Feedforward Neural Network Architectures for Complex Classification Problems*

Sung-Bae Cho† and Jin H. Kim

Center for Artificial Intelligence Research and Computer Science Department

Korea Advanced Institute of Science and Technology

373-1, Koosung-dong, Yoosung-ku, Taejeon 305-701, Republic of Korea

February 1, 1993

Abstract

This paper presents two neural network design strategies for incorporating *a priori* knowledge about a given problem into the feedforward neural networks. These strategies aim at obtaining tractability and reliability for solving complex classification problems by neural networks. The first type strategy based on multistage scheme decomposes the problem into manageable ones for reducing the complexity of the problem, and the second type strategy on multiple network scheme combines incomplete decisions from several copies of networks for reliable decision-making. A preliminary experiment of recognizing on-line handwriting characters confirms the superiority relative to a single large neural network classifier.

Key words: neural network architecture design, multistage neural network, multiple neural networks, synthesis method, voting method, expert judgement, handwriting character recognition

---

The idea of this paper was presented at the 2nd International Conference on Fuzzy Logic & Neural Networks, Iizuka-92.

To whom all correspondence should be addressed. E-mail: sbcho@gorai.kaist.ac.kr.

1 Introduction

A neural network is a parallel computational paradigm which solves a given problem with massive interconnections of simple processors, and several models have been proposed for a variety of difficult problems, especially classification problems [1]. Traditional classifiers test competing hypotheses sequentially, whereas neural network classifiers test the competing hypotheses in parallel, thus providing high computational rates. Among many different neural network classifiers, the feedforward neural network classifier has been successfully applied to many problems such as discriminating between underwater sonar returns and forming text-to-phoneme rules.

In the conventional approach for applying a neural network to real-world problems, one starts with a training database, chooses a network by making an educated guess, and then uses a training algorithm to load as many of the training examples as possible onto the network. In principle, one can always find a neural network that can solve a given problem, provided that there is no restriction on the size of the network and that an infinite amount of data is available. In practice, however, one has to deal with a limited amount of resources, and has to rely on the generalization abilities of the network.

Note that, once one fixes the structure of the network, i.e. the number of hidden layers and the number of nodes in each hidden layer are chosen, then the network adjusts its weights via the learning rule until the optimal weights are obtained. The corresponding weights along with the structure of the network create the decision boundaries in the feature space. In many practical pattern recognition problems, however, this usual neural network classifier tends not to converge to its solution state. Even if the network converges, the time required for convergence may be too prohibitive for practical purposes. To overcome this difficulty, a variety of modular neural networks has been proposed [2].

This paper presents two design methods of the feedforward neural networks based on divide and conquer approach. The strategies for constructing the effective structure of feedforward networks are described as a mechanism for incorporating *a priori* knowledge about a given problem. The first type strategy based on multistage scheme decomposes the

problem into manageable ones, and the second type strategy on multiple network scheme combines incomplete decisions from several copies of networks for reliable decision-making. The experimental results with the recognition problem of on-line handwriting characters confirm the usefulness of the presented methods.

The rest of this paper is organized as follows. Section 2 reviews the backpropagation neural network as a classifier. In Section 3 and 4, we present the proposed strategies, and the experimental results with the recognition of on-line handwriting characters are explained in Section 5. Section 6 discusses the summary and the future research.

2 Neural Network Classifier

A neural network can be considered as a mapping device between a set of input and a set of output. Mathematically speaking, a neural network represents a function F that maps I into O; $F : I \rightarrow O$, or $y = F(x)$ where $y \in O$ and $x \in I$. By the way, 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 backpropagation algorithm, as a classifier.

Figure 1 shows a two-layer backpropagation classifier with d neurons in the input layer, $2d+1$ neurons in the hidden layer, and c neurons in the output layer. Here, d is the number of features, c is the number of classes, and $2d+1$ is an appropriately selected number (this number is induced by Kolmogorov's theorem). The network is fully connected between adjacent layers. The operation of the backpropagation classifier for this network is as follows.

