Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Neurocomputing 63 (2005) 527–533
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Letters

# An improved PSO-based ANN with simulated annealing technique

Yi Da[a,*], Ge Xiurun[a,b]

[a]*School of Naval Architecture, Ocean and Civil Engineering, Shanghai Jiaotong University, Box A0210091 Shanghai 200240, People's Republic of China*
[b]*Institute of Rock and Soil Mechanics, The Chinese Academy of Sciences, Wuhan, People's Republic of China*

Received 3 June 2004; received in revised form 10 July 2004
Communicated by R. Newcomb
Available online 9 December 2004

**Abstract**

This paper presents a modified particle swarm optimization (PSO) with simulated annealing (SA) technique. An improved PSO-based artificial neural network (ANN) is developed. The results show that the proposed SAPSO-based ANN has a better ability to escape from a local optimum and is more effective than the conventional PSO-based ANN.


*Keywords:* Artificial neural networks; Particle swarm optimization; Simulated annealing

## 1. Introduction

In recent years, the use of artificial neural networks (ANN), in particular, three-layer feedforward neural networks, which can be trained to approximate virtually any continuous function [2,6], becomes popular in practice. In this study, three-layer feedforward neural networks are used. Similar to all neural network models, three-layer feedforward neural networks must be trained to obtain predicting ability with a

*Corresponding author. Tel.: +84-21-54741004.
*E-mail address:* yida@sjtu.edu.cn (Y. Da).

0925-2312/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2004.07.002

training set. Concerning the training, the back-propagation (BP) algorithm is mostly used to perform such a task. BP algorithm is a gradient-based method; hence some inherent problems are frequently encountered in the use of this algorithm, e.g., very slow convergence speed in training, easily to get stuck in a local minimum, etc [5].

The particle swarm optimization (PSO), firstly proposed by Eberhart and Kennedy [1], is a computational intelligence technique [1,3,4]. Some researchers have used PSO to train neural networks and found that PSO-based ANN has a better training performance, faster convergence rate, as well as a better predicting ability than BP-based ANN [5,7].

In this study, in order to improve the ability of conventional PSO to escape from a local optimum, the simulated annealing (SA) technique is used to modify PSO, a SAPSO-based ANN is developed and an example is studied.

## 2. SAPSO-based ANN

A typical three-layer feedforward neural network consists of an input layer, a hidden layer and an output layer. The nodes are connected by weights and output signals, which are a function of the sum of the inputs to the node modified by a simple nonlinear activation function. In this study, the activation function is the sigmoid function with threshold defined as

$$f\left(\sum_{i=1}^{n} w_i x_i - \theta\right) = 1/\left\{1 + \exp\left[-\left(\sum_{i=1}^{n} w_i x_i - \theta\right)\right]\right\}, \tag{1}$$

where $x_i$ is the input to the node and $w_i$ is the corresponding input weight, $\theta$ is a value which is usually called the threshold, $n$ is the number of the inputs to the node. The output of a node is scaled by the connecting weight and fed forward to be an input to the nodes in the next layer of the network. The input layer plays no computational role but merely serves to pass the input vector to the network. The input layer and the hidden layer are connected by weights and likewise the hidden layer and output layer also have connection weights. The network has the ability to learn through training. The training requires a set of training data, i.e., a series of input and associated output vectors. During the training, the network is repeatedly presented with the training data and the weights and thresholds in the network are adjusted from time to time till the desired input–output mapping occurs.

PSO is an optimization algorithm, modeled after the social behavior of flocks of birds. PSO is a population-based search process where individuals initialized with a population of random solutions, referred to as particles, are grouped into a swarm. Each particle in the swarm represents a candidate solution to the optimization problem, and if the solution is made up of a set of variables, the particle can correspondingly be a vector of variables. In a PSO system, each particle is ''flown'' through the multidimensional search space, adjusting its position in search space according to its own experience and that of neighboring particles. The particle therefore makes use of the best position encountered by itself and that of its

neighbors to position itself toward an optimal solution. The performance of each particle is evaluated using a predefined fitness function, which encapsulates the characteristics of the optimization problem. In this study, the larger the value of fitness function, the better the particle is.

In each iteration, every particle calculates its velocity according to the following formula:

$$v_i(t+1) = wv_i(t) + c_1 r_1(p_{id} - x_i(t)) + c_2 r_2(p_{gd} - x_i(t)), \tag{2}$$

where $t$ is the current step number, $w$ is the inertia weight, $c_1$ and $c_2$ are the acceleration constants, $r_1$ and $r_2$ are two random numbers in the range [0, 1], $x_i(t)$ is the current position of the particle, $p_{id}$ is the best one of the solutions this particle has reached, $p_{gd}$ is the best one of the solutions all the particles have reached.

