Proceedings of the First International Conference on Machine Learning and Cybernetics, Beijing, 4-5 November 2002

BP NEURAL NETWORK OPTIMIZATION BASED ON AN IMPROVED GENETIC ALGORITHM

BO YANG , XIAO-HONG SU, YA-DONG WANG

School of Computer Science and Engineering, Harbin Institute of Technology, Harbin 150001, China
E-MAIL: boy@mlg.hit.edu.cn, sxh@mlg.hit.edu.cn

Abstract:

An improved Genetic Algorithm based on Evolutionarily Stable Strategy is proposed to optimize the initial weights of BP network in this paper. The improvement of GA lies in the introducing of a new mutation operator under control of a stable factor, which is found to be a very simple and effective searching operator. The experimental results in BP neural network optimization show that this algorithm can effectively avoid BP network converging to local optimum. It is found by comparison that the improved genetic algorithm can almost avoid the trap of local optimum and effectively improve the convergent speed.

Keywords:

Evolutionarily stable strategy; Genetic algorithm; Neural network; Back propagation (BP) algorithm; Premature convergence

1 Introduction

In recent years, there have been many attempts in designing artificial neural networks automatically, in which the combination of evolutionary algorithms and neural networks has attracted a great deal of attention and one kind of evolutionary artificial neural network has been formed. Evolving neural networks by genetic algorithm were researched earliest of all.

The efficiency of GA has great influences on BP neural network (BPNN) optimization. During application of GA, however, there often exists a problem of premature convergence and stagnation [1]. Whitley think that selective pressure and selection noise are the main factors of affecting population diversity [2]. Higher selective pressure often leads to the loss of diversity in the population, which causes premature convergence at the same time of improving convergent speed. Therefore, keeping the balance between population diversity and convergent speed is very important to the performance of GA.

In recent years, many diversity preservation methods have been developed to avoid premature convergence to a local optimum. These can be divided into the following three subclasses:

1)Schemes of alleviating selective pressure to keep the biologic diversity, such as the modification of selection operator [3-5] and scale-transformation of fit function [6]. Unfortunately, these methods often cause another problem of slow rate of convergence or stagnation in searching global optimum at the same time of improving population diversity.

2)Non-static mutation rate control schemes including dynamic [7-10], adaptive or self-adaptive [10-12] mechanism to control the rate of mutation. The mutation operator is a main operator to keep the biologic diversity, especially in real-coded GA, because it introduces new search space and maintain the genetic diversity of a population, whereas the crossover operator only operates in the known search space. From this point of view, high mutation rate is good for searching the global solution. But too high mutation rate will result in blind stochastic search. It has been proved that deterministically varying mutation rates during the search have a better performance compared to the fixed mutation rate schemes. Unfortunately, there are some drawbacks in non-static mutation rate control schemes. The dynamic parameter control scheme requires for the user to devise a schedule specifying the rate at which the parameter is typically decreased. The self-adaptive scheme does not need such a specific schedule. Unfortunately it is rather complicated to explain to novice users, and as a result they usually prefer the simple fixed mutation rate scheme.

3)Spatial separation schemes [13-14]. One of the most important representatives is the distributed GA's (DGA's). Their premise lies in partitioning the population into several subpopulations, each one of them being processed by a GA independently of the others. Furthermore, a migration mechanism produces a chromosome exchange between the subpopulations. In this way, a distributed search and an effective local tuning may be obtained simultaneously. They are suitable for producing multi-resolution in search space but run risk of running too much CPU time.

A genetic algorithm based on evolutionarily stable strategy (ESSGA) is proposed in this paper to try to pursue better balance between population diversity and convergent speed by means of introducing a new kind of mutation operator under the control of a stable factor. Different from other mutation rate control schemes, this mutation operator only acts on some of the preponderant individuals under the control of a stable factor, which keeps the ratio of quantity

0-7803-7508-4/02/$17.00 ©2002 IEEE

of preponderant individuals to stable quantity of total individuals in each evolution generation. It keeps population diversity by restricting the over-reproducing of preponderant individuals, and enlarges the search space as well. In this way, the selective pressure can be alleviated and the problem of premature convergence can be avoided without increasing any running time. The experimental results in "XOR" problem show that ESSGA has great improvement over the traditional one both in accuracy and in speed.

2 ESS Model

Evolutionarily stable strategy (ESS)[15] proposed by Smith is such a strategy that is used by most of the individuals in a colony, and the strategy is better than other strategies, which is called ESS. That is to say, the individual's behavior should obey the appointments of the colony. If one ESS has been determined, it will be stable for a long time; anything that departure from the appointments should be punished.

