Combining Modular Neural Networks Developed by Evolutionary Algorithm

Sung-Bae Cho

Dept. of Computer Science, Yonsei University, Seoul 120-749, Korea
ATR Human Information Processing Research Laboratories, Kyoto 619-02, Japan

Abstract— **Evolutionary approach to artificial neural networks has been rapidly developing in the recent years and shows a great possibility as a powerful tool. However, most evolutionary neural networks use the simple node as a building block to evolve, and select the only one network producing the best result after evolution. In this paper we present the concepts and methodologies for evolutionary modular neural networks which boost up the overall performance by combining several potential networks emerged in the course of evolution. The experimental result with the recognition problem of handwritten numerals shows the possibility of combining a number of characteristic networks from gene pool.**

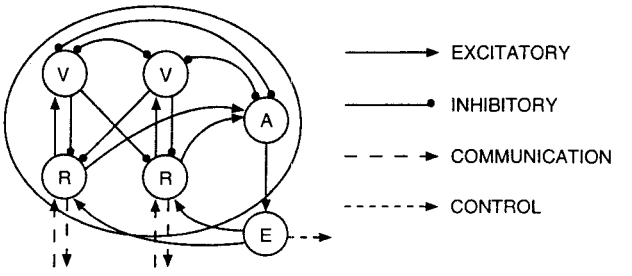

Fig. 1. Schematic diagram of the internal structure of a module.

I. Introduction

Most of the currently popular network architectures show little structural constraints. Some networks assume total connectivity between all nodes. Others assume a hierarchical, multi-layered structure where each node in a layer is connected to all nodes in neighboring layers. However, there is a large body of neuropsychological evidence showing that the human information processing system consists of modules, which are subdivisions in identifiable parts, each with its own purpose or function.

To design the information processing system with various modules, the concept of evolving neural networks has appeared recently as a subject of intensive scientific discussions from biological, social, and engineering points of view [1; 2; 3]. We proposed an evolvable model of modular neural networks in the previous work [4]. In this model we choose the best network with a winner-take-all model selection. However, recent theoretical and experimental work indicates that we can improve performance by considering methods for combining neural networks [5; 6; 7].

This paper reports the improvement of the evolutionary modular neural networks by combining with several methods. Although a serious theoretical investigation on the generality of the results is beyond the scope of this paper, we will demonstrate the effectiveness by experimental results on a recognition problem of handwritten numerals.

II. Evolutionary Modular Neural Networks

A. Overview

The basic idea of our approach is to consider a module as a building block resulting in local representations by competition, and develop complex intermodule connections with evolutionary mechanism. The initial network architecture is encoded as a gene. We have used a genetic encoding scheme which stores the wiring diagram for the network in tree structure. The encoding method has been developed to be robust with respect to the genetic operators such as mutation and crossover.

B. Modular Neural Networks

The activation value of each node is calculated as follows:

$$e_i = \sum_j w_{ij} a_j(t),$$

where w_{ij} denotes the weight of a connection from node j to node i. The effective input to node i, e_i, is the weighted sum of the individual activations of all nodes connected to the input side of the node. The input may be either positive (excitatory) or negative (inhibitory).

A module based on such nodes is designed to model neocortical minicolumns, as first proposed by J.M.J. Murre [8]. The constraints embodied in the model include:

1. Dale's principle (that individual neurons emit only one type of transmitter),
2. learning as a local phenomenon that does not require knowledge of the correct response, and
3. the capacity to differentiate between novel and familiar input and behave differently on that basis.

The internal structure of each module is fixed and the weights of all intramodular connections are non-modifiable during learning process (see Fig. 1).

In a module, R-node represents a particular pattern of input activations to a module, V-node inhibits all other nodes in a module, A-node activates a positive function of the amount of competition in a module, and E-node activation is a measure of the level of competition going on in a module. The most important feature of a module is to autonomously categorize input activation patterns into

0-7803-3949-5/97/$10.00 ©1997 IEEE

discrete categories, which is facilitated as the association of an input pattern with a unique R-node.

