ISSN 1009 - 3095 *Journal of Zhejiang University(SCIENCE)* V.1, No.3, P.322 - 326, July - Sep., 2000
http://www.chinainfo.gov.cn/periodical; http://www.zju.edu.cn;
http://lib.zju.edu.cn/English; http://www.zjupress.com

AN IMPROVED GENETIC ALGORITHM FOR TRAINING LAYERED FEEDFORWARD NEURAL NETWORKS*

LIU Ping (刘 平), CHENG Yi-yu (程翼宇)**,
(*Dept. of Chemical Engineering, Zhejiang University, Hangzhou, 310027, China*)
Received May.10, 1999; revision accepted May.22, 2000

Abstract: The new genetic algorithm for training layered feedforward neural networks proposed here uses a mutation operator for performing the search behaviors of local optimization. Combining the random restart method with the local search technique, the algorithm can converge asymptotically to the optimal solution. Test with a practical example showed that the improved genetic algorithm is more efficient than the conventional genetic algorithm.

Key words: artificial neural network, genetic algorithms, layered feedforward neural networks
Document code: A **CLC number:** TP18

INTRODUCTION

The most common tool for training layered feedforward neural networks is the backpropagation training algorithm, which has been shown to be effective for a number of problems. However, there are several disadvantages in using this approach. A limitation of the backpropagation algorithm results from the "scaling problem" (Montana et al., 1989). Due to the fact that the large amount of execution time required by large networks slows down the learning rate. This stems from the fact that gradient search techniques tend to get trapped at local minimum. The existence of local minimum may also be attributed to another drawback in the use of backpropagation. Since the algorithm does not guarantee an approximation, there are many cases in which it will oscillate forever. Furthermore, since backpropagation is based on gradient descent, it will not work in cases where the search criterion or function is non-continuous. One last drawback is that the ability of the backpropagation algorithm to find an approximation for a given problem is very dependent on the network structure. In other words, it seems that the use of this approach is not domain independent (Byungjoo et al., 1994).

Genetic algorithms (GAs) (Chen et al., 1996) are a class of probabilistic search and optimization algorithms gleaned from the model of organic evolution. They are highly parallel and believed to be robust in searching global optimal solution of complex optimization problems. GAs require from the environment only the payoff of penalty (i.e., objective) function called, the fitness function measuring the fitness score of each individual and no other information nor assumptions such as derivatives and differentiability. It is theoretically and empirically proven that GAs can find the global optimum. These advantages lead researchers to propose the use of a genetic algorithm as an alternative method for training neural networks. In general, the approximation characteristic of a genetic algorithm is such that, in contrast to backpropagation, the error rate decreases monotonically. Nevertheless, the premature convergence of GAs has been noted. Furthermore, although the rate of convergence is very fast during the early stages of the genetic algorithm, a drastic reduction in convergence velocity in the latter generations is often encountered before GAs provides an accurate solution. This is an illustration of one of the disadvantages of genetic algorithms, i.e. there is no mechanism for local fine-tuning (Kitano, 1990).

In this paper, a new genetic algorithm, called Mutation-based Genetic Algorithm (MGA), is proposed. The algorithm uses the

* Project supported by NSFC (No 39870940) and (G1999054405 - 973) the National Key Scientific Research & Development Porgram.
** Corresponding author.

mutation operator as the unique genetic operator to perform local search. Upon reaching the local optimum, the random restart technique is applied. The performance of the algorithm is compared with that of the conventional GA (CGA) by a practical example in training layered feedforward neural network. The results showed that the new algorithm has strong local convergence ability and easily and rapidly find a satisfactory solution.

IMPROVED GENETIC ALGORITHM

Since the crossover operator and the selection operator (which operates on the principle that the fitter survives) exert serious adverse effect on maturation, their employment may decrease the degree of population diversity and degrade the search capability of the GAs (Xu et al., 1996). As a result, premature convergence occurs and the search becomes trapped in a local region. On the other hand, though mutation is performed with a very low probability, it is of great importance to maintain the genetic diversity of the population and strengthen the local convergence ability of GAs. However, the randomness in performing mutation may cause a replacement of an important bit value, and consequently the fast convergence to a good solution may be affected. Sometimes it diverges the homogeneity of the population during final generations, which may delay the process of convergence. Therefore, it will be a new research approach for GAs, if only the mutation operator with high probability of occurrence is used and if it is guaranteed that the individual produced by mutation does not differ genetically very much from its ancestor.