Assuming that there is a species called "hawk and dove" –the individual is either hawk or dove, and there are only two strategies—"hawk strategy" and "dove strategy". The rules are as follows: An individual will gain 50 points for winning a bout, gain noting for being beaten, lose 100 points for being wounded, and lose 10 points for delaying time. Moreover, in the fight between two hawks, one alternative is sure to be wounded. In the fight between two doves, however, there is no one being wounded but delaying time. Smith regards these rules as currencies for gene's survival. The higher average profit an individual gains, the more genes of the individual will remain.

Clearly, only the strategy submitting to ESS is good for evolving. Which one in the two strategies is an ESS? In a colony constructed only by doves, every individual gains 15 points averagely. The profit that the hawk individual gains can increase to 50 points which is higher than the former situation only when a hawk is introduced into the dove colony. The result in the hawk colony is similar to this. These results indicate that none of the colonies that are constructed only by one species can keep stable state in evolving process. From the rules above, we can estimate that the colony constructed by 1/6 hawk and 5/6 dove is the most probably successful group and its average benefit is $16\frac{2}{3}$ points.

3 GA Based On ESS

In traditional GAs, mutation is often independent of the state of present generation and it only changes part of the natures of its parents. As a result the offspring often retains most of the characteristics of its parents. In this way, the mutation is efficient only when the individuals are distributed nearby the optimum field. It is so easy to cause premature convergence especially in the case of small mutation rate. But higher mutation rate leads to blind stochastic search that will strongly affect the convergent speed. As discussed above, complex mutation rate control schemes can improve the performance of searching global optimum to some extent, but it is too complicated and difficult to understand for novice users and it costs too much CPU time. A better way to avoid the searching falling into the local optimum is to reap significant improvements by a simpler and easier tuning method and try to keep balance between performance and convergent speed.

Both empirical and theoretical evidences indicate that the optimal rate of mutation is not only different from every problem encoding but will vary with evolutionary time according to the state of the search. From this point of view, we propose a new mutation operator that is controlled by a stable factor. This kind of mutation operator is different from the original due to great changes in the offspring, which do not remain any characteristics of its parents. More unknown search space can be introduced by this kind of mutation operator. Obviously it is good for maintaining the population diversity. From view of ESS, if we regard the preponderant individual as hawk, the others as dove, and the rule as the characteristic information of the optimizing function, the result of the mutation is equivalent to adding a hawk to a dove colony or adding a dove to a hawk colony, which keep the colony on evolving. The stable factor setting is based on the ratio of the most probably successful group in ESSGA. The mutation operation is taken after detecting the state of the colony. After remaining a stable number of preponderant individuals, the rest preponderant individuals will be mutated to a random value within the scope of the parameter encoding, i.e. the mutation rate is varied according to the quantity of preponderant individuals. In this way, the selection pressure can be alleviated at the same time of maintaining population diversity.

Suppose that the encoding of the preponderant individual in a colony $V(v1,v2,.....,vk)$ is 01001, current state is S, and the stable factor is w, then every code bit of the preponderant individual determined by Detect(w, S, V) is mutated to a random value. For binary-coded GA, the offspring is produced by randomly mutating every bit of the individual. As a result, it has 2^n (n is the length of binary-coding) possible candidates for the mutation results: 00001, 00010, 00011, ... And for real-coded GA, the mutation result of the variables of the individual is equal to re-initializing this selected individual.

Suppose that the new generated colony is $U(u1, u2,....., uk)$, the preponderant individual is Best(01001, ui), $i \in [1,k]$, and the state is w, then the search space is spread to the global space from the adjacent space around the fitter individual. It plays an important role on quickly fleeing the local optimum and exploring the entire solution space where a traditional mutation cannot reach during the course of a run. This process is governed by the stable factor w, which ensures that the genes of the preponderant individual can still be reproduced to the next generation.

The differences between ESSGA and the traditional GAs

lie in that: the traditional GAs and its variations rely on the principle of "survival-of-the-fittest", whereas ESSGA limits the fittest individual to a finite number, which satisfies the rule of "ecological equilibrium". ESSGA is implemented as follows:

Step1: Generate an initial population of A_0 by stochastic selecting N individuals in the solution space and set generation number k=0. Then set the stable factor w, the annealing controlling parameter T_k, precision of the solution L, and crossover probability Pc, *etc.* In these parameters, the stable factor w is an important factor that can effectively adjust the selective pressure. It can be set smaller when object function has many different peaks, which is called hawk strategy. Otherwise, it can be set bigger, which is called dove strategy. The following experiential function can be used for reference.

$$w=\left\lceil \frac{N}{\partial} \right\rceil : N, \quad (\partial \geq 1)$$

where ∂ is a multi-peak coefficient, and ∂ =1 when the object function has only one peak.