The process goes with the resolution of a winner-take-all competition between all R-nodes activated by input. In the first presentation of a pattern to a module, all R-nodes are activated equally, which results in a state of maximal competition. It is resolved by the inhibitory V-nodes and a state-dependent noise mechanism. The noise is proportional to the amount of competition, as measured through the number of active R-nodes by the A-node and E-node. Evolutionary mechanism gives a possibility of change to the phenotype of a module through the genetic operators.

The interconnection between two modules means that all R-nodes in one module are connected to all R-nodes in the other module. These intermodule connections are modifiable by Hebb rule with the following equation:

$$\Delta w_{ij}(t+1) = \mu_t a_i \left([K - w_{ij}(t)]\, a_j - L w_{ij}(t) \sum_{f \neq j} w_{if}(t) a_f \right),$$

$$\mu_t = d + w_{\mu_E} a_E,$$

where a_i, a_j and a_f are activations of the corresponding nodes, respectively; $w_{ij}(t)$ is the interweight between R-nodes j and i, $w_{if}(t)$ indicates an interweight from a neighboring R-node f (of j) to R-node i, and $\Delta w_{ij}(t+1)$ is the change in the weight from j to i at time $t+1$. Note that L and K are positive constants, and a_E is the activation of the E-node. As a mechanism for generating change and integrating the changes into a system, we use module duplication and elimination, and genetic algorithm to determine the parameters in the above learning rule and structure of intermodule connections.

C. Gene Representation and Operators

Three kinds of information should be encoded in the genotype representation: the structure of intermodule connection, the number of nodes in each module, and the parameters of learning and activation rules. The intermodule weights are determined by the Hebb rule mentioned at the previous section. In order to represent the information appropriately, a tree-like structure has been adopted. An arc in a tree expresses an intermodule connection, and each node represents a specific module and the number of nodes therein.

An example of the genotype is shown in Fig. 2. Each node has a number representing a specific module. In this figure other information such as the number of nodes and parameters of the learning and activation rules are omitted. The root of the tree is the input module that replaces the start symbol. A child node has a module number to be applied to the symbols which represent the modules connected by its mother module.

By performing the genetic operators in the gene pool, the interconnection between modules as well as the number of them are changed. Designing the gene to represent the interconnectivity makes it possible to generate a variety of offsprings and to evolve them. For the details in the operators used, refer to the previous publication [4].

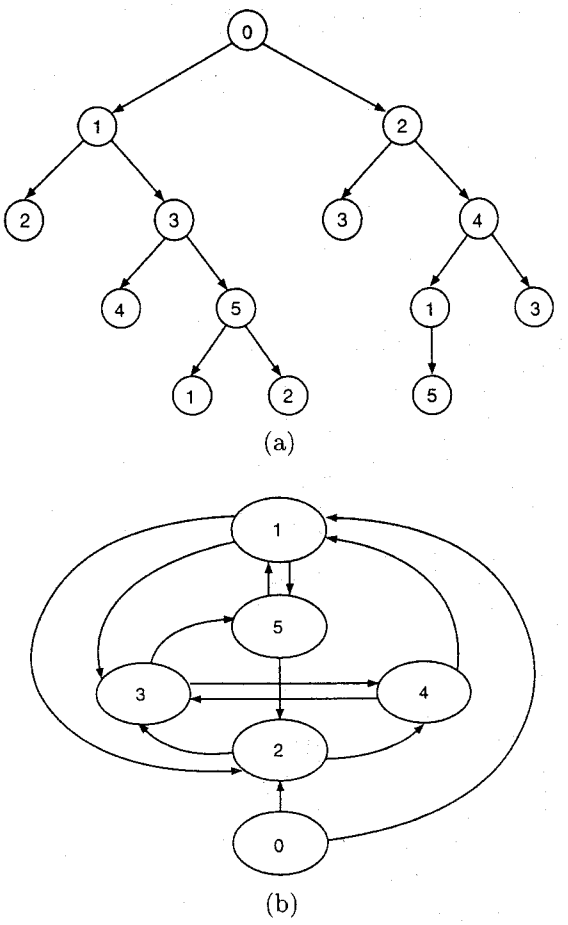

Fig. 2. (a) Genetic code encoded by tree structure; (b) A modular neural network architecture developed by the gene of diagram (a).

