Volume 9, Number 6
1009-1963/2000/09(06)/0408-06
CHINESE PHYSICS
June, 2000
© 2000 Chin. Phys. Soc.

FAST EVOLVING MULTI-LAYER PERCEPTRONS FOR NOISY CHAOTIC TIME SERIES MODELING AND PREDICTIONS*

Zhang Jia-shu(张家树) and Xiao Xian-ci(肖先赐)
Department of Electronic Engineering, University of Electronic Science and Technology of China, Chengdu 610054, *China*

(Received 17 January 2000; revised manuscript received 5 March 2000)

A fast evolutionary programming (FEP) is proposed to train multi-layer perceptrons (MLP) for noisy chaotic time series modeling and predictions. This FEP, which uses a Cauchy mutation operator that results in a significantly faster convergence to the optimal solution, can help MLP to escape from local minima. A comparison against back-propagation-trained networks was performed. Numerical experimental results show that the FEP can help MLP better capturing dynamics from noisy chaotic time series than the back-propagation algorithm and produce a more consistently modeling and prediction.

PACC: 0545

I. INTRODUCTION

Recently nonlinear dynamic modeling and prediction of noisy chaotic time series have become a very active field[1−10] since we may model the background noise and synthesize very complex time waveforms such as sea clutter.[5] Multi-layer perceptron (MLP) with time-delay inputs was successfully used to predict a chaotic time series.[2,3] Therefore, in principle, MLP can be used to implement the reconstruction space prediction necessary for dynamic modeling of noisy chaotic time series because of their ability to recognize both spatial and temporal patterns and to learn the underlying dynamics from the time series directly.[11] However, the MLP trained by conventional back-propagation algorithm performed inconsistently for reconstructing strange attractor.[3] Back-propagation algorithms are derived from the method of steepest descent, making them susceptible to becoming trapped at local optima. Albano *et al.*[3] suggested doing several runs of the back-propagation algorithm with different initial weights to alleviate this trouble (even then one cannot be assured that a reconstructed attractor was achievable).

Evolutionary programming (EP) has received much attention in the past few years because of its powerful optimization and structure determining capabilities.[12] Advantages include ease of implementation, insensitivity to the type of criterion function, and inherent parallelizability. The EP has been applied to varioius signal processing problems, such as a finite precision filter design,[13] time series modeling.[14−16] These results demonstrate the potential of the EP approach for training MLP's.

In the modeling and prediction of noisy chaotic time series, the MLP is used to reconstruct the attractor of a dynamical system where only a small set of noisy samples from that system is available. The significant contribution of this work is that fast evolutionary programming (FEP) is first used for training MLP. Our results indicate that a FEP-trained MLP will consistently outperform those trained via back-propagation in this type of application.

II. FAST EVOLUTIONARY PROGRAMMING

Fogel *et al.*[12] proposed a population-based random search strategy termed evolutionary programming, where instead of keeping the single best point each generation, a population of search points is maintained. During each generation (iteration of the EP algorithm) a population of individuals (potential solutions) are evolved to produce new solutions. Only the highest-fit solutions survive to become parents for the next generation. FEP, which uses a Cauchy mutation operator,[16] has been empirically shown to perform better than EP using the Gaussian mutations for several multi-modal functions with many local minima. Furthermore it is comparable to EP in performance for unimodal and multimodal functions with only a few local minima. FEP is implemented as follows:

1) *Initialization*: Generate the initial population of μ randomly selected individuals, and set the generation number, n to 1. Each individual is taken as a pair of real-valued vectors, $(x_i, \eta_i), \forall i \in \{1, \cdots, \mu\}$, where $\{x_i\}$ is the vector elements and $\{\eta_i\}$ the corresponding

* Project supported by the National Defense Foundation of China (Grant No. 98JS05.4.1DZ0205).

variances. Typically the set of x_i' is selected randomly following a uniform distribution over the search space.

2) Evaluate the error score for each individual, $(x_i, \eta_i), \forall i \in \{1, \cdots, \mu\}$, in terms of the objective function $f[x_i(k)]$.