Let $X = \{x_1, x_2, \ldots, x_d\}$ be the feature set. Then each node x'_k in the hidden layer calculates its output through

$$x'_k = f\left(\sum_{j=1}^{d} w_{kj}^{mi} x_j\right), \qquad 1 \leq k \leq 2d+1, \tag{1}$$

where w_{kj}^{mi} is a weight between the jth input node and the kth hidden node, and f is

Figure 1: A two-layer neural network architecture. In this network, d is the number of features, c is the number of classes, and $2d + 1$ is an appropriately selected number.

a sigmoid function such as $f(x) = 1/(1 + e^{-x})$. After this process, each node y_i in the output layer calculates the output in the similar way;

$$y_i = f\left(\sum_{k=1}^{2d+1} w_{ik}^{om} {x'}_k\right), \qquad 1 \leq i \leq c, \tag{2}$$

where w_{ik}^{om} is a weight from the kth hidden node to the ith class output.

Let $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$ be the class set. Then the decision rule is to decide ω_i if

$$i^* = \arg\max_i \ y_i, \qquad 1 \leq i \leq c, \tag{3}$$

in other words, the neural network selects the largest output as the correct class.

This can be thought of as a nonlinear decision-making process; Given an unknown

input, each output node estimates the possibility y_i of belonging to this class by

$$y_i = f\left\{\sum_{k=1}^{2d+1} w_{ik}^{om} f\left(\sum_{j=1}^{d} w_{kj}^{mi} x_j\right)\right\}, \tag{4}$$

and then the node having maximum value is selected as the corresponding class. The key point is how to determine the weight values, w_{ik}^{om} and w_{kj}^{mi}.

The process of weight tuning for optimal classification is called training for which the backpropagation algorithm adjusts the weights between the layers according to the following equation:

$$\Delta w_{ji} = \eta\ \delta_j\ y_i \tag{5}$$

where η is a learning rate, δ_j is a difference between desired output and actual output of node j, and y_i is the output of the node i. The δ_j can be rewritten as follows according to the layers when the sigmoid function is used.

- when node j is at the output layer:

$$\delta_j = (d_j - y_j)\ y_j\ (1 - y_j) \tag{6}$$

- when node j is at the hidden layer:

$$\delta_j = y_j\ (1 - y_j)\ \sum_k \delta_k w_{kj} \tag{7}$$

where d_j is a desired output of node j, w_{kj} is the weight between node k and node j. The term $y_j(1 - y_j)$ is a result to differentiate the sigmoid function.

3 Strategy I: Multistage Scheme

According to the previous formalization of the neural network as a classifier, a neural network having d binary input nodes and c output nodes can be considered as a classifier to assign a given sample with d features to one of predefined c categories. The two-

Figure 2: The multistage neural network scheme (two-stage case). The coarse network performs the mapping NN_c, which is a switch for selecting one of the k classifiers in the fine stage.

stage neural network decomposes the classification problem into several manageable ones. The design of a two-stage scheme for efficient classification requires some methodological considerations [3, 4, 5, 6]. The first step may be to find the partition $\{\xi_1, \xi_2, \ldots, \xi_k\}$ by using a clustering technique such as *k-means algorithm*, or using some *a priori* knowledge about problem structure. As the result, we are given the coarse partition of the total class Ω, $\{\xi_1, \xi_2, \ldots, \xi_k\}$, and the fine partition of Ω, $\{\omega_1, \omega_2, \ldots, \omega_c\}$.

Our two-stage classifier is shown in figure 2. The coarse network in the figure performs the mapping NN_c, which is a switch for selecting one of the k classifiers in the fine stage. The networks in the second stage are realizations of the mappings NN_f^i. This kind of neural network approaches, where the desired mappings are accomplished with several smaller neural networks, typically require less training time than approaches that utilize a single large network to carry out the mappings.

Now, we will demonstrate how much amount of complexity is reduced in the two-stage neural network classification scheme. In order to appreciate the difficulty of training the neural network with large number of connections, let us consider a single layer network

using linear discriminant functions first.

Suppose there are n sample points in d dimensions, and each point is labelled either ω_1 or ω_2. Of the 2^n possible dichotomies of n points in d dimensions, a certain fraction $f(n,d)$ is said to be linear dichotomies. These are the labellings for which there exists a hyperplane separating the points labelled ω_1 from the points labelled ω_2. It can be shown that this fraction is given by

$$f(n,d) = \begin{cases} 1 & \text{if } n \leq d+1 \\ \dfrac{2}{2^n} \displaystyle\sum_{i=0}^{d} \begin{pmatrix} n-1 \\ i \end{pmatrix} & \text{otherwise} \end{cases} \tag{8}$$

This means there may not exist a hyperplane to classify sample points if n is not less than $d+1$. However, if we think of a single-layer neural network as a linear pattern classifier, the number of connections becomes the number of degrees of freedom for the discriminant function, and the number of samples can be several times as large as the number of connections.

