximately 166 PPI [26]. Among the data, 4000 digits were used for training and 2000 digits for testing. Figure 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.

Handwritten digits are essentially line drawings: One-dimensional structures in a two-dimensional space. Thus, local detection of line segments seems to be adequate for extracting features. For each location in the image, information about the presence of a line segment of a given direction is stored in a feature map [11]. Especially, in this paper Kirsch masks have been used for extracting directional features as proposed by [10].

Kirsch defined a nonlinear edge enhancement algorithm as follows [21]:

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

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 $(k = 0,1,\ldots,7)$ is eight neighbors of pixel (i,j) defined 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 × 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 [10]. The

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).

final step in extracting the features compresses each 16 × 16 directional vector into 4 × 4 vector with averaging operator, which produces a value for 2 × 2 pixels with dividing by 4 the value obtained by summing the four values. Moreover, 4 × 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, 8 directional chain codes are obtained. 128 histograms for chain codes in 4 × 4 subregions are finally generated for a digit pattern. As a result, available features include four 4 × 4 local features, one 4 × 4 global features, and structural features extracted from the contours of the digits. For more details on the features used, see the recent publication [4].

6.2. Experimental results

To evaluate the performance of the combining methods, we have implemented three different networks, each of which is a two-layered 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.

For the use of weighting among the networks, we assigned the degree of importance of each network, g^i,

Table 1
Results of combining networks by the fuzzy integral on three different networks

Data index	Actual class	Partial decision			Fuzzy integral
		NN_1	NN_2	NN_3	decision
1	6	6 (0.9968)	6 (0.9985)	5 (0.3301)	6 (0.6877)
2	8	8 (0.9999)	0 (0.0022)	8 (0.9996)	8 (0.6668)
3	2	2 (0.9922)	2 (0.9998)	2 (0.9920)	2 (0.9946)
4	7	8 (0.0162)	8 (0.0087)	7 (0.9615)	7 (0.3205)
5	9	9 (0.9965)	9 (0.9958)	8 (0.0740)	9 (0.6691)
6	3	3 (0.9987)	3 (0.9912)	3 (0.9999)	3 (0.9966)
7	7	8 (0.0365)	0 (0.0460)	7 (0.4831)	7 (0.1610)
8	5	5 (0.9311)	3 (0.1304)	3 (0.6245)	5 (0.3265)
9	2	8 (0.3470)	2 (0.9983)	8 (0.2092)	2 (0.3327)
10	9	9 (0.9989)	1 (0.9944)	9 (0.9902)	9 (0.6705)
11	4	4 (0.9998)	4 (0.9999)	4 (0.9995)	4 (0.9997)
12	8	8 (0.9998)	0 (0.9882)	8 (0.8353)	8 (0.6204)

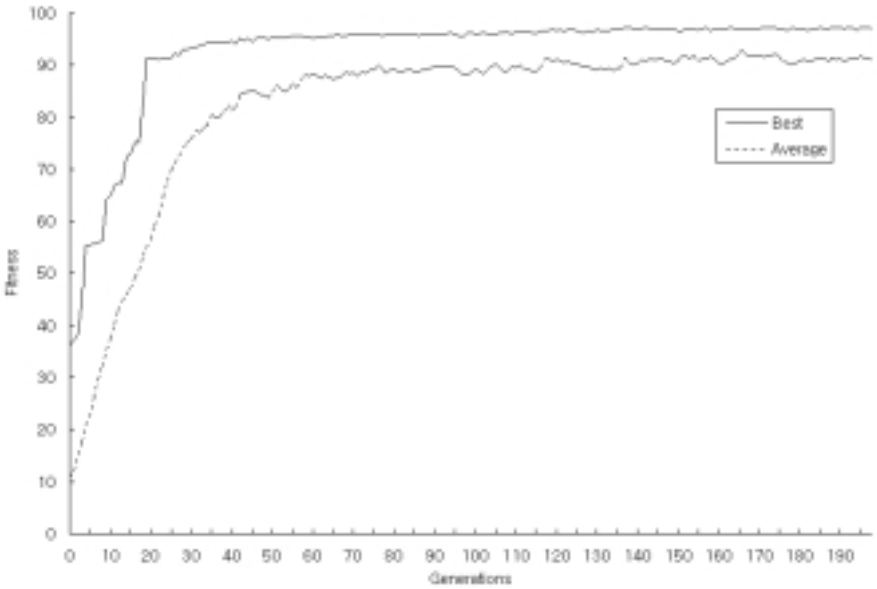

Fig. 5. Best and average fitness changes as generation goes.

based on how good these networks performed on the train data. We computed these values as follows:

$$g^i = \frac{p_i}{\sum_j p_j} \cdot d\text{sum}, \tag{13}$$

where p_i is the performance of network NN_i for the train data and $d$sum is the desired sum.

Table 1 reports some examples of the results of combining networks by the fuzzy integral just to show the general tendency of the results. In this table the value in the parentheses represent the confidence of the evaluation result.