3) *Mutation*: Each parent $(x_i, \eta_i), \forall i \in \{1, \cdots, \mu\}$, create a single offspring x_i', η_i' by

$$x_i'(j) = x_i(j) + \eta_i(j)C(0,1), \tag{1}$$

$$\eta_i'(j) = \eta_i(j)\exp[\tau' N(0,1) + \tau N_j(0,1)], \tag{2}$$

for $j = 1, \cdots, n$, where $x_i(j), \eta_i(j)$ and $x_i'(j), \eta_i'(j)$ denote the jth components of the vectors $x_i, x_i', \eta_i, \eta_i'$, respectively, $C(0,1)$ produces a random variable with a Cauchy distribution centered at the origin and with the scale parameter equal to 1, $N(0,1)$ a normally distributed one-dimensional random variable number with mean zero and standard deviation one, $N_j(0,1)$ indicates that the random number is generated anew for each value of j, the factor τ and τ' are set to $(\sqrt{2\sqrt{n}})^{-1}$ and $(\sqrt{2n})^{-1}$, respectively.

4) *Fitness*: Calculate the error of each offspring $(x_i', \eta_i'), \forall i \in \{1, \cdots, \mu\}$.

5) Conduct pair-wise comparison over the union of parents (x_i, η_i) and $(x_i', \eta_i'), \forall i \in \{1, \cdots, \mu\}$. For each individual, q opponents are chosen randomly from all the parents and offsprings with equal probability. For each comparison, if the individual's error is no greater than the opponents, it receives a "win".

6) *Selection*: Select the μ individuals out of (x_i, η_i) and $(x_i', \eta_i'), \forall i \in \{1, \cdots, \mu\}$, that have the most wins to be parents of next generation.

7) Stop if the halting criterion is satisfied, otherwise, increment to generation number, that is $k = k + 1$, and go to step 3.

Ⅲ. MULTI-LAYER PERCEPTRONS AND NOISY CHAOTIC TIME SERIES MODELING

Suppose a scalar time series $\{x(t), t = 1, 2, \cdots, N\}$ is a measurement on the chaotic dynamical system in the state space. If the embedding dimension m and the time delay τ have been chosen for the phase reconstruction, a reconstructed vector can be written as follows:

$$\boldsymbol{x}(t) = (x(t), x(t+\tau), \cdots, x(t+(m-1)\tau)). \tag{3}$$

By Taken's embedding theorem or its extensions, there exists a function F such that

$$x(t+T) = F(\boldsymbol{x}(t)) \qquad t = 1, 2, \cdots, N, \tag{4}$$

where integer T is the look-ahead time. Let the reconstructed attractor be denoted by S, then $F : S \to S$. Thus it is necessary to fit F only in the domain S by using prediction model. A model for time series is obtained by finding a functional form $\hat{F}$ for the training set pairs, $(\boldsymbol{x}(t), d(t)), t = 1, \cdots, N$, where $d(t) = x(t+T)$ is the output of the model, and $\boldsymbol{x}(t)$ is the input of the model. $\hat{F}$ relates $\boldsymbol{x}(t)$ to $d(t)$:

$$x(t+T) = \hat{F}(\boldsymbol{x}(t)) + \varepsilon(t), \tag{5}$$

where $\varepsilon(t)$ represents the noise or fitting error.

In this case, a good predictive model for a time series should have good training and predictive accuracy in the presence of noise or fitting error, and should only require a reasonable amount of computing time and data. As is well-known, artificial neural networks have gained a lot of interest as empirical models for their powerful representational capacity, fault tolerance ability and multi-input-and-output mapping characteristics.[11] More recently, back-propagating MLP networks[3] have been used to predict chaotic time series presumably noise-free data as input. Figure 1 shows the architecture of a MLP, with M input units, H units in single hidden layer, and one output unit. A time series $\{x(k), k = 1, \cdots, N\}$ is used as a trained set, M successive values of the time series, $x(k-M), x(k-M+1), \cdots, x(k-1)$, are used to predict $x(k+T)$, where integer T is the look-ahead time, the output of the MLP at the time $k+T$ is then

