Artif Intell Rev (2011) 36:153–162
DOI 10.1007/s10462-011-9208-z

An optimizing BP neural network algorithm based on genetic algorithm

Shifei Ding · Chunyang Su · Junzhao Yu

Published online: 18 February 2011

Abstract A back-propagation (BP) neural network has good self-learning, self-adapting and generalization ability, but it may easily get stuck in a local minimum, and has a poor rate of convergence. Therefore, a method to optimize a BP algorithm based on a genetic algorithm (GA) is proposed to speed the training of BP, and to overcome BP's disadvantage of being easily stuck in a local minimum. The UCI data set is used here for experimental analysis and the experimental result shows that, compared with the BP algorithm and a method that only uses GA to learn the connection weights, our method that combines GA and BP to train the neural network works better; is less easily stuck in a local minimum; the trained network has a better generalization ability; and it has a good stabilization performance.

Keywords Genetic algorithm (GA) · BP neural network · Connection weight · UCI data

1 Introduction

A back-propagation (BP) neural network is one of the most maturely studied neural network algorithms; it has good self-learning, self-adapting, robustness and generalization ability. A three-layered BP neural network can approach any non-linear functions with any precision. The BP neural network has been widely applied in many fields, such as pattern recognition, function approximation and image processing etc.

However, the BP algorithm has some disadvantages, such as poor rate of convergence, and getting stuck in local minimum easily. Furthermore, for BP algorithm is based on the gradient information of error function, when the problems are complex or the gradient infor-

S. Ding (✉) · C. Su · J. Yu
School of Computer Science and Technology, China University of Mining and Technology, 221008 Xuzhou, China
e-mail: dingsf@cumt.edu.cn

S. Ding
Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, 100080 Beijing, China

mation is hard to get, BP may be helpless. To overcome the disadvantages, many optimization algorithms have been introduced in the study and design of neural networks such as constructing a neural network based on particle swarm optimization algorithm (Chen and Yu 2005), and using evolutionary algorithms to optimize the neural networks (Eysa and Saeed 2005; Harpham 2004; Venkatesan 2009; Yao and Islam 2008), which have been proved feasible and effective.

The Genetic algorithm (GA), one of the evolutionary algorithms, is a heuristic stochastic search algorithm. For GA has good global searching ability and can learn the near-optimum solution without the gradient information of error functions, it has been a powerful tool of optimization, searching and machine learning (Yao 2004). The combination of evolutionary algorithms and artificial neural networks is attracting much greater attention, and a field called evolutionary neural networks is formed (Yao 1999). In recent years, since the research in this field is very active, a lot of valuable conclusions and results have been obtained, and some of them have been successfully applied.

In this paper, the BP algorithm is optimized based on GA, that is, to optimize the connection weights of the neural networks with GA, so as to overcome the disadvantage of getting stuck in local minimum easily; And the connection weights by combining GA and BP are studied, in order to increase BP's convergence rate and improve the generalization ability of neural networks. The parameters of GA, the selection mechanisms, fitness scaling, genetic operators, and the time to change from GA to BP are researched in the paper. So are the generalization ability and training efficiency of the BP neural network, the GA network and the GABP network. The experiment result shows that the generalization ability and training efficiency of the GABP neural network are better than the other two in some classification problems.

2 GABP neural networks

2.1 A brief introduction of BP

The BP neural network will be briefly introduced here. The BP neural network is simply a gradient descent method designed to minimize the total error (or mean error) of the output computed by the network. Figure 1 shows such a network. In this network, there is an input layer, an output layer, and one or more hidden layers between them.

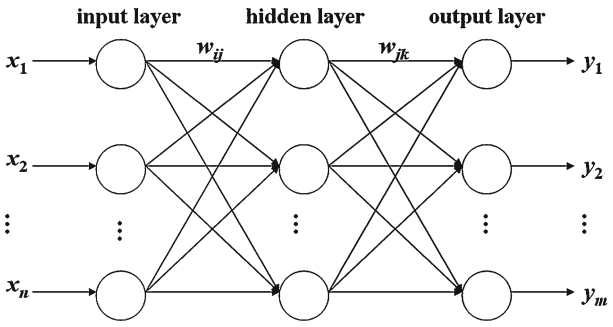

Fig. 1 The topology of BP neural network

Suppose there are n inputs and m outputs in the network, and s neurons in the hidden layer, the output of the hidden layer is b_j, the threshold value of the hidden layer is θ_j, the threshold value of the output layer is θ_k, the transfer function of the hidden layer is f_1, the transfer function of the output layer is f_2, the weight from input layer to hidden layer is w_{ij}, the weight from hidden layer to output layer is w_{jk}. Then we can get the output of the networky_k, the desired output is t_k, and the output of jth neuron of the hidden layer is:

$$b_j = f_1\left(\sum_{i=1}^{n} w_{ij}x_i - \theta_j\right) \quad (i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, s)$$