III. Combining Modular Neural Networks

The modular neural network developed by the evolutionary algorithm 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. Especially, 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 presented method is to collect n best modular neural networks that evolved independently, and to classify a given input pattern by utilizing combination methods to decide the final classification [7] (see Fig. 3). In the terminology of pattern recognition, the classifications of an input X based on a set of real value measurements can be written as follows:

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

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

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

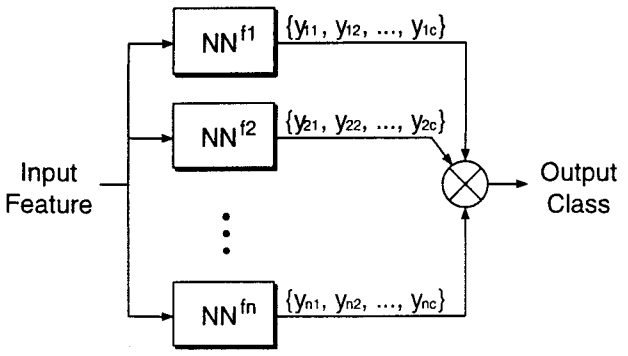

Fig. 3. The multiple modular neural networks combined.

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:

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

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:

$$P(\omega_i|X) = \sum_{k=1}^{n} r_k P_k(\omega_i|X), \quad 1 \le i \le c,$$

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

Another alternative is to use the maximum value of $P_k(\omega_i|X)$ denoted by $P_m(\omega_i|X)$, to replace the correspondent average value. Since $\sum_{i=1}^{c} P_m(\omega_i|X) \neq 1$, we use the following normalized values as the new estimations:

$$P(\omega_i|X) = \frac{P_m(\omega_i|X)}{\sum_{j=1}^{c} P_m(\omega_j|X)}, \quad 1 \le i \le c.$$

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 will introduce the two of them: majority voting and Borda count.

The majority voting rule chooses the classification made by more than half the networks. When there is no agreement among more than half the networks, the result is considered an error. To appreciate the network performance, assume that all neural networks arrive at the correct classification with a certain likelihood $1-p$ and that they make independent errors. The chances of seeing exactly k errors among n copies of the network is then

$$\binom{n}{k} p^k (1-p)^{n-k}$$

which gives the following likelihood of the majority rule being in error

$$\sum_{k>n/2}^{n} \binom{n}{k} p^k (1-p)^{n-k}.$$

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.

IV. Simulation Results

In order to confirm the possibility of the proposed model, a data set of handwritten numerals was used as a source of both training and test samples. A sample of the task was initially obtained by having different writers draw 200 digits within prepared square boxes in order to facilitate segmentation. The size of a pattern was normalized by fitting a coarse, 10×10 grid over each digit. The proportion of blackness in each square of the grid provided 100 continuous activation values for each pattern. Network architectures generated by the evolutionary mechanism were trained with 100 patterns in two rounds of subsequent presentations. A single presentation lasted for 60 cycles (i.e., iterative updates of all activations and learning weights). A fitness value was assigned to a solution by testing the generalization performance of a trained network with the untrained half of the 200 digits.

Initial population consisted of 50 neural networks of having random connections. Each network contains one input module of size 100, one output module of size 10, and different number of hidden modules. Every module can be connected to every other module. Nearly all the best solutions present in the population after a few number of generations scored 100% correct recognition for the training set, but the generalization rates were gradually improved as the generation goes.

In the early stages of the evolution some complicated architectures emerged, but they were disappeared as the search of the optimal solution matured. The earlier good specific solutions probably overfitted the training set with lack of generality. Fig. 4 shows some of the modular neural networks evolved.

To evaluate the performance of the proposed method, we used the three modular networks in Fig. 4 and obtained the results with combining methods. Table 1 shows the recognition rates with respect to the three different networks and their combinations by utilizing consensus methods. The reliability in the table is computed as the following equation:

$$\text{Reliability} = \frac{\text{Correct}}{\text{Correct} + \text{Error}} \times 100$$

As can be seen, every method of combining multiple networks produces better results than individual networks.

