A Modified Differential Evolution Algorithm for Optimization Neural Network

Guiying Ning   Yongquan Zhou

College of Mathematics and Computer Science, Guangxi University for Nationalities, Nanning 530006, P.R.China

Abstract

A Modified Differential Evolution (MDE) is proposed, which is based on the basic Differential Evolution (DE) algorithm principle and implementing framework of DE. Optimizing the initial individuals with the 1/2 rule, then by introducing the reorganization of Evolution Strategies during the period of mutation procedures. The MDE is used to optimize the weights of the feed-forward multilayer neural network, and compared with the basic DE and BP algorithm with momentum term. Finally, the numerical simulation results show that this method has good quality of high-speed global convergence and effectively improves the precision and convergence speed for feed-forward multilayer neural network. It has been proved the effectiveness and feasibility.

Keywords: Evolution strategies, Neural network, 1/2 rule, Evolution algorithm, Differential evolution.

1. Introduction

Differential Evolution (DE) algorithm is an evolutionary algorithm, which was proposed by Rainer Storn in 1995 [1]. DE algorithm has no requirements to the convex of the function, differential calculus and consecution etc. In the aspect of resolving complex global optimization problems. DE algorithm can search the global optional solution effectively [2].In recent years, DE algorithm is beginning to prevail in the fields of international evolutionary algorithm. Compared with Particle Swarm Optimization (PSO) and Genetic Algorithm (GA), DE algorithm has many advantages, such as faster convergence speed, stronger stability, easy to realize [3] and so on, so it is noticed by many researchers. But the individuals of the basic DE algorithm is random during the period of evolutionary procedures and it is prone to cause unsteady easily.

In this paper, basic DE algorithm was carried on some modification: at first, computing the fitness value of individuals, then using 1/2 rule to select individuals, finally carrying on mutation of DE and reorganization of Evolution Strategies respectively to born new groups, the new groups were used to conduct the crossover and selection of DE. Moreover self-adaptive mutation was introduced in order to improve the capability of DE further. The good global optimum ability of the Modified Differential Evolution (MDE) was used to train the weights and biases of three-layer feed-forward neural network; it can overcome the shortage of BP algorithm and can help to find out the optimal solution globally over a domain. The results of experiments show that MDE algorithm is superior to the basic DE and momentum BP algorithm.

2. Basic DE Algorithm

Basic DE algorithm is a kind of evolutionary algorithm, which is used to optimize the minima of functions, and its code is based on real number, the whole structure is similar to the GA**,** and the main difference between standard GA and DE is mutation operation. The mutation is a main operation of DE, and it revises each individual's value according to the difference vector of the population. Its basic idea lies in applying the difference of current population individual to reorganize to obtain the middle population, then, using the direct offspring and parents individual fitness value competition to get the new generation [4]. The process of standard DE algorithm can be described as follows:

Step 1. Initialization.

Assume population size M , the rate of crossover p_c , selection the initial population randomly:

$$X(0) = (X_1(0), X_2(0), ..., X_M(0))$$

Set $t = 0, X_i(0)$ is n-dimensional vectors.

Step 2. Population evolution. Let each individual $X_i(t)$ of $X(t)$ to operate as follows:

1) *Mutation.* Mutation is the main operation of DE.

Let

$$u_{ij}(t+1) = x_{bestj}(t) + \eta(x_{p1j}(t) - x_{p2j}(t))$$

where $x_{p1j} - x_{p2j}$ is the difference vectors, and the indices $i, p1$ and $p2$ are distinct from $1,2,...,M$. x_{bestj} is the best individual of the t-th generation, η is mutation factor, $i = 1,2,...,M$; $j = 1,2,...,n$.