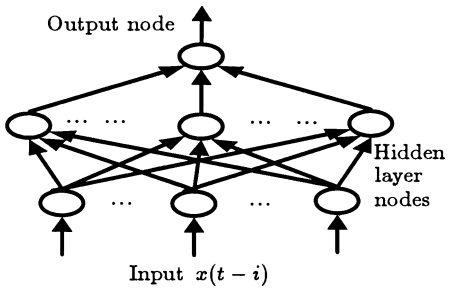


Fig.1. Multi-layer perceptrons.

$$x(t+T) = \sum_{j=1}^{L} w_j^2 h_j + \theta_2, \tag{6}$$

$$h_j = f\Big(\sum_{i=1}^{M} w_{ij}^1 \; x(t-i) + \theta_j^1\Big), \tag{7}$$

$$f(x) = (1 - \mathrm{e}^{-x})/(1 + \mathrm{e}^{-x}), \tag{8}$$

the synaptic weights w_i and w_{ij} are estimated from a training sample $\{x(k), k = 1, \cdots, N\}$, θ_j^1 and θ_2 are the thresholds of the hidden neurons and the output node,

respectively. Proper settings for the synaptic weights allow the MLP to "average" out the noise and capture any underlying dynamics in a time series.[14] The MLP is trained using a time series of N samples. The objective is to select the synaptic weights such that the mean-square error (denoted by ε^2) should be below some given threshold. The quantity ε^2 is defined as

$$\varepsilon^2 = \sum_{k=2}^{N-T} [x(k+T) - \hat{x}(k+T)]^2, \tag{9}$$

where the current input to the MLP is $x(k-M), x(k-M+1), \cdots, x(k-1)$. In the above equation $\hat{x}(k+T)$ is the prediction of the sample $x(k+T)$.

With respect to our current problem, each individual in the population is an MLP and the associated synaptic weight matrix $\boldsymbol{W}$ completely describes its "genetic makeup", an MLP (genotype) receives an input time series (environment) which yields a particular ε^2 (evaluated phenotype). The mutation operator used for reproduction is quite simple. Recall that $w_{ij} \in \boldsymbol{W}$ is the weight that the ith node attaches to the jth input. Individuals in the population are composed of both object parameters (w_{ij}'s and w_k's) and strategy parameters (σ_{ij}'s and σ_k's). Each weight is perturbed by

$$w'_{ij} = w_{ij} + \sigma_{ij} C_{ij}(0,1), \tag{10}$$

$$\sigma'_{ij} = \sigma_{ij} \exp[\tau' N(0,1) + \tau N_j(0,1)]. \tag{11}$$

The notation $C_{ij}(0,1)$ denotes that a new random variable is generated for each w'_{ij}. Perturbing all of the weights in a row modifies how all the inputs to a specific node are processed. Conversely, perturbing all of the weights in a column modifies how all the nodes process a specific input. In a similar manner, the weights for output neuron are perturbed by

$$w'_k = w_k + \sigma_k C_k(0,1), \tag{12}$$

$$\sigma'_k = \sigma_k \exp[\tau' N(0,1) + \tau N_k(0,1)], \tag{13}$$

where $w_k \in \boldsymbol{w}$ and $C_k(0,1)$ denotes a new random variable is generated for each w_k.

IV. EXPERIMENTAL RESULTS

To illustrate the effectiveness of using FEP for training MLP to reconstruct attractor, the Hénon map, one of the more widely analyzed 2-dimensional maps, was chosen as an example of chaotic time series, which is defined as

$$\begin{cases} x(n+1) = \alpha - x^2(n) + y(n); \\ y(n+1) = \beta x(n), \quad x(n) \in [-2,2], \quad \alpha = 1.4, \quad \beta = 0.3. \end{cases} \tag{14}$$

