A New Indirect Encoding Method with Variable Length Gene Code to Optimize Neural Network Structures

Kunikazu Kobayashi† and Masanao Ohbayashi

Department of Computer Science & Systems Engineering,
Faculty of Engineering, Yamaguchi University
†e-mail: k@csse.yamaguchi-u.ac.jp

Abstract

A new encoding method for optimizing neural network structures is proposed. It is based on an indirect encoding method with variable length gene code. The search ability for finding an optimal solution is higher than the direct encoding methods because redundant information in gene code is reduced and the search space is also reduced. The proposed method easily operates adding and deleting hidden units. The performance of the proposed method is evaluated through computer simulations.

Introduction

In designing neural networks (NNs), it is crucial to optimize network structures to derive both optimal approximation ability and generation ability

Genetic algorithm (GA) [1, 2] has been successfully applied to variety kinds of optimization problems, such as traveling salesman problem and scheduling problem. In neural computing, there are also a lot of work on designing NNs using GA. One of the most important points is how to encode a network structure and parameters of NN (phenotype) in gene code (genotype), i.e. coding scheme.

Dasgupta et al. proposed a design algorithm using structured GA based on a direct encoding method [3], where the network structure and the parameters are directory encoded in gene strings and the optimal solutions of both are searched by GA. On the other hand, Kitano proposed NGL (Neurogenetic Learning) [4]. NGL is based on an indirect encoding method and has interesting properties.

In the present paper, a new encoding method for optimizing neural network structures is proposed. It is based on the indirect encoding method with variable length gene code. The search ability for finding an optimal solution is higher than than the direct encoding methods because redundant information in gene code is reduced and the search space is also reduced. It is also possible to easily add or delete hidden units. The performance of the proposed method is evaluated through computer simulations.

Encoding Method

A coding scheme is shown in Fig.1. This can be applicable to three-layered NNs.

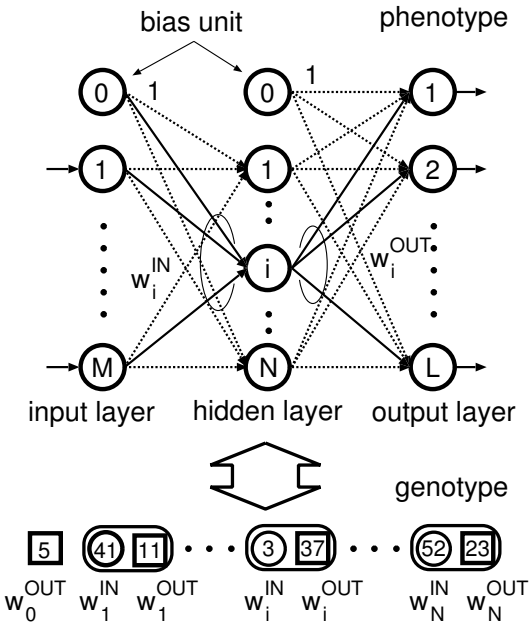

Figure 1: Coding scheme

The two weight vectors of fan-in and fan-out with respect to hidden unit i are denoted by $w_i^{IN}(i = 1 \sim N)$ and $w_i^{OUT}(i = 0 \sim N)$, respectively.

$$w_i^{IN} = \{w_{i0}^{IN}, w_{i1}^{IN}, \cdots, w_{ij}^{IN}, \cdots, w_{iN}^{IN}\}, \quad (1)$$

$$w_i^{OUT} = \{w_{i1}^{OUT}, \cdots, w_{ik}^{OUT}, \cdots, w_{iM}^{OUT}\}, \quad (2)$$

where w_{i0}^{IN} is a weight vector to bias unit 0 whose output is always 1 in the input layer, i.e. a bias of hidden unit i.

As seen in Fig.2, w_i^{IN} and w_i^{OUT} have a parameter table with size P and Q, respectively. The elements of these two tables are initialized with random real number. Then, a gene string is defined by a sequence of integers. An integer in each locus represents a set of weights, w_i^{IN} in the even position or w_i^{OUT} in the odd position. Initially they are randomly given by positive integers from $1 \sim P$ and $1 \sim Q$, respectively. As a result, the length of the gene string is $2N + 1$ and is variable with network size, N. That is, a different NN has a different length gene string. Our method has a variable length gene code.