2) *Crossover.* The trail vector v_i is obtained through the following formula:

$$v_{ij}(t+1) = \begin{cases} u_{ij}(t+1) & randl_{ij} \le p_c \ or \ j = rand(i) \\ x_{ij}(t) & randl_{ij} > p_c \ or \ j \ne rand(i) \end{cases}$$

where p_c is the rate of crossover and $randl_{ij}$ is the random fraction in the range of [0, 1], $rand(i)$ is the random integer in the range of [1, n]. This method can ensure that $X_i(t)$ has one corresponding element at least in the $X_i(t+1)$.

3) *Selection.* Comparing with the fitness value of $V_i(t+1)$ and $X_i(t)$, the new individuals can be obtained through the following formula:

$$X_i(t+1) = \begin{cases} V_i(t+1) & if \ J(V_i(t+1)) < J(X_i(t)) \\ X_i(t) & otherwise \end{cases}$$

Step 3. Judgment on conditions.

The new population coming from step 2 can be defined as follow:

$$X(t+1) = (X_1(t+1), X_2(t+1), \ldots X_M(t+1))$$

The best individual of $X(t+1)$ is denoted as $X_{best}(t+1)$.If the given accuracy or the maximum iterative number is satisfied, the algorithm operation ceases and output is result, otherwise set $t = t+1$,and then shifts to step 2.

3. A Modified Differential Evolution Algorithm (MDE)

According to the basic DE characteristics, in this paper, basic DE was modified as follow:

1) Initialize the M individuals randomly, generally, M is no less than 20.

2) Calculation of fitness. In each generation, the best individual was kept to attend mutation operation.

3) Choose the better 50% individuals (the fitness smaller individuals) to attend population evolution:

① *Mutation.*

Conducting reorganization of the Evolution Strategy and mutation of DE, respectively, it should be born M individuals totally. The calculation is conducted according to the following formulas:

$$u_{ij}(t+1) = x_{bestj}(t) + \eta(x_{p1j}(t) - x_{p2j}(t))$$
$$A_{ij}(t+1) = (x_{p3j}(t) + x_{p4j}(t))/2$$

Where the indices $i, p1, p2, p3$ and $p4$ are distinct from $1,2,...M/2$. The new M individuals are denoted as L.

② *Crossover.*

Crossover operation is conducted as follow:

$$v_{ij}(t+1) = \begin{cases} l_{ij}(t+1) & randl_{ij} \le p_c \ or \ j = rand(i) \\ x_{ij}(t) & randl_{ij} > p_c \ or \ j \ne rand(i) \end{cases}$$

Where p_c is the rate of crossover and $p_c \in$ (0, 1], $randl_{ij}$ is the random fraction in the range of [0, 1], $rand(i)$ is the random integer in the range of [1, n], $i = 1,2,...,M$, $j = 1,2,...,n$.

③ *Selection.*

According to their fitness value, $M/2$ individuals were selected from the offspring and parents. Mutation, crossover and selection continue until some stopping condition is satisfied.

During the searching, the rate of mutation of DE algorithm is usually a fixed real number, which was selected from 0 to 2.And the rate of mutation is hard to decide in the concrete implement. If the rate of mutation is too big, the best solution is easy to be broken in the searching process and the rate of finding global solution is low. If the rate of mutation is too small, the population diversity is low, it is easy to trap into local minimum point and cause to algorithm precocious. The numerical experiment show that mutation factor should be a little bigger at start, with the increasing of iterative number, the distance of individual is more and more small, mutation factor at this time should be a little smaller in order to ensure its convergence. So in this paper, self-adaptive mutation is put forward. Which can adjust the rate of mutation by self-adaptive during the searching process. This self-adaptive mutation can make the algorithm has bigger rate of mutation to keep the diversity of the population at beginning , avoid precocious; and reduce the rate of mutation gradually at later stage, reserve a good information, avoid the best solution to be broken and increase the rate of searching the global solution. Mutation factor was modified as follow:

$$\eta = \beta(e^{\lambda} - 1)\,.$$

where $\beta \in$ [0.2, 0.6], $\lambda = \frac{G_{\max}}{G_{\max} + G}$, $G_{\max}$ is the maximum iterative number, G is the current iterative number.