Training data for the MLP was obtained by iterating the Hénon map 5000 times, the first 1000 iterations were discarded to remove any possible transients. Each sample was then perturbed by additive noise as follows:

$$x_r(n) = x(n) + 0.3r(n), \tag{15}$$

where $x(n)$ is the Hénon map output and $r(n) \sim N(0,1)$ is a Gaussian-distributed random variable which is sufficient to produce a signal-to- noise ratio (SNR) of about 10db. To show clearly the predictive results of the MLP trained by the FEP for making 1 step future predictions of noisy chaotic series, only one is plotted for each 4 points compared with free-noise values. It was empirically determined that this value of mean-squared error was sufficient to reconstruct the attractor from a noisy data set. While a low ε^2 is the necessary condition, it is not sufficient to guarantee that the chaotic attractor can be reliably reconstructed. A randomly selected data set of 256 successive noisy samples was extracted, although only the first 128 samples were actually used for training. Specifically, the first 128 samples of the noisy data were used to find synaptic weights which yield a low ε^2. So the FEP was run until a candidate MLP with $\varepsilon^2 < 6.0$ in Eq.(9). The other 128 samples were used to select the best MLP out of the final population by the prediction error as

$$\varepsilon_{\mathrm{p}}^2 = \sum_{n=129}^{256} [x_r(n) - \hat{x}(n)]^2, \tag{16}$$

where $x_r(n)$ is noisy data and $x(n)$ the predicted output of MLP. To reconstruct chaotic attractor, the first predicted output $\hat{x}(128)$ was computed using $x_r(128), x_r(127)$ as inputs while the second predicted output $\hat{x}(129)$ computed using $\hat{x}(128)$ and $x_r(128)$ as inputs. Subsequent predicted values were obtained using two successive predicted values which were fed back to the input of the MLP. This recursive process

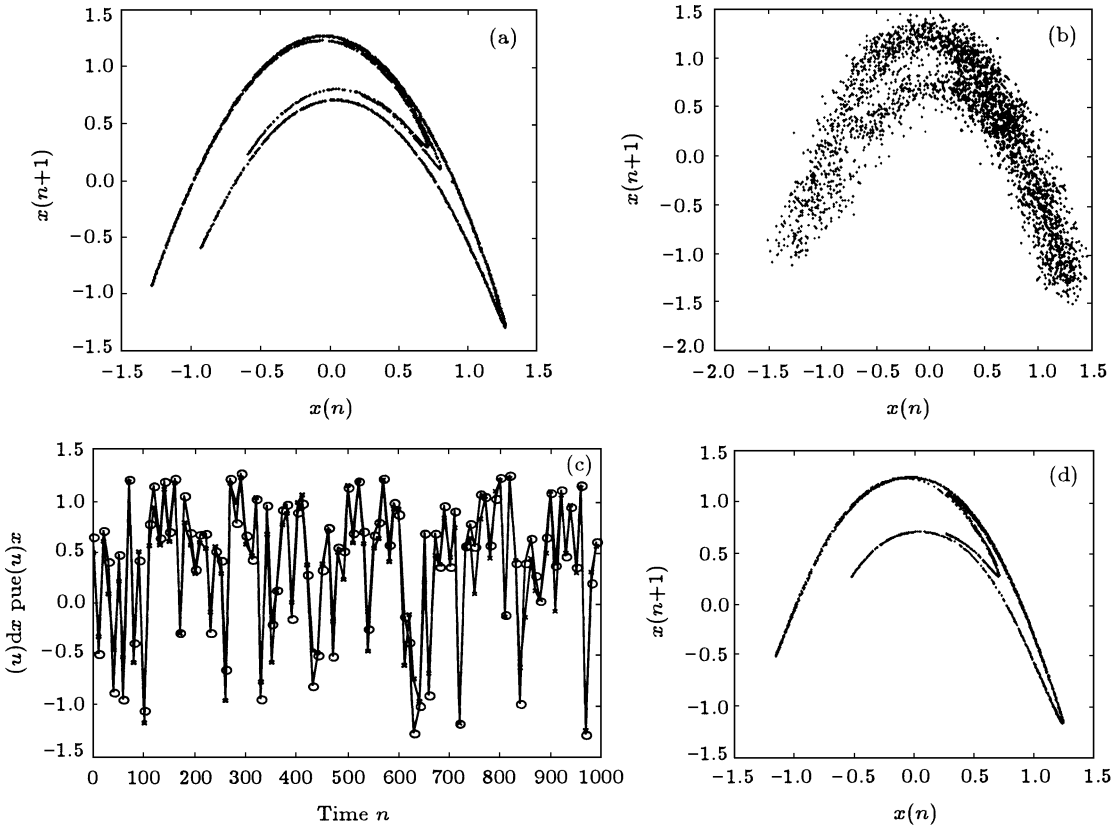


Fig.2. (a) The reconstructed Hénon map chaotic attractor; (b) the noisy Hénon attractor; (c) predicted results of the noisy Hénon chaotic time series (RMS=0.0057,o: predicted values, ×: noise-free true values); (d) the reconstructed Hénon chaotic attractor.