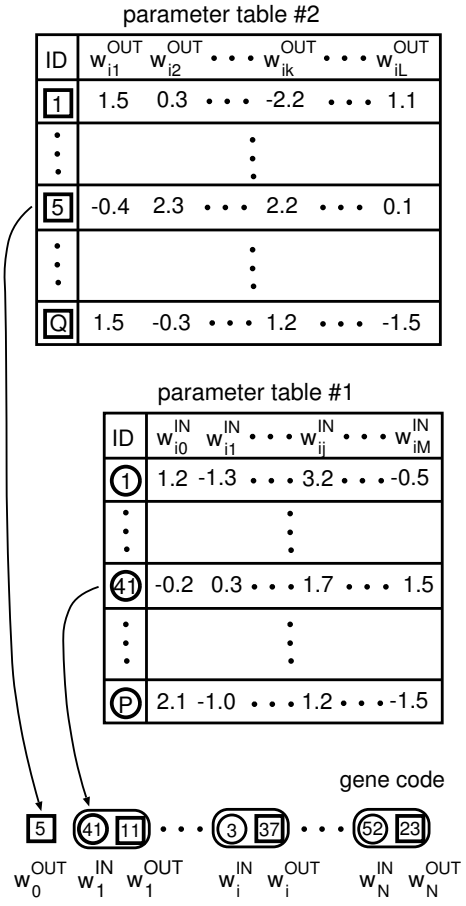

Figure 2: Parameter tables

An optimal network structure is searched by GA and its parameters are updated by backpropagation (BP) algorithm [5]. A fitness function is defined as follows.

$$\text{fitness} = \frac{1}{E} + \frac{c}{N}, \tag{3}$$

where c is a weighting parameter between error, E, and the number of hidden units, N.

The standard genetic operators, i.e. selection, crossover and mutation are used; roulette selection with elite preserving strategy, one-point crossover with probability $P_{\text{crossover}}$ and mutation which replaces an integer in a locus with a different integer with probability P_{mutation}.

In addition to these genetic operators, two more new operators, i.e. addition and deletion, shown in Fig.3 are introduced in the present paper. The addition adds two integers, which corresponds to one hidden unit, to the last position and the deletion deletes two integers. These operators play an important rule on finding the optimal network structures.

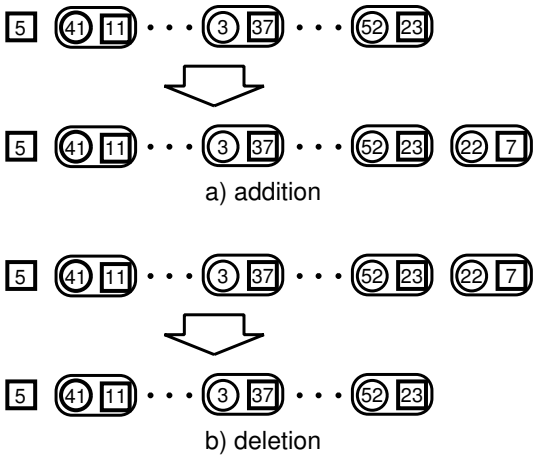

Figure 3: Two genetic operators: addition and deletion

Moreover, an operation to the parameter table is proposed. After each generation, i.e. after the BP training is finished, since the best individual may have a global solution its parameters in both tables (See Fig.2) are replaced with the values of its solution.

Computer Simulations

The computer simulations were carried out to evaluate the performance of our method. The following benchmark problems were used; three binary problems (XOR, 4-bit parity, and 8-bit encoding-decoding problem), two pattern recognition tasks (digit in 10x10 pixels and 20 alphabets; $A \sim T$ in 5x3 pixels) and two function approximation tasks (target functions; $\sin(x)$ and $x(x-2)(x+1)^2$). Our method was compared with two other methods, i.e. the direct encoding method (DEM) proposed by Dasgupta et al [3] and NGL (Neurogenetic Learning) proposed by Kitano [4].