Since multi-layer neural network can act as nonlinear discriminants, the above results can not be directly used to analyze the multi-layer neural network classifier. By the way, because the basic idea of the neural network is to generate automatically the mapping rule in given data pairs of input and output, the complexity of the multi-layer classifier can be measured by the number of possible mappings that should be considered in the neural network.

Since we are given a neural network with d binary input neurons and c classes, the number of possible instances described by all these d features is 2^d, and the number of possible mappings considered by the neural network classifier is

$$_c\Pi_{2^d} = c^{2^d}. \tag{9}$$

On the contrary, if we decompose the c classes into k subclasses $c_1, c_2, \ldots, c_k$, then the

number of possible mappings becomes

$$ {}_k\Pi_{2^d} + \sum_{i=1}^{k} {}_{c_i}\Pi_{2^d} = k^{2^d} + \sum_{i=1}^{k} c_i^{2^d}. \tag{10} $$

However, the actual number of possible mappings required to classify a given sample point in subclass c_i is

$$ {}_k\Pi_{2^d} + {}_{c_i}\Pi_{2^d} = k^{2^d} + c_i^{2^d}. \tag{11} $$

For example, suppose we are given a neural network with 3 input nodes and 5 output nodes. Then, the number of possible mappings that may be treated is

$$ 5^{2^3} = 390,625. \tag{12} $$

However, if we decompose the given classification problem into two 2 and 3 class subproblems, the number of possible mappings required to classify a sample point in subclass 2 is

$$ 2^{2^3} + 3^{2^3} = 6,817. \tag{13} $$

Therefore, the load to neural network classifier is greatly reduced.

In order to appreciate the lower bound of the superiority of decomposition with respect to the number of possible mappings, assume the partitions of the first stage have unit spacing. If the number of possible mappings is N and the first stage classifier contains N^α divisions ($0 < \alpha \leq 1$), each second stage classifier contains $N^{1-\alpha}$ divisions. Then, the number of possible mappings needed to the two-stage classifier is $N^\alpha + N^{1-\alpha}$. If $\alpha = 1/2$, it becomes $2\sqrt{N}$. In this case, the complexity of the two-stage classifier is less than that of the single shot classifier if N is greater than 4.

4 Strategy II: Multiple Network Scheme

The networks train on a set of example patterns and discover relationships that distinguish the patterns. A network of a finite size does not often load a particular mapping completely

or it generalizes poorly. Increasing the size and number of hidden layers most often does not load 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. Therefore, if we can make the network consider the only specific part of the complete mapping, it will perform it's job better.

Several researchers have attempted to combine the outputs of independently trained networks to reach a decision [7]. Mori and Yokosawa [8] developed a large-scale neural network for recognizing Kanji characters by dividing the original task into several subtasks, with networks for each subtasks, and then integrating these subnetworks in a large network. Jacobs et al. [9] also proposed a modular version of a multilayer supervised network. It is a tightly coupled system composed of many separate networks, each of which learns to handle a subset of the complete set of training cases.

In this section, we present a multiple neural network architecture which is similar to the forementioned networks but takes generic non-neural methods for multinetwork combination. This approach may provide a possibility for incorporating any *a priori* information regarding the underlying problem to improve the ability of the network to generalize. Presently, several kinds of effort have been made toward this direction, and we propose to conduct a systematic investigation into the problem of multinetwork combination.

The basic idea of the multiple network scheme is to develop N independently trained neural networks with particular features, and to classify a given input pattern by obtaining a classification from each copy of the network and then using a consensus scheme to decide the collective classification by utilizing combination methods [10, 11]. Two general approaches, one based on fusion techniques and a second on voting techniques form the basis of the methods presented.

In the multiple network scheme, the output value of each node is taken to the estimated *a posteriori* probability of the training samples belonging to that class. Let $Y = \{y_1, y_2, \ldots, y_n\}$ be the set of output values of each network as shown in figure 3; $y_1 = \{y_{11}, y_{12}, \ldots, y_{1c}\}$, $y_2 = \{y_{21}, y_{22}, \ldots, y_{2c}\}, \ldots,$ $y_n = \{y_{n1}, y_{n2}, \ldots, y_{nc}\}$. Then it nat-

Figure 3: The multiple neural network scheme. N independently trained neural networks classify a given input pattern by using a consensus method to decide the collective classification.

urally raises the question of obtaining a consensus on the results of each individual network or expert.