4. A MDE for Optimization Neural Network

The neural network is a large-scale self-organization and self-adaptation nonlinear dynamic system. Artificial neural network technology is an effective way to solve complex nonlinear mapping problem. In numerous neural network models, feed-forward multilayer neural network model is one of the most widely used models in current, there is the research show that three-layer feed-forward neural network can with arbitrary accuracy approximate any continuous function and its each order derivatives [5]. Back propagation (BP) learning can realize the training of feed-forward multilayer neural network. The algorithm mainly revises neural network weights according to the gradient descent methods to reduce error. This kind of method calculates simply. But there are still many drawbacks if neural network are used alone, for example, low training speed, easy to trap into local minimum point, and poor global searching ability, and so on. Though many improvements have already been carried on in this aspect, such as introducing momentum parameter, but it can't solve problem by the root. The MDE in this paper can overcome the barriers of BP algorithm.

When training the neural network with MDE, the output error is regarded as the fitness function. Learning is carried on to calculate the fitness value defined by the following equation:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{T}(d_{j,i} - y_{j,i})^2$$

Where E is the value of the fitness function, N is the number of training samples, $d_{j,i}$ is the ideal output value of the i-th sample in the j-th output node, $y_{j,i}$ is the real output of the i-th sample in the output node, T is the number of network output neurons. The function is a nonlinear function with many minimum points. The process of training the neural network is to adjust each of the neuron's weights and bias, until the output error arrives to the minimum. Realization processes of MDE algorithm training feed-forward neural network are shown as follows:

Step 1. Given the topological structure and the activation function.

Step 2. Given the number of the weights and the biases (in this paper, biases are regarded as the special weights), the target function expression E, the control parameter η, p_c, the input, ideal output and stopping condition.

Step3. Production initial population W randomly, where $W \in$ [-1, 1].

Step4. Computation target function value, choosing the better 50% individuals (the fitness smaller individuals) to attend population evolution, keeping the smallest value of E and its corresponding vectors.

Step 5. Mutation of MDE.

Step 6. Crossover of MDE.

Step 7. Selection of MDE.

Steps 4 to *Step 7* continue until some stopping condition is reached.

5. Experimental Results

In order to test the capability of MDE training the feed-forward neural network, three typical functions: e^x, $\sin(x)$ and $\cos(x)$ were tested by *MATLAB 7.0*. The three functions all adopt three layers feed-forward neural network, the weights from input-layer to middle-layer all are fixed and equal to 1, the activation function of each middle-layer unit is denoted as follow respectively:

$$\phi_1(x) = 1, \phi_2(x) = x, \phi_{i+1}(x) = 2 \times x \times \phi_i(x) - \phi_{i-1}(x)$$

Where $i = 1,2,...,n.$, the output-layer is the identity function, that is $\phi(x) = x$, population size M is 30, β is 0.5, p_c and is the random number in the range of (0, 1], momentum parameter is 0.8. The test network topological structure of e^x is 1-8-1, $\sin(x)$ and $\cos(x)$ are 1-10-1. The basic model of them is shown in Figure1. At the same time, in order to improve the convergence speed, the input data are regularized [6]. In this paper, each set of input data is mapped to [-1.0, 1.0],

$$x' = -1.0 + \frac{2(x - x_{\min})}{(x_{\max} - x_{\min})},$$

Where $x_{\max}$ and $x_{\min}$ are the maximum and the minimum of the inputs, respectively, x is the number before being regularized and x' is the number after being regularized. The results are shown from Table 1 to Table 3; the evolutional curves are shown in Figure 2 and Figure 3.