The simulation condition is described as follows:

$$\begin{aligned} P = Q &= 50 \\ w_i^{IN}(0) &\in [-2.0, 2.0] \\ w_i^{OUT}(0) &\in [-2.0, 2.0] \\ c &= 10 \end{aligned}$$

population size = 30
the number of generation = 30
$P_{\text{crossover}}$ = 0.9
P_{mutation} = 0.03
addition rate = 0.1
deletion rate = 0.01
the number of BP training = 1000
slope of sigmoid function = 1.0
learning coefficient = 0.05
momentum coefficient = 0.9

Table 1 shows the results for 8-bit encoding-decoding problem. In this table, CPU time represents the relative value with respect to that of NGL which is assumed to 100. Then success means the successful rate in 10 trials. Moreover, N_0 and $\overline{N}_\infty$ are the number of hidden units in the initial state and the average of the number of hidden units in the final state. In DEM, the initial number of hidden units is varied from 3 to 7.

As seen in this table, both IEM and NGL achieve a perfect successful rate but IEM has lower CPU time and smaller number of hidden units than NGL. On the other hand, DEM has lower CPU time in most cases but the successful rates are much lower than IEM. As a result, it is shown that our method (IEM) has better performance than DEM and NGL from the point of both computational cost and network size.

Table 1: Result of 8-bit encoding-decoding problem

method	CPU time	success [%]	$\overline{N}_\infty$	N_0
IEM	42	100	3.4	16
NGL	100	100	13.1	16
DEM	28	0	-	3
	34	16	4.0	4
	40	42	5.0	5
	45	88	6.0	6
	50	96	7.0	7

The similar results were obtained for other benchmark problems described above. Two of them are shown in Table 2 and 3).

The key point of the present paper is to use of variable length gene code in the indirect encoding method. Since redundant information included in gene code is reduced compared with the direct encoding method the search ability is improved. In NGL, since the number of units is restricted in a power of 2 the ability to find an optimal network structure is lower than our method.

Conclusions

Table 2: Result of digit recognition task

method	CPU time	success [%]	$\overline{N}_\infty$	N_0
IEM	29	100	4.7	16
NGL	100	100	12.3	16
DEM	18	0	-	4
	47	4	5.0	5
	56	40	6.0	6
	63	74	6.8	7
	70	94	7.7	8

Table 3: Result of function approximation task, $f(x) = \sin(x)$

method	CPU time	success [%]	$\overline{N}_\infty$	N_0
IEM	63	100	3.4	16
NGL	100	12	13.1	16
DEM	19	0	-	2
	25	44	4.0	3
	34	76	5.0	4
	41	84	6.0	5
	47	84	7.0	6

In the present paper, we have proposed the new indirect encoding method with variable length gene code and the algorithm to construct NNs using GA. Through various computer simulations, it is shown that our method has better performance than DEM and NGL on improvement of search ability and reduction of computational cost.

The trade-off parameter c in the fitness function, eq.(3), is determined in advance and fixed over generation. However, the value of c should be small in the early generation and should be big in the late generation. To develop an algorithm to determine the optimal value of trade-off parameter c or to define more feasible fitness function is one of our future problems.

Acknowledgment

The present work was partly supported by Electric Technology Research Foundation of Chugoku.

References

[1] J. H. Holland. The dynamics of searches directed by genetic algorithms. In *Evolution, Learning, and Cognition*, pages 111–127. World Scientific, 1988.

[2] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[3] D. Dasgupta and D. R. McGregor. Designing application-specific neural networks using the structured genetic algorithm. In *International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pages 87–96, 1992.

[4] H. Kitano. Neurogenetic learning: An integrated method of designing and training neural networks using genetic algorithms. *Physica D*, 75:225–238, 1994.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, volume 1, chapter 8, pages 318–362. Bradford Books/MIT Press, 1985.