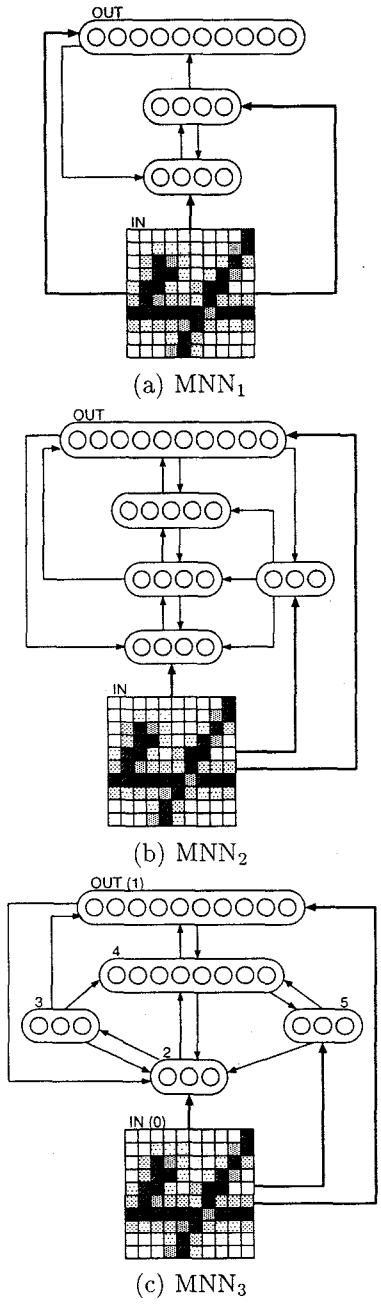

Fig. 4. Some of the modular neural networks evolved.

Table 1. The result of recognition rates (%).

Methods	Correct	Error	Reject	Reliability
MNN_1	92	8	0	92.00
MNN_2	90	10	0	90.00
MNN_3	89	11	0	89.00
Average	94	6	0	94.00
wAverage	95	5	0	95.00
Maximum	93	7	0	93.00
Voting	95	4	1	95.96
Borda Count	94	6	0	94.00

V. Concluding Remarks

We have presented the evolutionary modular neural networks that boost up the performance by using the ensemble of several characteristic networks from gene pool. Each modular network has a basic structure with intramodular competition, and intermodular excitatory connections.

Future efforts will concentrate on extending the method with the concept of co-evolution, which evolves each modular network as well as the combination method. This sort of network will also take an important part in several engineering tasks exhibiting adaptive behaviors. Furthermore, we are exploring the concept of fitness sharing to give the population kind of diversity [9], which might lead a natural way to select the multiple networks from the final population.

Acknowledgements

This work was supported in part by a grant no. 961-0901-009-2 from the Korea Science and Engineering Foundation (KOSEF) and a grant from the Ministry of Science and Technology in Korea. The author would like to thank Y. Tohkura and K. Shimohara at ATR HIP laboratories for continuous encouragement.

References

[1] S.A. Harp, "Towards the genetic synthesis of neural networks," in *Proc. Genetic Algorithms*, pp. 360–369, 1989.

[2] H. Kitano, "Designing neural networks using genetic algorithms with graph generation system," *Complex Systems*, vol. 4, no. 4, pp. 461–476, 1990.

[3] X. Yao, "Evolutionary artificial neural networks," *Int. Journal of Neural Systems*, vol. 4, no. 3, pp. 203–222, 1993.

[4] S.-B. Cho and K. Shimohara, "Modular neural networks evolved by genetic programming," in *Proc. Int. Conf. Evolutionary Computation*, pp. 681–684, Nagoya, May 1996.

[5] R.A. Jacobs, M.I. Jordan, S.J. Nowlan and G.E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, pp. 79–87, 1991.

[6] J.B. Hampshire II and A. Waibel, "The meta-pi network: Building distributed knowledge representations for robust multisource pattern recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 7, pp. 751–769, 1992.

[7] S.-B. Cho and J.H. Kim, "Multiple network fusion using fuzzy logic," *IEEE Trans. Neural Networks*, vol. 6, no. 2, pp. 497–501, 1995.

[8] J.M.J. Murre, R.H. Phaf and G. Wolters, "CALM: Categorizing and learning module," *Neural Networks*, vol. 5, pp. 55–82, 1992.

[9] P. Darwen and X. Yao, "Automatic modularization by speciation," in *Proc. IEEE Int. Conf. Evolutionary Computation*, pp. 659–664, Nagoya, May 1996.