is repeated until 3000 predicted sequence was obtained. The MLP chosen to reconstruct the attractor was the one which minimized the prediction error. In this work, the populatin size was 15. Numerical results are given as follows:

Experiment 1. Modeling and prediction of the same noisy chaotic time series of the Hénon map using the MLP trained by the fast evolutionary programming with different initial weights.

Over 50 independent FEP training sessions were conducted with all sessions using the same input data. All of these FEP-trained MLP reconstructed attractors are smilar to that depicted in Fig.2(d). The reconstructed attractor correlation dimension is $D_2 = 1.2683$.

Experiment 2. Modeling and prediction of the same noisy chaotic time series of the Hénon map using the MLP trained by the back-propagation algorithm with different initial weights.

From the experimental results listed above, we can see clearly that the predicted values for the case of one step agree well with the noise-free true values. A visual comparison of Figs.2(a) and (b) with a 3000-point reconstructed attractor in Fig.2(d) shows that most of the relevant features of the Hénon map attractor were captured. This compares favorably with a previously estimated dimension of 1.25 for the (noise-free) Hénon map attractor. Experimental results in Fig.3 show successfully BP-trained one step MLP predictors normally fail in dynamical modeling of noisy chaotic time series. Even if doing several runs of the back-propagation algorithm with different weights and biases, one cannot be assured that a reasonable reconstruction was achieved, it might reconstruct only short segments, or get trapped in a periodic motion on a small set of points. This may be due to the fact that back-propagation algorithm is derived from the method of steepest descent which makes MLPs susceptible to becoming trapped at local optima. In contrast, over 50 independent FEP training sessions were conducted with all sessions using the same input data. All of these FEP-trained MLP's reconstructed attractors similar to that depicted in Fig.2(d). This implies that FEP training may be a better approach for producing a more consistently performing MLP.