The methods based on fusion techniques yield the final decision as w_I if

$$I = \arg\max_i y_i^*, \qquad 1 \le i \le c, \tag{14}$$

where y_i^* can be obtained by the following methods:

$$y_i^* = \begin{cases} \max\limits_{1\le k\le n} \{y_{ki}\}, & \text{optimistic;} \\ \min\limits_{1\le k\le n} \{y_{ki}\}, & \text{pessimistic;} \\ \dfrac{1}{n}\displaystyle\sum_{k=1}^{n} y_{ki}, & \text{indifferent.} \end{cases} \tag{15}$$

The Dempster-Shafer method for combining evidence may also be used.

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. For any particular

class c, the Borda count is the sum of the number of classes ranked below c by each network; Let $B_j(c)$ be the number of classes ranked below the class c by the jth network. Then, the Borda count for class c is $B(c) = \sum_{j=1}^{n} B_j(c)$. The final decision is given by selecting the class label whose Borda count is the largest. (Several literature have reported the usefulness of the voting procedures in classification area.)

5 Experimental Results

In order to give an idea of the practical application of the presented strategies in pattern recognition, a data set of handwriting characters has been used as a source of training and test patterns. Handwriting characters were inputted to the computer (SUN workstation) by an LCD tablet of Photron FIOS-6440 which samples 80 dots per second. The tasks were to classify the numerals, the uppercase letters, and the lowercase letters which were collected from 13 writers.

An input character consists of a set of strokes, each of which begins with a pen-down movement and ends with pen-up movements. Several preprocessing algorithms were applied to successive data points in a stroke to reduce quantization noises and fluctuations of the writer's pen motion. The processes used are as follows: the wild point reduction, the dot reduction, the hook analysis, the three point smoothing, the peak preserving filtering, and the N point normalization. A sequence of preprocessed data points is approximated by a sequence of 8-directional straight-line segments [12].

For training the neural network, 40 examples per each class were used, while for recognition a further 500 examples were used as test inputs. In order to compare the recognition results, we implemented a two-layered neural network which has 10 input nodes, 20 hidden nodes and 10 or 26 output nodes. The parameter values used for training were: learning rate is 0.25 and momentum parameter is 0.75. Each network was trained until an average squared error of 0.04 over the training set was obtained.

The groups at the coarse stage are as follows; (0, 6), (1, 5, 7), (4, 8, 9); (D, P), (E, F, I, O), (N, W, X, Y), (U, V); and (a, d, m), (b, p), (c, i, l), (e, z), (g, s), (h, n, y), (k,

Table 1: Recognition rates of multistage neural networks (%).

Subject	Simple NN	Multistage NN	
		Coarse	Fine
Numerals	77.4	83.2	77.8
Uppercases	73.2	81.8	76.0
Lowercases	59.0	73.8	68.8

q, t), (u, v). Table 1 shows the recognition rates of the multistage neural networks with the conventional ones. Though the proposed scheme did not improve the recognition rates remarkably, the training time was greatly shortened.

To evaluate the performance of the multiple network scheme, we implemented three different networks, including the previous one, each of which is a two-layered neural network having different number of input neurons and 20 hidden neurons. NN_1, NN_2 and NN_3 has 10 input neurons, 15 input neurons, and 20 input neurons, respectively. In this fashion each network makes the decision through its own resolution; NN_1 using sparsely sampled input produces the result by means of coarser view of input image, while NN_3 uses finer view. Thus, NN_1 has large possibility to overcome the variation or noise of input image though it utilizes rather blurred input. The consensus schemes used are the followings; (1) in fusion techniques, maximum, minimum, and average; (2) in voting techniques, unanimity, majority rule, and Borda count.

Figure 4 shows the recognition rates of numerals, uppercase letters, and lowercase letters with respect to the three different networks and their combinations by utilizing the consensus methods. In this figure, we can learn that the recognition rate of the consensus methods outperformed that of the individual networks in the cases of numerals and uppercase letters, but did not in the case of lowercase letters.

In order to understand why the consensus scheme did not produce better result than any of the individual networks in the lowercase letter, let's consider three networks with error rates e_1, e_2, and e_3. If we take the minimum error to be e_1, the condition that the

Figure 4: The recognition rates of the three different networks and their combinations.

consensus (e.g., majority rule) error rate be less than the best individual error rate is

$$e_1 > e_1e_2e_3 + (1 - e_1)e_2e_3 + e_1(1 - e_2)e_3 + e_1e_2(1 - e_3). \tag{16}$$

While the other two cases satisfy the above condition, the lowercase letter case does not (e_1 is 0.302, but the righthand side becomes 0.322).