After calculating the velocity, the new position of every particle can be worked out

$$x_i(t+1) = x_i(t) + v_i(t+1). \tag{3}$$

The PSO algorithm performs repeated applications of the update equations above until a specified number of iteration has been exceeded, or until the velocity updates are close to zero.

SA is another intelligent method to solve optimization problems [8]. In the search process, the SA accepts not only better but also worse neighboring solutions with a certain probability. Such mechanism can be regarded as a trial to explore new space for new solutions, either better or worse. The probability of accepting a worse solution is larger at higher initial temperature. As the temperature decreases, the probability of accepting worse solutions gradually approaches zero. This feature means that the SA technique makes it possible to jump out of a local optimum to search for the global optimum. In this study, the SA technique is used to deal with every particle as: if $p_{id} > p_{gd}$, accept $p_{gd} = p_{id}$ with the probability 1; if $p_{id} < p_{gd}$, accept $p_{gd} = p_{id}$ with the probability $prob$ defined as

$$\left.\begin{aligned} prob &= 1 - \exp\left(-\frac{p_{gd} - p_{id}}{temp}\right) \\ temp &= itempt \end{aligned}\right\}, \tag{4}$$

where $prob$ is the probability, $temp$ is the current temperature, $itemp$ is a constant selected as initial temperature, $p_{id}$ is the best one of the solutions this particle has reached, $p_{gd}$ is the best one of the solutions all the particles have reached, and $t$ is the current step number.

The above treatment can increase the diversity in the particles and enable PSO to accept a bad solution with the probability, which will gradually decrease to zero as the temperature increases. Although the operation here is not the same as that of the traditional SA, in which the temperature decreases, their spirits are the same.

The above SAPSO is adopted to train the three-layer neural network. Without loss of generality, it is denoted that vector $W$ is the vector including all the weights and thresholds in the network. When SAPSO is used to train the network, the $i$th particle is represented as $W_i$. The fitness function, which measures the performance of each

particle, is defined as

$$fitness = -\sqrt{\frac{\sum_{i=1}^{np}\sum_{i=1}^{no}(y_j - o_{ij})^2}{nonp}}, \tag{5}$$

where $y_i$ is the expected output, $o_{ij}$ is the predicted output of the network, *no* is the number of output nodes, and *np* is the number of training set samples. The larger the *fitness*, the better is the set of weights and thresholds.

Thus a SAPSO-based ANN is developed, of which the goal is to obtain a set of weights and thresholds that maximizes the fitness function. If the SA technique is not used, it is a conventional PSO-based ANN.

## 3. Application of SAPSO-based ANN

In rock engineering, triaxial compression tests are usually carried out on rock samples in order to get some important information about the rock material used. The relationship between the confining pressure and the peak stress as well as its corresponding strain, which is complex and nonlinear, is an important property of the material. In this study, as an example, SAPSO-based and conventional PSO-based ANN were applied to modeling the relationship between the confining pressure and the peak stress as well as its corresponding strain based on the experimental data obtained during the tests.

We carried out triaxial compression tests on marble samples under six different kinds of confining pressures. Six sets of experimental data were obtained (Table 1), among which the first five sets were training data and the last set served to verify the successfulness of the trained networks. Every set consists of three measurements, the confining pressure, the input of the network, the peak stress and its corresponding strain, the outputs of the network. In order to suit the consistency of the network, all source data were firstly normalized in the range [0.4, 0.8], and then returned to original values after the simulation using $x_{\text{norm}} = 0.4 + 0.4(x - x_{\min})/(x_{\max} - x_{\min})$.