Combining random restart method with local search, an improved genetic algorithm, called mutation-based genetic algorithm, is proposed. The new method is a stochastic iterative algorithm. Each iteration of the algorithm consists of two steps: generation of new parent individual (i.e., initial solution) and the local search performed by the mutation operator.

Suppose a layered feedforward neural network has n weight variables. A randomly coded binary bit string of length $L = n \times m$ (sub-string of length m is assumed for each variable), which represents a code version of an initial solution for the local search and an existing optimal solution obtained during the evolutionary process, is taken as the initial parent individual P.

Based on the initial parent individual, the algorithm uses the mutation operator to search the solution space. To select a high mutation rate, each mutation operation only acts on a single variable, not on all variables. Namely, only those genes that represent the variable i undergo mutation operation with respective mutation rates while the mutation rates for the others are zero.

Usually, the output error function of the neural network is a complex multimodal function. For each modality of multimodal function, the bottom of its modality is larger and the top is smaller. According to this characteristic, the algorithm first searches the solution space in a medium range at medium precision. After searching for a period of time, the search is performed in a large range at low precision. Then the search range is decreased and the search is done in a medium range at medium precision, and finally the algorithm searches in a small range at high precision. Applying this strategy, the algorithm can improve the convergence ability effectively and yield a satisfactory result in a limited time.

During generations, the search range will alter self-adaptively if the parent individual keeps constant in t generations. If t is too small, the final computational accuracy will be poor. On the contrary, a large t will increase the generation number and make the algorithm quite time consuming. Usually, if the modality of the function is unknown or complex, or an exact solution is required, larger t is selected.

To control the search range and precision, different mutation rates, instead of a unique mutation rate in the CGA, are distributed to the genes of the variable i. Meanwhile, these rates will vary with the change of search range during generations.

Each mutation operation generates an offspring individual and each variable should be implemented at least one mutation operation in each generation, so the offspring population size is $N = n \times d\,(d \geq 1)$, where the integer d represents the number of mutation operations performed on each variable in one generation.

After at least one mutation operation has been performed on each variable, the individual that has the maximum fitness value is selected as

the new parent individual P from the parent and offspring population; and a better solution is obtained.

Apparently, the search works on a unique solution. So the search method performed by mutation is local in scope over the solution space. It can improve the local convergence ability of the algorithm but may make the search trapped in a local optimal region. Upon reaching the local optimum, the random restart or random choice of a new starting value for the search is applied. The parent individual is reinitialized and the algorithm is restarted using the new parent individual to explore the new region.

If the parent individual does not change in t generations when the algorithm searches the solution space in a small range, it can be considered that the search converges to a local optimum. At this time, the existing parent individual that represents a possible optimal solution is kept. A new parent individual is generated randomly to restart a new search to explore other regions.

The above iteration is repeated until the upper limit for number of generations has been reached or the fitness value of the optimal individual satisfies the predetermined request.

In the algorithm, the principle for setting the mutation rate per gene of each variable is as follows.

In the large range at low precision, mutation rates of a variable reduce gradually from high to low bits. In the medium range at medium precision, mutation rates reduce from the middle bits to both sides. In the small range at high precision, mutation rates reduce gradually from low to high bits.

If the mutation rate is too large, the gene will change too frequently to be advantageous to optimization. So the value of mutation probability is not higher than 0.6. Since the search of the algorithm is stochastic based on the probability rules, the determination of the mutation rate of each gene has no accurate criterion. Usually, except that the mutation rates of those genes that locate in high position should be smaller than 0.1 in small-range search, the mutation rates are selected from 0.1 to 0.6 according to the above principle. These values can feasibly yield the corresponding search range and precision as a whole.