Calculating the output y_k of the output layer, this is:

$$y_k = f_2\left(\sum_{j=1}^{s} w_{jk}b_j - \theta_k\right) \quad (j = 1, 2, \ldots, s;\ k = 1, 2, \ldots, m)$$

Defining the error function by the network actual output, that is:

$$e = \sum_{k=1}^{m} (t_k - y_k)^2$$

The network training is a process of continual readjustment between the weights and the threshold, in order to make the network error reduce to a pre-set minimum or stop at a pre-set training step. Then input the forecasting samples to the trained network, and obtain the forecasting results.

2.2 A brief introduction of GA

GA is one of evolutionary algorithms, and the characteristics of evolutionary algorithms are population-based evolution, survival of the fittest, directed stochastic, no dependence on the gradient information.

GA is an iterative computation process, and its main steps include: encoding, initialization of the population, selection, genetic operation (crossover, mutation), evaluation and stop decision (Yao and Xu 2006).

Comparing with the traditional optimization algorithms, the main characteristics of GA are:

1. Population-based searching strategy. The traditional optimization algorithms use the point to point searching, while GA uses population to population searching, so GA is easier to reach global optimum.
2. The search does not relay on the gradient information of objective function, only the fitness that can evaluate the quality of individuals are needed. So, GA has a wider application, especially is fit for the problems that are complex and nonlinear.
3. The evolution process is not blind, but heuristic.
4. It is simple, general-purpose, and robust.

2.3 Optimize the neural networks' weights based on GA

The traditional network weight training generally uses gradient descent method; these algorithms get stuck to local optimum easily and can not get the global optimum. The evolution of connection weights is to introduce a self-adapted and global method to train. One aspect

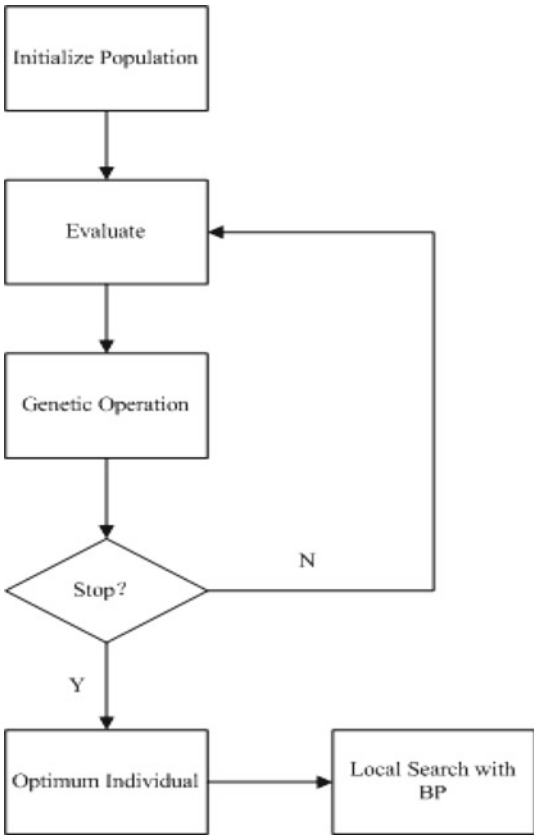

Fig. 2 The flow chart of using GA to optimize BP algorithm

of using GA in neural networks is to use GA to learn the connection weights, in this way can we replace some traditional learning algorithms to overcome their defects.

The typical algorithm that uses GA to optimize the connection weights of BP neural networks is as follows and the flow chart of the algorithm is as the Fig. 2 shows:

1. RandomGenerate(P[0]);
2. pANN = new ANN;
3. Set ***iGeneration*** = 1;
4. REPEAT
5. For ***i*** = 1, 2, 3,…, Size
6. pANN→ SetANNWeight(P[***iGeneration***]);
7. Fitness[***i***] = C - pANN→ComputeError();
8. NewP = Select, Crossover, Mutate;
9. Set ***iGeneration*** = ***iGeneration***+1;
10. P[***iGeneration***] = NewP;
11. UNTIL halting criteria are satisfied;
12. Local search with the BP algorithm.