Step2: Evaluate the fitness of each individual at the present generation.

Step3: Memorize the best chromosome.

Step4: Put some individuals selected according to roulette-wheel in matching pool M.

Step5: Detect the state of the present generation and get the stable factor S.

Step6: Mutate some preponderant individuals separated by the Detect(w, S, M) and generate new colony M_k.

Step7: Generate new colony B_k by operating crossover for each pair of parents with crossover probability Pc.

Step8: Offspring variables are mutated under control of the annealing parameter to get the next generation A_{k+1}.

Step9: Modify the annealing parameter T_k.

Step10: Repeat the previous 1-9 step until a stop condition is reached.

When ESSGA is adopted to learn the weights in BPNN, the combination algorithm can be described as follows:

Step1: Initialize N sets of weights and thresholds of BPNN stochastically and code them to N individuals.

Step2: Generate a best individual by using ESSGA. The fitness function is defined as follows:

$$f = 1/\sum_{k=1}^{m}\sum_{j=1}^{q}(y_j^k - y_j)^2$$

where m is the sample numbers used for training, q is the neuron numbers of output layers, y_j^k is the desired output of the jth neuron, and y_j is the computed output of the jth neuron.

Step3: Decode the best individual to the weights and thresholds of a BPNN, which are used as its initial weights and thresholds.

Step4: Training the BPNN for fixed iterations or until the desired error is reached.

4 Simulation Results

The performance of ESSGA with the above-described mutation operator is tested on the "XOR" problem in the rate of convergence, accuracy and CPU running times. The simulation is run on machine Duron 800/256M.

In the experiment, we use the following parameter settings: real-coded, the coding length is 9, the population size is 50, and the maximum iteration number is 100. The crossover rate Pc is 0.5, the Gauss mutation rate Pm is 0.05, the stable factor is set as 15:50, and the annealing control parameter is T_k=log (T_0/k+1), T0=100.The training iteration of BPNN with 2-2-1 three-layer architecture and sigmoid function is 50000.

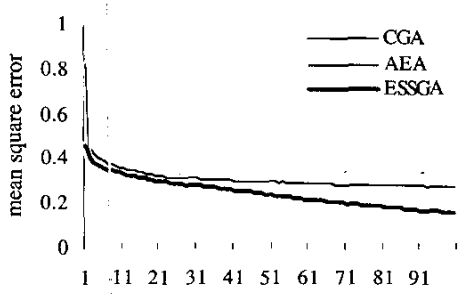

Fig.1. generation vs error within [-5,5]

For three methods including Canonical Genetic Algorithm (CGA) with elite-preservation strategy, Annealing Evolution Algorithm (AEA), and Genetic Algorithm based on Evolutionarily Stable Strategy (ESSGA), the average results taken over 100 independent trials are plotted in Fig.1. The simulation results indicate that ESSGA with the above-described mutation operator is well behaved in improving the rate of convergence and the capacity of searching global optimum compared to the other two methods. The specific comparisons in performances among the three algorithms are shown in Table 1.

Table 1.Comparisons of performances with initial weights and thresholds ranging within [-5,5]

	CGA	AEA	ESSGA
Number of local optimum	48	46	5
Average mean square error before training BP	2.80345 E-01	2.58876 E-01	1.56532 E-01
Average mean square error after training BP	13.2728 E-02	11.9341 E-02	1.25970 E-02
Average final error for successful optimum	6.54189 E-05	6.0557 E-05	6.30329 E-05
Fault rate (%)	48	46	5

The results shown in Table 1 imply that the performance of ESSGA is much better than that of CGA and AEA. The mutation operator under control of stable factor plays an important role in avoiding premature convergence and improves the reliability and stability of the algorithm.

Table 2 shows the comparisons of the three algorithms in average convergent iterations and CPU running time to reach the desired global error 1.0E-04 over 100 runs. The training iteration is 500000.

Table 2. Comparisons of performances for reaching desired global error 1.0E-04

	BP	CGABP	AEABP	ESSBP
Iteration	302350	291162	242366	74144
Running time(sec.)	6.05	6.29	5.49	1.99

The results in Table 2 show that BPNN combined with ESSGA costs less time in reaching the desired global error 1.0E-04 than the other two methods.

A very interesting thing is that the error changes lower, as the range of weights and thresholds of BPNN is wider. But the ability of BPNN training changes weaker as the range of parameters is wider. When the range of weights and thresholds of BPNN is enlarged from [-5,5] to [-50,50], the average results of the three methods over 100 runs are shown in Fig.2 and Table.3.