Table 1
The comparison between the outputs of the trained networks and the expected outputs

| No. | Input | Expected output | | SAPSO-based ANN output | | PSO-based ANN output | |
|---|---|---|---|---|---|---|---|
| | Pressure | Stress | Strain | Stress | Strain | Stress | Strain |
| 1 | 0 | 89.29 | 4.124 | 89.263110 | 4.124842 | 89.229359 | 4.226999 |
| 2 | 5 | 107.361 | 8.094 | 107.487027 | 8.088164 | 108.105204 | 7.499647 |
| 3 | 10 | 131.0175 | 8.623 | 130.882039 | 8.627882 | 130.287268 | 9.306268 |
| 4 | 20 | 193.535 | 15.202 | 193.586608 | 15.200651 | 193.615973 | 14.993994 |
| 5 | 30 | 221.201 | 25.715 | 221.194449 | 25.715119 | 221.277049 | 25.731990 |
| 6 | 15 | 160.829 | 10.724 | 160.310250 | 10.511718 | 159.951287 | 11.614892 |

In this case, the structure of the network was 1-8-2, one node in the input layer, eight nodes in the hidden layer and two nodes in the output layer. Every weight or threshold in the network was in the range [−20, 20]. The population size was 200, 200 particles for the swarm. Every initial particle was a set of weights and thresholds generated at random in the range [−20, 20]. The inertia weight $w$ was 0.3. The acceleration constants $c_1$ and $c_2$ were the same 2.0. The maximum velocity was 20 and the minimum velocity was −20. The initial velocities of the initial particles were generated at random in the range [−20, 20]. In each iteration, if the calculated velocity was larger or smaller than the maximum velocity or the minimum velocity, it would be reset to 20 or −20. The initial temperature *itemp* was 0.53. The maximum step of iteration was 30000.

After training, the best *fitness* of SAPSO-based ANN is −0.061418489, and that of PSO-based ANN is −0.4446763. The weights and thresholds of SAPSO-based ANN are shown as

$$W_1 = \begin{bmatrix} -4.042006 & -1.165408 & 20.000000 & 1.710849 & -20.000000 & 2.689824 \\ 20.000000 & 20.000000 & -9.551829 & & -13.352369 & \\ -20.000000 & 4.279752 & -20.000000 & & 20.000000 & \\ 16.531106 & -16.692980 & 20.000000 & & -10.878871 & \\ -20.000000 & -20.000000 & -20.000000 & & -20.000000 & \\ -17.266809 & 11.445778 & -1.710270 & & -1.244230 & \\ 18.863501 & -4.085601 & 8.734860 & & 12.530377 & \\ 4.730707 & -5.923173 & -1.975544 & & 1.866153 & \end{bmatrix}, \tag{6}$$

where the each value in the first column is the input weight of each one of the eight nodes in the hidden layer and the values in the second column are the corresponding thresholds, the values in the third column are the input weights of one of the two nodes in the output layer and the value in the fourth column is the threshold, the values in the fifth column are the input weights of the other node in the output layer and the value in the sixth column is the threshold. In the same way, the weights and thresholds of PSO-based ANN are shown as

$$W_2 = \begin{bmatrix} -20.000000 & 20.000000 & 13.165425 & 3.345577 & 1.206396 & 8.484030 \\ -20.000000 & 20.000000 & -7.131512 & & -8.533140 & \\ -20.000000 & -20.000000 & -20.000000 & & -20.000000 & \\ 4.409953 & -4.895071 & 10.093513 & & 20.000000 & \\ -20.000000 & 4.175293 & -17.255388 & & -4.539283 & \\ 19.992865 & -18.671456 & 5.563109 & & -20.000000 & \\ 7.237641 & 0.575593 & -20.000000 & & -20.000000 & \\ 14.409052 & -1.295866 & 9.675679 & & 17.289934 & \end{bmatrix}. \tag{7}$$

The complete fitness curves of SAPSO-based and PSO-based ANN during training are shown in Fig. 1(a). The comparison between the outputs of the trained