As can be seen, cases 1 and 2 are misclassified by NN_3 and NN_2, respectively. However, in the final evaluations they are correctly classified. In cases 4 and 9, one network with strong evidence overwhelmed the other networks, producing correct classification. Furthermore, in case 7, the fuzzy integral made a correct decision despite that the partial decisions from the individual neural networks are totally inconsistent. The effect of misclassification by the other networks has given rise to small fuzzy integral values for the correct classification in this case.

As the second experiment, GA is used to obtain the optimal parameters to combine the three neural networks with different features. 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 0.01 mutation rate. A fitness value is assigned to a string

by testing the recognition rate with the trained digits. Figure 5 shows the best and average fitness changes of the GA based method. The figure shows that the fitness increases as the iteration goes and the overall fitness settles down soon after the initial radical improvements.

Table 2 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 methods. NN_{all} here means the network trained with all the available features, and Hybrid means the method that combines the three neural networks by fuzzy integral with different importances that are assigned by genetic algorithm.

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

$$\text{Reliability} = \frac{\text{Correct}}{\text{Correct} + \text{Error}}. \tag{14}$$

As can be seen, every method of combining neural networks produces better results than individual networks, and the overall classification rates for the softcomputing techniques are higher than those for other conventional 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 softcomputing methods have a statistically significant ($p > 0.995$) advantage in recognition rates obtained by the conventional methods. 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 3 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 1.95%, which is a big improvement compared with those of the previous methods. Some of the cited studies do not with the combinations of classifiers, and there might be other potential reasons for the improved performance reported in this table. However, this performance is still remarkable and can stand comparison with the best results reported in the literature.

Table 2
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
Fuzzy	97.78	1.90	0.32	98.10
Genetic	97.90	2.10	0.00	97.90
Hybrid	98.05	1.95	0.00	98.05

Table 3
Comparisons of the presented method with the related (%)

Methods	Correct	Error	Reject
Lam et al. [13]	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. [12]	86.40	1.00	12.60
Krzyzak et al. [12]	94.85	5.15	0.00
Mai et al. [17]	92.95	2.15	4.90
Suen et al. [26]	93.05	0.00	6.95
Kim et al. [10]	95.40	4.60	0.00
Kim et al. [10]	95.85	4.15	0.00
The hybrid method	98.05	1.95	0.00

7. Concluding remarks

This paper has presented three combining methods of neural networks for producing an improved performance on real-world classification problem, in particular 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 possibility of the softcomputing techniques which are not so strict as the usual probabilistic or mathematical approaches. However, we need to work further to extend some limitations of the current approach. Most urgent matters are to prove that the proposed methods give significant improvements in theoretical point of view, and to devise the best method to create candidate ensemble members. For the former problem we are attempting to utilize the bias-variance dilemma and Bayesian model selection [16,31], and for the latter to exploit the speciated evolutionary algorithms.

Furthermore, future efforts will concentrate on refining the feature extraction to capture more information, and testing the efficacy of the softcomputing techniques

on larger data sets. The complementary nature of fuzzy logic and genetic algorithm leads us to believe that a further refined genetic fuzzy neural system will significantly improve the state-of-the-art pattern recognizers.

Acknowledgements

This paper was supported by Brain Science and Engineering Research Program sponsored by Korean Ministry of Science and Technology. The author would like to thank Prof. Xin Yao, Prof. Hojjat Adeli, and the three anonymous reviewers for constructive comments.

References

[1] J.A. Benediktsson and P.H. Swain, Consensus theoretic classification methods, *IEEE Trans. Syst. Man. Cyber.* **22** (1992), 418–435.

[2] S.-B. Cho and J.H. Kim, Two design strategies of neural network for complex classification problems, in: *Proc. 2nd Int. Conf. Fuzzy Logic & Neural Net*, 1992, pp. 759–762.

[3] S.-B. Cho and J.H. Kim, Combining multiple neural networks by fuzzy integral for robust classification, *IEEE Trans. Syst. Man. Cyber.* **25** (1995), 380–384.

[4] S.-B. Cho, Neural-network classifiers for recognizing totally unconstrained handwritten numerals, *IEEE Trans. Neural Networks* **8** (1997), 43–53.

[5] T. Fukuda, Fuzzy-neuro-GA based intelligent robotics, in: *Computational Intelligence: Imitating Life*, J.M. Zurada, R.J. Marks II, C.J. Robinson, eds, IEEE Press, 1994, pp. 252–263.

[6] J.B. Hampshire II and A. Waibel, Connectionist architectures for multi-speaker phoneme recognition, *Adv. Neural Info. Proc. Syst* **2** (1990), 203–210.

[7] L.K. Hansen and P. Salamon, Neural network ensembles, *IEEE Trans. Patt. Anal. Mach. Inte.* **12** (1990), 993–1001.

[8] T.K. Ho, A theory of multiple classifier systems and its application to visual word recognition, Ph.D. Dissertation, University of Buffalo, 1992.