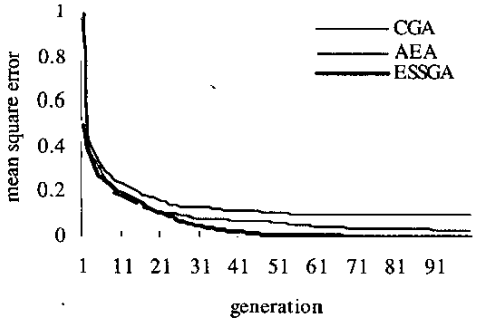

Fig.2.generation vs error within [-50,50]

Table 3. Comparisons of performances with initial weights and thresholds ranging within [-50,50]

	CGA	AEA	ESSGA
Average final error after training BP	7.4276 e -02	5.8420 e-02	2.3622 e-06
Fault rate (%)	28	21	0

The results show that none of the 100 runs is premature convergence to a local optimum in ESSGA. So the BPNN evolved by ESSGA is more robust and stable than the others. This implies that ESSGA is more powerful and effective in avoiding the trap of local optimum within a large problem space than CGA and AEA.

5 Conclusions

Incorporating the ideas of evolutionary stable strategy and ecological equilibrium into the genetic algorithm, a new genetic algorithm based on evolutionary stable strategy is proposed in this paper. The new mutation operator governed by stable factor is found to be a very powerful searching operator. Moreover, it is very simple and easy to be implemented. As demonstrated in the experiments about the BPNN optimization problems, ESSGA is superior to CGA and AEA in the rate of convergence, accuracy, reliability and stability. The new mutation operator controlled by stable factor is efficient in preserving population diversity and so it can overcome the shortcomings of premature convergence and stagnation.

Of course, there is only empirical but no theoretical guidance about how to select different stable factors for different optimization problems appropriately. This is a future research work.

References

[1] Goldberg D.E., Genetic Algorithm in Search, Optimization & Machine Learning, Reading, M A: Addison-Wesley, 1992.

[2] Whitley, D., The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation Reproduction Trials is Best, in Proc. of the 3rd International Conference on Genetic Algorithm, 1989.

[3] De Jong, K. A., An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Doctoral dissertation, University of Michigan, 1975.

[4] Baker, J. E., Adaptive Selection Methods for Genetic Algorithms, in Proc. of the 1st International Conference on Genetic Algorithms, 1985:101-111.

[5] Goldberg, D.E., Deb,K. and Korb, B., Do not Worry, Be Messy, in Proc. of the 4th International Conference on Genetic Algorithms, 1991:24-30.

[6] Goldberg.D. Genetic Algorithms in Search, Optimization and Machine Learning. USA, Addison-Wesley Publishing Company. 1988,7-10:59-308.

[7] Fogarty T., Varying the Probability of Mutation in the Genetic Algorithm, Proc. Of the 3rd International Conference on Genetic Algorithms, 1989:104-109.

[8] Hesser J. and Manner R., Towards an Optimal Mutation Probability in Genetic Algorithms, Proc. Of the 1st Parallel Problem Solving from Nature. Springer, 1991:23-32.

[9] Hesser J. and Manner R., Investigation of the m-heuristic for optimal mutation probabilities, Proc. Of the 2nd Parallel Problem Solving from Nature. Elsevier, 1992:115-124.

[10] Back T., and Schutz M., Intelligent Mutation Rate Control in Canonical Genetic Algorithms, Proc. Of the International Symposium on Methodologies for Intelligent Systems, Springer, 1996:158-167.

[11] J.Smith, and T.C.Fogarty, Self Adaptation of Mutation Rates in a Steady State Genetic Algorithm, Proc. IEEE Int'l Conf. on Evolutionary Computation, 1996:318-326.

[12] Srinivas, M., Patnaik, L.M. Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Transactions on Systems, Man, & Cybernetics, 1994,24(4): 656-665.

[13] H. Muhlenbein, M. Schomisch, and J. Born, The parallel genetic algorithm as function optimizer, in 4th in. Conf. Genetic Algorithms, R.Belew and L.B.Booker, Eds. San Matero, CA: Morgan Kaufmann, 1991:271-278.

[14] Fancisco Herrera and Manuel Lozano, Gradual Distributed Real-Coded Genetic Algorithms, IEEE Transactions on evolutionary computation, 2000, 4(1): 43-62.

[15] Dawkins, Richard. THE SELFISH GENE, Oxford University Press Reprinte d 1977.