The mutation rate for each gene can be determined approximately by the above-mentioned principle as long as the length of the sub-string for each variable is fixed. In other words, the selection of the mutation rate is independent of the studied problems.

The main algorithm structure for mutation-based evolution is given as follows:

Algorithm

Step 1. Initialize the parent individual P and specify the size N of offspring population, control parameters and stopping criterion.

Step 2. Perform mutation operation and produce N offspring individuals.

Step 3. Calculate the fitness value of offspring individuals and select the individual, which has the maximum fitness value, from all $N + 1$ individuals as the new parent individual P.

Step 4. If P keeps constant in t generations, then alter the search range automatically.

Step 5. If P does not change in t generations when the algorithm searches the solution space in a small range, it can be considered that the search has converged to a local optimum. Save the current parent individual (i.e. optimal solution) and go to step 6. Otherwise, go to step 2.

Step 6. If the stopping criterion is not satisfied, then reinitialize the parent individual and go to step 2. Otherwise, terminate the algorithm.

The algorithm routines are written in C language and were compiled and tested on an Intel-MMX 166 PC with the Visual C^{++} compiler (version 5.0) under WIN95 operating system.

RESULTS AND DISCUSSIONS

The performance of MGA was compared with that of the CGA by using a practical example. Ranking selection, a two-point crossover operator, and a bit-reversal mutation operator were applied in the CGA. However, the crossover operation and the mutation operation with high frequency will destroy the obtained model. To avoid the premature convergence of the CGA, adaptive crossover and mutation operations were adopted. In the early generations the crossover rate and the mutation rate were kept high and they de-

creased with time. In this paper, the initial value of the crossover rate and the mutation rate are 1.0 and 0.01, respectively. The formula that adaptive change the rate of crossover and mutation is defined as follow:

$$P = P_0\left(1 - \frac{g}{2G_{\max}}\right)^2 \quad (1)$$

Where P_0 is the initial value of the crossover rate and the mutation rate, g is the current generation and $G_{\max}$ is the upper limit for the number of generations.

A three-layer feedforward neural network with 9 input nodes (including one threshold one), 4 hidden layer nodes, and 1 output node is used to implement the training and prediction for the PVT properties of 40 alkane compounds. The input parameters consist of the molecular connectivity indexes, temperature (T) and pressure (P) of the compounds. The output is the volume (V) of compounds. These compounds' PVT data used in this paper come from references (Buford, 1986; Vargaftik, 1975). Twenty-five compounds were selected from 40 alkanes and 15 ~ 20 PVT data were extracted randomly from each compound to form the training group. The rest ot the data of these compounds (usually, 8 ~ 12 PVT data in each compound) were collected in the correlative group. The predictive group consisted of the PVT data from the other 15 compounds.

The fitness function, the training and predictive precision of the network are defined as follows:

$$F(x) = 2.0\sum_{n_c}(y_{\text{norm}}^{\text{cal}} - y_{\text{norm}})^2 / \sum_{n_c} y_{\text{norm-range}}^2 \quad (2)$$

$$\text{SDEC} = \left(\sum_{n_c}(y^{\text{cal}} - y)^2/n_c\right)^{1/2} \quad (3)$$

$$\text{SDEC} = \left(\sum_{n_p}(y^{\text{pred}} - y)^2/n_p\right)^{1/2} \quad (4)$$

Here, $y_{\text{norm}}^{\text{cal}}$, y_{norm}, $y_{\text{norm-range}}$ and represent, respectively, the normalized obtained output, the normalized target output and the normalized output range. y^{cal}, y^{pred}, and y denote the obtained training output, the predictive (test) output and the real output, respectively. n_c and n_p are the number of samples used for the training and the prediction.

The training and test plan is given below.

(1) Since the structure of the network is 9 - 4 - 1, the total number of weight parameters is $n = 40$. Due to space limitations, each parameter is encoded using 12 bits only, i.e. $m = 12$;

(2) The population size of CGA and the offspring population size of MGA are $N = 40 \times 3 = 120$;

(3) For MGA, $t = 35$, the mutation rates per gene of each variable is determined below:

$P_m^{9,10,11} = 0.5$, $P_m^{6,7,8} = 0.4$, $P_m^{3,4,5} = 0.2$, $P_m^{0,1,2} = 0.1$ (large-range).