[9] R.A. Jacobs, M.I. Jordan, S.J. Nowlan and G.E Hinton, Adaptive mixtures of local experts, *Neural Comp.* **3** (1991), 79–87.

[10] Y.J. Kim and S.W. Lee, Off-line recognition of unconstrained handwritten digits using multilayer backpropagation neural network combined with genetic algorithm, in: *Proc. 6th Workshop on Image Processing and Understanding*, 1994, pp. 186–193, (in Korean).

[11] S. Knerr, L. Personnaz and G. Dreyfus, Handwritten digit recognition by neural networks with single-layer training, *IEEE Trans. on Neural Networks* **3** (1992), 962–968.

[12] A. Krzyzak, W. Dai and C.Y. Suen, Unconstrained handwritten character classification using modified backpropagation model, in: *Proc. 1st Int. Workshop on Frontiers in Handwriting Recognition*, Montreal, Canada, 1990, pp. 155–166.

[13] L. Lam and C.Y. Suen, Structural classification and relaxation matching of totally unconstrained handwritten ZIP-code numbers, *Pattern Recognition*, **21** (1988), pp. 19–31.

[14] R. Legault and C.Y. Suen, Contour tracing and parametric approximations for digitized patterns, in: *Computer Vision and Shape Recognition*, A. Krzyzak, T. Kasvand and C.Y. Suen, eds, World Scientific Publishing, Singapore, 1989, pp. 225–240.

[15] K. Leszeynski, P. Penczek and W. Grochulskki, Sugeno's fuzzy measures and fuzzy clustering, *Fuzzy Sets Syst.* **15** (1985), 147–158.

[16] D. Madigan et al., Bayesian model averaging, in: *Proc. AAAI Workshop on Integrating Multiple Learned Models*, 1996, pp. 77–83.

[17] T. Mai and C.Y. Suen, A generalized knowledge-based system for the recognition of unconstrained handwritten numerals, *IEEE Trans. Syst. Man. Cyber.* **20** (1990), pp. 835–848.

[18] C. Nadal and C.Y. Suen, Recognition of totally unconstrained handwritten numeral by decomposition and vectorization, Technical Report, Concordia University, Montreal, Canada, 1988.

[19] D.W. Opitz and J.W. Shavlik, Actively searching for an effective neural network ensemble, *Connection Science* **8** (1996), pp. 337–353.

[20] M.P. Perrone and L.N. Cooper, When networks disagree: Ensemble methods for hybrid neural networks, in: *Neural Net. for Speech and Image Proc*, R.J. Mammone, ed., Chapman-Hill, London, 1993.

[21] W.K. Pratt, *Digital Image Processing*, Wiley, New York, 1978.

[22] C. Scofield, L. Kenton and J. Chang, Multiple neural net architectures for character recognition, in: *Proc. Compcon.*, IEEE Computer Society Press, San Francisco, CA, 1991, pp. 487–491.

[23] A.J.C. Sharkey, On combining artificial neural nets, *Connection Science* **8** (1996), 299–313.

[24] S. Shlien, Multiple binary decision tree classifiers, *Patt. Recog.* **23** (1990), 757–763.

[25] M. Srinivas and L.M. Patnaik, Genetic algorithms: A survey, *IEEE Computer* (June 1994), 17–26.

[26] C.Y. Suen, C. Nadal, T. Mai, R. Legault and L. Lam, Recognition of handwritten numerals based on the concept of multiple experts, in: *Proc. 1st Int. Workshop Frontiers in Handwriting Recognition*, Montreal, Canada, 1990, pp. 131–144.

[27] M. Sugeno, Fuzzy measures and fuzzy integrals: A survey, *Fuzzy Automata Dec. Proc.*, Amsterdam, North Holland, 1977, pp. 89–102.

[28] H. Tahani and J.M. Keller, Information fusion in computer vision using the fuzzy integral, *IEEE Trans. Syst. Man. Cyber.* **20** (1990), 733–741.

[29] H. Takagi, Fusion technology of fuzzy theory and neural networks, in: *Proc. Int. Conf. Fuzzy Logic & Neural Networks*, 1990, pp. 13–26.

[30] H. Takagi, Fusion technology of neural networks and fuzzy systems: A chronicled progression from the laboratory to our daily lives, *Int. Journal of Applied Mathematics and Computer Science* **10** (2000), 647–673.

[31] C. Volinsky and A.E. Raftery, Bayesian information criterion for censored survival models, *Biometrics* **56** (2000), 256–262.

[32] D. Wolpert, Stacked generalization, *Neural Net.* **5** (1992), 241–259.

[33] L. Xu, A. Krzyzak and C.Y. Suen, Methods of combining multiple classifiers and their applications to handwriting recognition, *IEEE Trans. Syst. Man. Cyber.* **22** (1992), 688–704.

[34] R.R. Yager, Element selection from a fuzzy subset using the fuzzy integral, *IEEE Trans. Syst. Man. Cyber.* **23** (1993), 467–477.