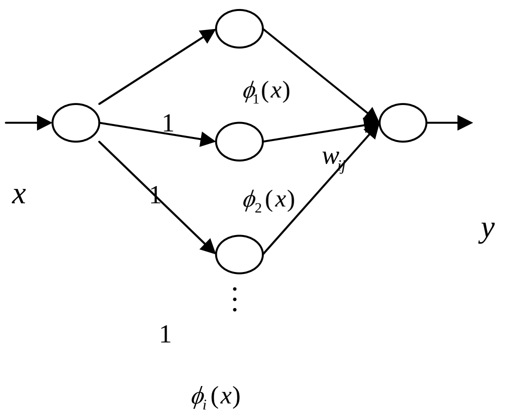

Fig.1: The structure of neural network.

Test methods	The Sum- squared error of the N times		
	N =400	N =500	N =800
Moment-um BP	0.00295121	0.00093582	0.00021402
DE	5.38401790e-4	5.79813322e-6	6.84449810e-7
MDE	3.45230215e-5	1.50693080e-7	6.16873208e-10

Table.1: Comparison of some training methods on $\sin(x)$ function.

The methods	The Sum- squared error of the N times		
	N =400	N =700	N =1000
Momentum BP	0.00620533	0.00529263	0.00234263
DE	7.95702031e-4	1.53390457e-5	7.404452057e-8
MDE	5.29506607e-5	5.74123437e-7	5.22650606e-11

Table.2: Comparison of some training methods on $\cos(x)$ function.

Momentum BP	DE	MDE	Real number
2.37025779	2.45410263	2.45960489	2.45960311
2.16148791	2.23040291	2.22554096	2.22554093
1.43886037	1.35658042	1.34985867	1.34985881
0.46079542	0.48160357	0.49658524	0.49658530

Table.3: Comparison of the real number and the test number for 20 times on e^x.

Table.1 to Table.3 show the results of training 50 times. For the function, e^x, each algorithm was run 50 times and iterated 800 epochs, Figure.2 and Figure.3 are the best approximation curves during the running and the result in Figure.2 was simulated in the neural network toolbox.

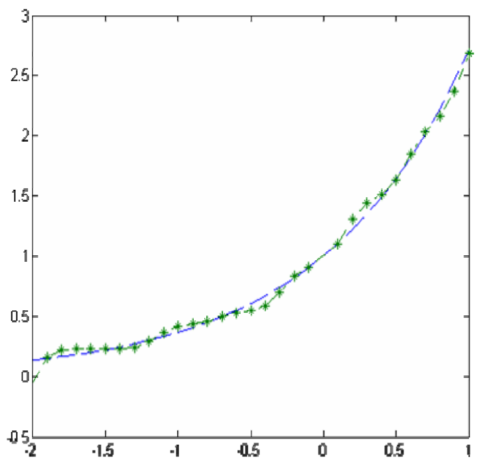

Fig. 2: Approximation curve of e^x by momentum BP algorithm.
-*-*- Approximation curve by momentum BP;
------ Target curve.

In order to test this modified algorithm further, another typical nonlinear function $e^{-x}\sin(2\pi x)$ was tested. The structure of neural network is 1-10-1; the other parameters are the same as above. Each algorithm was run 30 times and iterated 200 epochs, considering the space of this paper, the best approximation curves by MDE, DE and momentum BP are only given, which are shown in Figure 4 and Figure 5.

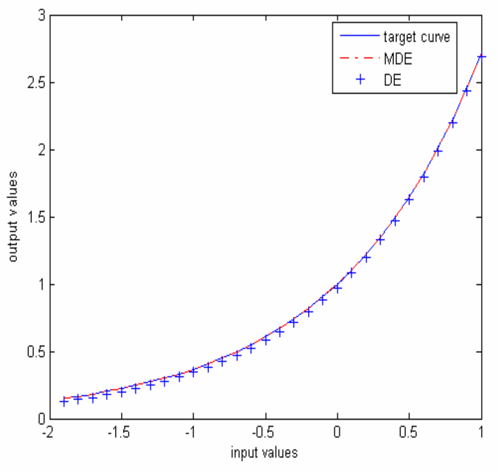

Fig.3: Approximation curves of e^x by MDE and DE.