$P_m^{5,6} = 0.6$, $P_m^{4,7} = 0.5$, $P_m^{3,8} = 0.35$, $P_m^{2,9} = 0.2$, $P_m^{1,10} = 0.1$, $P_m^{0,11} = 0.05$ (medium-range)

$P_m^{0,1,2,3,4} = 0.5$, $P_m^{5,6} = 0.3$, $P_m^{7,8} = 0.1$, $P_m^{9,10,11} = 0.01$ (small-range)

(4) The algorithms to stop the search after $G_{\max} = 20000$;

(5) The number of training samples is 305. The testing samples are divided into two groups. The first, called correlative group, consists of the data that are not included in the training group from those 25 compounds whose molecular connectivity indexes take part in the network training. The sample number of this group is 262. The second, called predictive group, is made up of the data from 15 compounds whose molecular connectivity indexes and PVT data are not provided to the network training. The number of this group is 345.

Table 1 shows the training and testing results of the neural network using MGA and the CGA, respectively.

It is seen from Table 1 that the training precision of MGA is obviously higher than that of the CGA. The testing results confirm that MGA can reappearance learning production exactly.

CONCLUSION

A new genetic algorithm combined random search with local minimization is proposed. Four different features distinguish the algorithm from the CGA. a) The mutation operator of the new algorithm, which only acts on single variable,

Table 1 Results for neural network training and testing

	Training			Testing			
	$F(x)$	SDEC	R	SDEP (correlative)	R	SDEP (predictive)	R
MGA	1.99982	2.15022	0.99775	2.28739	0.99736	2.37832	0.99405
CGA	1.99733	8.24294	0.96701	8.34680	0.96508	11.26389	0.88746

can effectively prevent serious change on the structure of the existing best individual and optimize the individual to the better structure gradually during the evolutionary process. Its effect is similar to that of the crossover operator in the CGA. So the crossover operation is omitted by the new method. b) Since the parent generation has only one individual, not a population consisting of a set of individuals, the selection operation of the algorithm is simpler than the CGA. c) The mutation operation of the algorithm only acts on those genes that represent the variable i and the mutation rates of the other genes are zeros, while the simple mutation operator of the CGA acts on for each gene of the individual with the same mutation rate. Furthermore, in the algorithm, different time-varying mutation rates are distributed to those genes of the variable i, instead of a unique mutation rate in the CGA. d) The new algorithm works on a unique solution and not on a number of search points, so the search performed by the mutation operator is local in scope. The global performance of the algorithm is performed by the random restart heuristic approach.

The effectiveness of this algorithm over that of the CGA was demonstrated experimentally. It was found that the new algorithm produces better results than the CGA.

References

Buford, D. S., Rakesh, S, 1986. Thermodynamic Data for Pure Compounds. Elsevier, Amsterdam.

Chen, G. L., Wang, X. F. et al, 1996. Genetic algorithm and its application. People's Post Press, Beijing (in Chinese).

Kitano, H., 1990. Empirical studies on the speed of convergence of neural network training using genetic algorithms, *In*: Proceedings AAAI-90, p. 789 - 795.

Montana, D. J., Davis, L., 1989. Training feedforward neural networks using genetic algorithms. *In*: Proceedings IJCAI - 89, p. 762 - 767.

Vargaftik, N. B., 1975. Tables on the Thermophysical Properties of Liquids and Gases. Hemisphere Publishing Corporation, Wiley, New York, p.99 - 175.

Xu, Z. B., Gao, Y., 1996. The characteristic analysis and prevention of premature convergence of genetic algorithms. *Science in China, Ser. E*, **26**(4): 364 - 375.

Byungjoo, Yoon, Holmes, Dawn J., et al., 1994. Efficient genetic algorithms for training layered feedforward neural networks. *Information Sciences*, **76**: 67 - 85.