The key problems are as follows:

(1) Coding Strategy:

There are two ways to encode the connection weights and the threshold in the neural network. One is binary encoding, the other is real encoding. Binary encoding is that each weight is expressed by fixed length 0, 1 string. Real encoding is to express each weight with a real number; it overcomes the abuse of binary encoding, but it need to re-design the operators, such as the crossover and mutate operators.

(2) Determine the fitness function

Using GA to evolve the weights of neural networks, if the network architecture is fixed, generally, the network with big error, the fitness is small. For example, the fitness function may be: $F = C - E$. Here F is the fitness of the individual, C is a big number, and E is the training error of the network.

(3) Evolution process

Determine the global search operator of the algorithm, such as selection, crossover and mutation operator, also can design special operator.

(4) Train the network

Because GA is good at searching large-scale, complex, non-differentiable and multimodal spaces; it doesn't need the gradient information of the error function. On the other hand, it doesn't need to consider whether the error function is differentiable, so some punishment may be added to the error function, so as to improve the network commonality, and reduce the complexity of the network.

The results of GA and a BP algorithm are both sensitive to the parameters. The result of a BP algorithm also depends on the original state of the network, but a BP algorithm appears to be more effective when used in local search, GA is good at global search. On the fact of this, connection weights' evolution algorithm can be implemented in the following way. First, Use GA to optimize the initial weight distribution and locate some better search spaces in the solution space. Then use a BP algorithm to search the optimal solution in these small solution spaces. Generally, the efficiency of the hybrid training is superior to the training methods that only uses a BP evolution or only use a BP training (Meng 2000). As for when to change GA to a BP algorithm, in fact this is just like the design of the neural networks, the specific data set and the specific parameter settings should be considered.

This paper only discussed using GA to optimize the weights of a BP neural network, GA also can be used to find the network structure and learning parameters (Gupta and Sexton 1999; Ghosh and Verma 2003). Most of times, a BP neural network is built with one hidden layer, the network's performance is decided by the network structure (that is, the hidden neurons), the weights, the learning iteration, etc.

3 Experiments and results

Use Iris in the UCI data set (UCI Maching Learning Repository) to do experiments. The four used datasets are as Table 1 shows. The numbers of the samples, attributes and classes of each dataset are showed.

Experiments preparation:

1. Normalize the original data.
2. GA used real encoding. The length of the chromo is the total number of connection of the neural networks, the parameter settings are as follows:

Table 1 The general situation of used datasets

Dataset	Samples	Attributes	Classes
Iris	150	4	3
Wine	178	13	3
Vehicle	846	18	4
Glass	214	9	6

Population size: 30
Crossover rate: 0.9
Mutation rate: 0.01
Crossover operator: Single-point crossover
Mutation operator: Gaussian mutation
Selection mechanism: Roulette wheel selection
Use the elitism strategy (reserve the 4 fittest individuals to the next generation).
Use linear fitness scaling.

(3) The neural network has only one hidden layer which has ten neurons, the transfer functions of the hidden layer and the output layer both are sigmoid functions. The learning rate of the BP algorithm is 0.1; the BP algorithm is without any improvement.

Experiment 1:

Use the whole data set to train the neural network, the training error-epoch graphs are as follow:

In the three figures, the error(Y axis) is squared error, and that is $MSE = \sqrt{\sum_{i=1}^{N} E_i^2/N}$. One point to be noted is that the three figures above are only three certain training processes (Figs. 3, 4). For the BP neural networks, when the original weights are different, the training epochs will be quite different, for the GA neural networks, even the original weights are the same, the training epochs will still be different. The weight adjustment of BP's each iteration is determinate while GA's is uncertain (for the crossover and mutation operators are uncertain processes). For Fig. 5, the neural network is firstly trained by GA, the by BP, that is, first use GA to make the error descend to 0.01, then use BP to continue the training, until the error falling to 0.001. In addition, the iteration's cost (consuming time) of GA and BP are different (determined by the GA's parameter settings and the specific data set); here the consuming time of GA's one iteration is about 4 times of the BP's.

Experiment 2:

Randomly divide the data set to training set and testing set by the ratio of 4:1, to test the generalization ability of each algorithm.

Experiment method:

In each time training (no matter which algorithm), the data set should be divided randomly again. Record the iterations, classification precision and running time. Each algorithm will be run 25 times, and compute the average values.