It indicates that the average or the maximum may be adequate only if each neural network produces relatively reliable results. In this experiment, the fusion techniques were generally better than the voting technique. However, if the number of the different neural networks would be increased, the state of the results might be reversed.

Table 2 reports the aggregate of the recognition rates. Figure 5 illustrates the error rates of the multiple network scheme as compared with each network. In this figure, the consensus means the average in numerals and uppercase letters, and the minimum in lowercase letters.

Figure 5: A comparison of the error rates of the consensus scheme with the three neural networks.

Table 2: Recognition rates of multiple neural networks (%). Una and Maj stands for unanimity and majority, respectively.

Subject	NN_1	NN_2	NN_3	Max	Min	Avg	Una	Maj	Borda
Numerals	77.4	76.0	76.2	80.0	75.4	81.0	67.4	77.8	78.2
Uppercases	73.2	66.8	70.8	74.8	73.8	79.2	49.8	74.0	77.4
Lowercases	59.0	69.8	57.4	64.6	73.2	66.8	41.4	65.4	67.8

6 Concluding Remarks

In this paper, we present two design strategies of the neural network classifier, which are the multistage scheme and the multiple network scheme. The relative superiority to a single large neural network classifier is sketched by means of the comparison of the number of possible mappings to be considered in the neural networks. Initial trials to use the methods for classifying a large set of on-line handwriting characters were promising.

Several works are remained for further research. The relatively easy ones are to increase the recognition rate of each base classifier network for practical usage and to try the

same experiments with the increased number of classifiers. Furthermore, another fusion methods such as Fuzzy integral and Dempster-Shafer method may be applied to combine the decisions of the networks.

Acknowledgement

This work was supported (in part) by Korea Science and Engineering Foundation (KOSEF) through Center for Artificial Intelligence Research (CAIR), the Engineering Research Center (ERC) of Excellence Program.

References

[1] R.P. Lippmann, "Pattern Classification Using Neural Networks," *IEEE Communications Magazine*, November (1989) 47–64.

[2] F. Fogelman Soulie, "Neural Network Architectures and Algorithms: A perspective," *Artificial Neural Networks*, Elsevier Science Publishers B.V. (North-Holland; 1991) 605–615.

[3] S.-B. Cho and J.H. Kim, "A Two-Stage Classification Scheme with Backpropagation Neural Network Classifiers," *Pattern Recognition Letters*, **13**(5) (1992) 309–313.

[4] T. Matsuoka, H. Hamada and R. Nakatsu, "Syllable Recognition Using Integrated Neural Networks," in *Proc. Int. Joint Conf. Neural Networks*, **I** (1988) 251–258.

[5] A. Iwata, K.-I. Hotta, H. Matsuo, N. Suzumura, S. Matsuda and M. Yoshida, "A Large Scale Neural Network 'CombNET' on a Neural Network Accelerator (Neuro -turbo)," in *Proc. Int. Conf. Fuzzy Logic & Neural Networks* (1990) 329–333.

[6] H. El-shishiny, M.S. Abdel-mottaleb, M. El-raey and A. Shoukry, "A Multistage Algorithm for Fast Classification of Patterns," *Pattern Recognition Letters*, **10** (1989) 211–215.

[7] F. Fogelman Soulie, "Neural Network Architectures and Algorithms: A Perspective," *Artificial Neural Networks*, Elsevier Science Publishers B.V. (1991) 605–615.

[8] Y. Mori, and K. Yokosawa, "Neural Networks that Learn to Discriminate Similar Kanji Characters," *Advances in Neural Information Processing Systems I* (Morgan Kaufmann; 1989) 332–347.

[9] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation*, **3** (1991) 79–87.

[10] L.K. Hansen and P. Salamon, "Neural Network Ensembles," *IEEE Trans. Pattern Analysis and Machine Intelligence*, **12**(10) (1990) 993–1001.

[11] S. Shlien, "Multiple Binary Decision Tree Classifiers," *Pattern Recognition*, **23**(7) (1990) 757–763.

[12] H. Freeman, "Computer Processing in Line Drawing Images," *Computing Survey*, **6** (March 1974) 57–98.