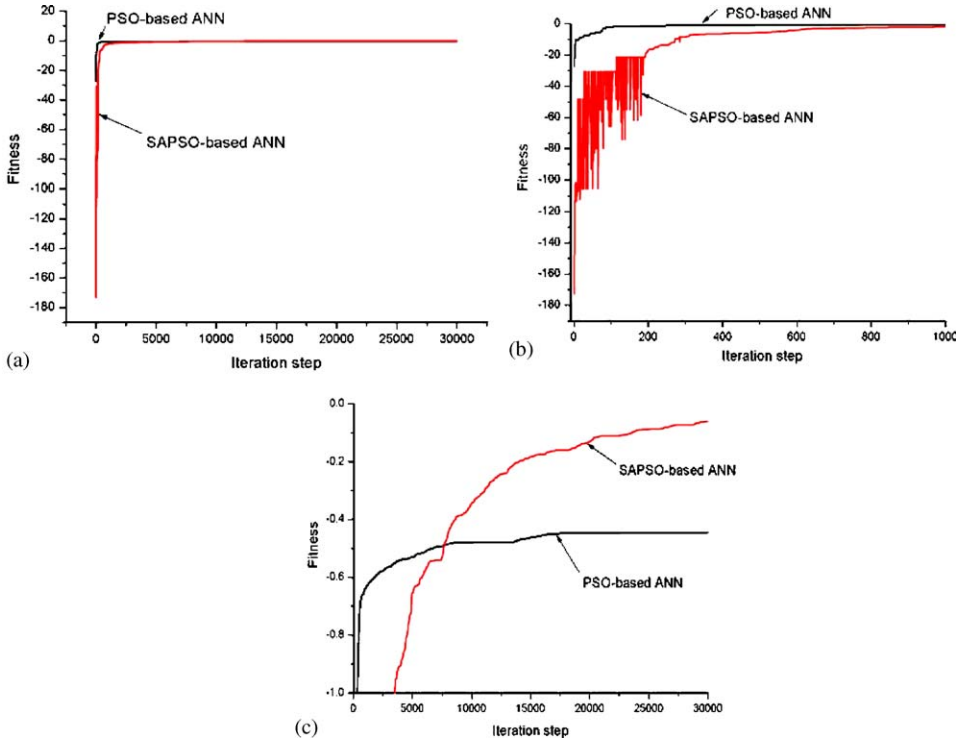


Fig. 1. The fitness curves of SAPSO-based and PSO-based ANN during training. (a) The complete curves during training, (b) The curves in first 1000 steps, (c) The curves that fitness ordinate in the range [−1.0, 0].

networks and the expected outputs is in Table 1. The computing time of training SAPSO- and PSO-based ANN were almost the same, nearly 3 min, on the computer equipped with AMD 1800 + CPU and 256 M memory.

From Fig. 1(b), it can be seen that some parts of the fitness curve of SAPSO-based ANN obviously vibrate but the fitness curve of PSO-based ANN does not, which indicates that the SAPSO-based ANN accepted worse solutions sometimes during training and shows its better ability to escape from a local optimum. From Fig. 1(c), we find that the SAPSO-based ANN finally got a better *fitness* than the PSO-based ANN, and at the end of the iteration, the PSO-based ANN converged but the SAPSO-based ANN still had a tendency to reach better solutions, which was caused by the SA technique applied to increasing the diversity in the particles. All these show the SAPSO-based ANN is more effective than the conventional PSO-based ANN.

From the analysis of the data in Table 1, it can be found that the nonlinear relationship between the peak stress, its corresponding strain and the confining pressure of the marble can be modeled by using a SAPSO- and a PSO-based ANN on the basis of experimental data, and the model established by using the SAPSO-

based ANN is more successful and obviously better than that established by using the PSO-based ANN.

## 4. Conclusions

A SAPSO-based ANN is presented. The results show that the proposed algorithm has a better ability to escape from a local optimum, a better training performance as well as a better predicting ability than the conventional PSO-based ANN in this case. Owing to the flexibility of this approach, it can be adapted to other problems very easily.

## References

[1] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, October 1995, pp. 39–43.
[2] B. Irie, S. Miyake, Capability of three-layered perceptrons, Proceedings of IEEE International Conference on Neural Networks, San Diego, USA, July 1988, pp. 641–648.
[3] J. Kennedy, The particle swarm: social adaptation of knowledge, Proceedings of the 1997 International Conference on Evolutionary Computation, Indianapolis, Indiana, USA, April 1997, pp. 303–308.
[4] J. Kennedy, R.C. Eberhart, Particle swarm optimization, Proceedings of the 1995 IEEE International Conference on Neural Networks, Perth, Australia, December 1995, pp. 1942–1948.
[5] W.Z. Lu, H.Y. Fan, S.M. Lo, Application of evolutionary neural network method in predicting pollutant levels in downtown area of Hong Kong, Neurocomputing 51 (2003) 387–400.
[6] J.L. Meng, Z.Y. Sun, Application of combined neural networks in nonlinear function approximation, Proceedings of the Third World Congress on Intelligent Control and Automation, Hefei, China, June 2000, pp. 839–841.
[7] C.K. Zhang, H.H. Shao, An ANN's evolved by a new evolutionary system and its application, Proceedings of the 39th IEEE Conference on Decision and Control, Sydney, Australia, December 2000, pp. 3562–3563.
[8] M. Zhou, S.D. Sun, Genetic Algorithms: Theory and Application, National Defence Industry Press, Beijing, China, 1999.