The experiment results show that MDE algorithm has high-speed global convergence and can obtain the global optional solution to the optimization problems.

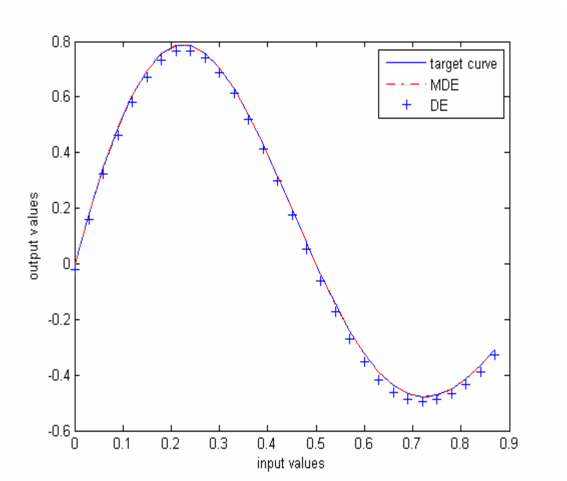

Fig.4: A pproximation curves of $e^{-x}\sin(2\pi x)$ by MDE and DE.

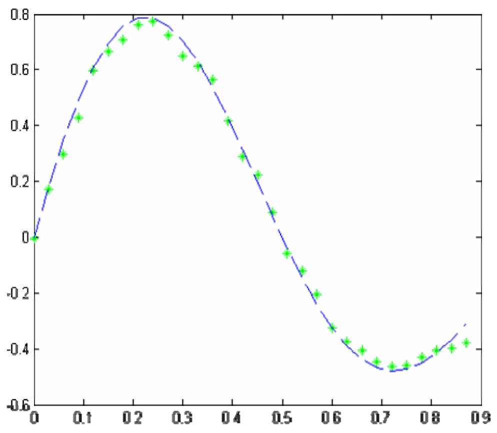

Fig.5: Approximation curve of $e^{-x}\sin(2\pi x)$ by momentum BP algorithm.
-*-*- Approximation curve by momentum BP;
------ Target curve.

6. Conclusions

In this paper, a kind of MDE algorithm is proposed and is used to train three layers feed-forward neural network. The results of test show that the MDE algorithm has faster convergence speed, and it can obtain the lesser mean square error, it is superior to the basic DE algorithm and momentum BP algorithm. The method has more obvious advantages in training feed-forward neural network.

Acknowledgement

This work is supported by National Nature Science Foundation of China (Grant No.60461001) and Nature Science Foundation of Guangxi (Grant No.0542048).

References

[1] R. Storn, K. Price, Differential Evolution - A simple and efficient adaptive scherne for global optimization over continuous space, *Technical Report TR-95-012*, International Computer Science Institute, 8:22-23,1995.

[2] N. K. Madavan, Multiobjective optimization using a Pareto differential evolution approach**.** *Prof. of the 2002 Congress on Evolutionary Computation*, pp.145-150, 2002.

[3] J, Vesterstrom, R. Thomsen, A comparative study of differential evolution, particle swarm optimization. and evolutionary algorithms on numerical benchmark problems**.** *Congress on Evolutionary Computation*, pp. 980-987, 2004.

[4] Z. Xu. *Computer intelligence* (the first volume) (in Chinese). Beijing: Higher education press, 2004.

[5] S. Hu. *Neural network introduction* (in Chinese). Changsha: National Defense Science and Technology University Press, 1993.

[6] J. Yang, Z. Wu, C. Gu, Dam deformation monitoring model and forecast based on BP algorithm of artificial neural network(in Chinese). *Journal of Xi'an University of Technology*, 17:25-29. 2001.(in Chinese)