The parameter settings of each algorithm are as above, for GABP, firstly the GA will be run 300 generations, and then BP is used. This can be seen as using GA to find a better weights space, then searching in this space with BP; also can be seen as use GA to initialize the connection weights.

First use BP to train; BP's defect which gets stuck in local minimum easily is exposed. In the continuous 25 times testing, 16 times are failed, which means to get stuck in local minimum (Tables 2, 3 and 4).

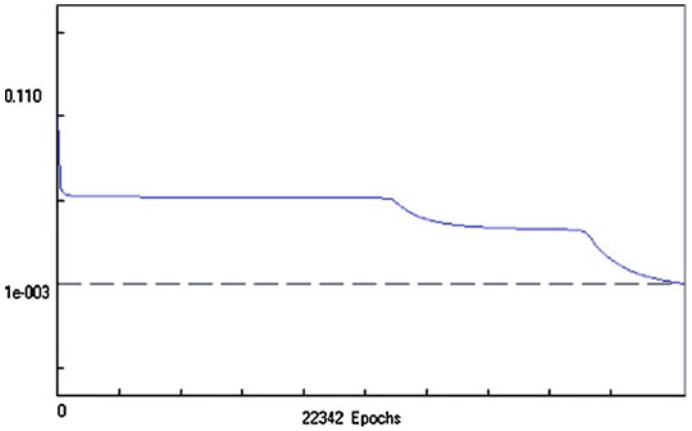

Fig. 3 The error-epoch graph of BP neural network

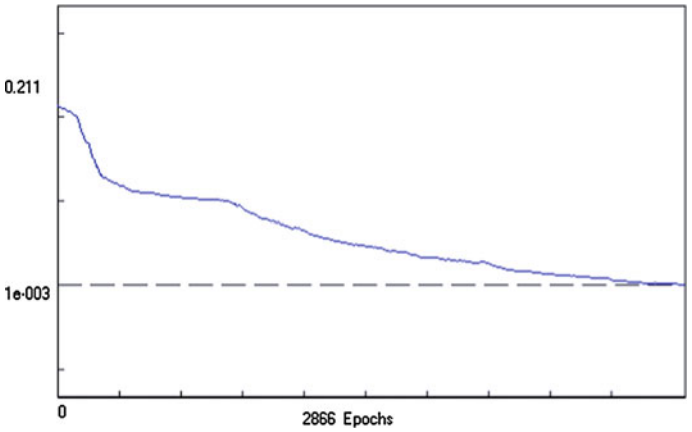

Fig. 4 The error-epoch graph of GA neural network

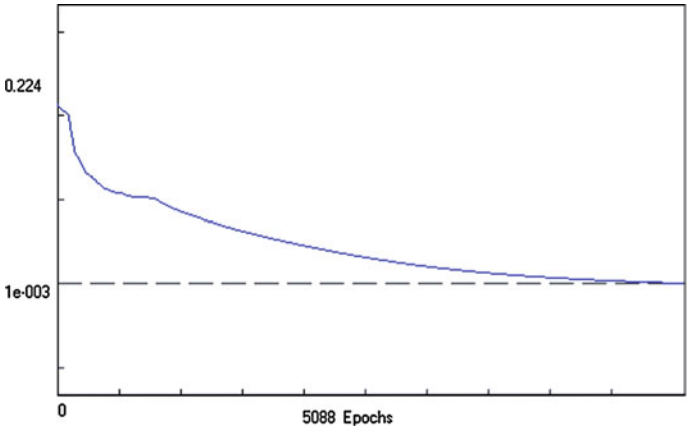

Fig. 5 The error-epoch graph of GABP neural network

Table 2 The training results of BP neural network

Iterations	Precision	Time (s)
9454	0.97	20
3004	0.9	6
10299	0.9	22
2118	0.9	5
3451	0.93	7
8544	0.93	18
9102	0.97	19
8898	0.97	19
4323	0.9	9

Table 3 The training results of GA neural network

Iterations	Precision	Time (s)
2578	0.83	22
813	0.9	7
2135	0.87	18
2642	0.93	22
2979	0.93	26
789	0.87	7
1750	0.9	15
733	0.93	6
1321	0.9	12
1730	0.9	15

Table 4 The training results of GABP neural network

Iterations	Precision	Time (s)
7754	0.97	18
2372	0.97	7
7658	0.97	19
4418	0.93	12
9295	0.97	22
4303	0.97	11
5240	0.97	14
4492	0.9	12
4420	0.97	11
3729	0.97	10

The training results of the 9 times that do not get stuck in local minimum are as follow:

Then use GA to train, in the continuous 25 times testing; only 1 time is failed.

A part of the training results are as follow:

The use GABP to train, in the continuous 25 times testing; 2 time are failed.

A part of the training results are as follow (the iterations in the following table are the total iterations of GA and BP, GA's iterations is 300, the time is the total time of GA and BP):

Table 5 The comparison of the three algorithms

Algorithm	Iterations	Precision	Time (s)	Successful
BP	6577	0.93	13.9	9
GA	2067	0.92	17.5	24
GABP	6819	0.95	16.6	23

Table 6 Results of different data sets with different algorithms (testing precision)

Algorithm	Wine data sets	Vehicle data sets	Glass data sets
BP	0.95	0.72	0.63
GA	0.96	–	–
GABP	0.97	0.74	0.6

The values in Table 5 are the average values. It is observed that the BP is relatively fast when it does not get stuck in local minimum. But the problem is that a BP algorithm is easy to get stuck in local minimum. GA and GABP algorithm, these two both are hard to get stuck in local minimum, but the latter is more effective than the former and has a better generalization ability. It should be noted that comparing the iterations of GA and BP is meaningless, for the two are so different, so here the running time is listed. The generalization ability of GABP is better, this may because GA is good at local searching, and the weight adjustment is exquisite.

Then we use other three data sets (Wine, Vehicle and Glass) to test the algorithms' performance. The neural networks are built with one hidden layer and eight hidden neurons. Different parameters are used for the three data sets and each experiment is done for 10 times and the results are the average values. We find that when processing Vehicle and Glass data sets, GA is hard to converge, so the results are ignored.

The results show that GABP is effective for some data sets, but is not suitable for some other data sets. When data sets are complex, GA is so slow and hard to process them; it can only be treated as a pre-search technique, that is, to find a better search space. However, GABP is not always valid; its parameters are also hard to decide (Table 6).

4 Conclusions

This paper combines the BP neural networks and GA and show the method that use GA to optimize the connection weights of neural networks. The experiments show the effectiveness of the algorithms; this is determined by the high robustness and effectiveness of GA. Of course the GA's weakness is obvious, that is, although it is globally convergent but not suitable for the tune of the candidate solution. A BP algorithm has better local searching ability. When using GA to globally optimize the network to some extent (that has no theory to guide, may be the GA can stop after a certain generation or the error is reduced to a pre-defined lower limit, this may be tried several times), then use BP to learn. This will improve the convergence rate of the network and reduce the training failure, and the neural network's generalization ability is better than the algorithms that only use GA.

Acknowledgments This work is supported by the Basic Research Program (Natural Science Foundation) of Jiangsu Province of China under Grant No. BK2009093, and the National Nature Science Foundation of China Grant No. 60975039.

References

Chen GC, Yu JS (2005) Particle swarm optimization neural network and its application in soft-sensing modeling [J]. Lecture Notes Comput Sci 3611:610–617

Eysa S, Saeed G (2005) Optimum design of structures by an improved genetic algorithm using neural networks [J]. Adv Eng Softw 36(11–12):757–767

Ghosh R, Verma B (2003) A hierarchical method for finding optimal architecture and weights using evolutionary least square based learning [J]. Int J Neural Syst 13(1):13–24

Gupta JND, Sexton RS (1999) Comparing backpropagation with a genetic algorithm for neural network training [J]. Omega 27(6):679–684

Harpham C et al (2004) A review of genetic algorithms applied to training radial basis function networks [J]. Neural Comput Appl 13(3):193–201

Meng XP et al (2000) A hybrid method of GA and BP for short-term economic dispatch of hydrothermal power systems [J]. Math Comput Simul 51(3–4):341–348

UCI Maching Learning Repository. http://archive.ics.uci.edu/ml

Venkatesan D et al (2009) A genetic algorithm-based artificial neural network model for the optimization of machining processes [J]. Neural Comput Appl 18(2):135–140

Yao X, Islam MM (2008) Evolving artificial neural network ensembles [J]. IEEE Comput Intell Mag 3(1):31–42

Yao WS et al (2004) The researching overview of evolutionary neural networks. Comput Sci 31(3):125–129

Yao X (1999) Evolving artificial neural networks [J]. Proc IEEE 87(9):1423–1447

Yao X, Xu Y (2006) Recent advances in evolutionary computation [J]. J Comput Sci Technol 21(1):1–18