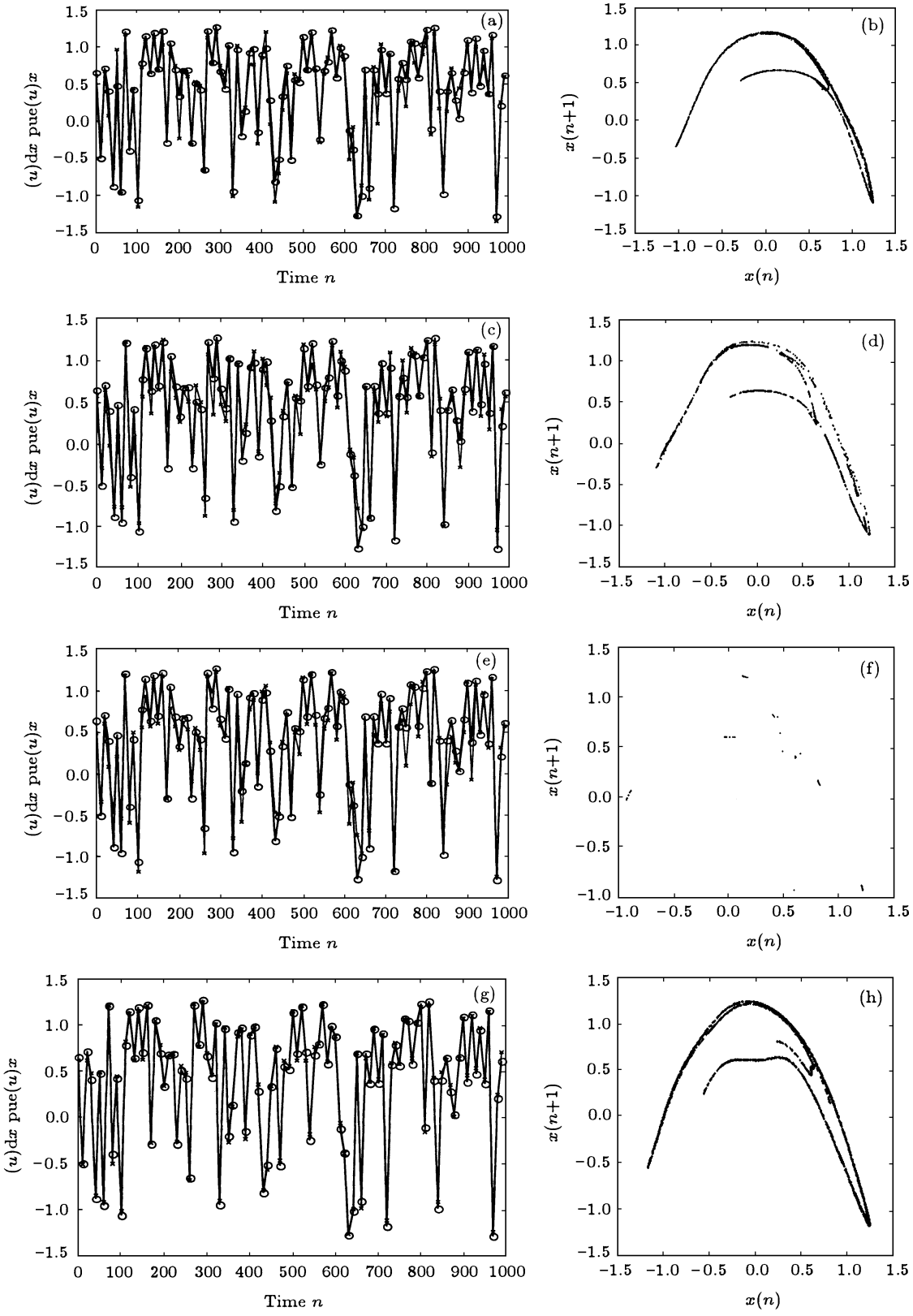


Fig.3. (a) The predicted results of the noisy Hénon chaotic time series (RMS=0.0573, o: predicted values; ×: noise-free true values); (b) the Hénon attractor reconstructed by the weights in (a); (c) the predicted results of the noisy Hénon chaotic time series (RMS=0.067, o: predicted values, ×: noise-free true values); (d) the Hénon attractor reconstructed by the weights in (c); (e) the predicted results of the noisy Hénon chaotic time series (RMS=0.0653, o: predicted values; ×: noise-free true values); (f) the Hénon attractor reconstructed by the weights in (e); (g) the predicted results of the noisy Hénon chaotic time series (RMS=0.0573, o: predicted values; ×: noise-free true values); (h) the Hénon attractor reconstructed by the weights in (g).

V. CONCLUSION

The FEP technique has been successfully applied to optimize the MLP used for noisy chaotic time series modeling and prediction. The performance of the FEP algorithm has been shown to be superior to that of the back-propagation algorithm in term of faster convergence and capability of capturing dynamics from noisy chaotic time series. This better performance may be attributed to the ability of the Cauchy random variable to generate occasionally useful large jumps, thus providing an escape mechanism from local optima. The usefulness of such a property in optimization problems, in the modeling of nonlinear signals, and identification of systems, is an open area for research.

REFERENCES

[1] Y. C. Lai, C. Grebogi, *Phys. Rev. Lett.*, **82**(1999), 4803.

[2] J. N. Hwang, S. Y. Kung, M. Niranjan, J. C. Principe, *IEEE Signal Proc. Lett.,* **14**(November, 1997), 28.

[3] A. M. Albano, A. Passamante, T. Hediger, M. E. Farrell, *Physica*, **D58**(1992), 1.

[4] L. Jaegers, H. Kantz, *Chaos*, **6**(1996), 440.

[5] S. Haykin, X. B. Li, *Proc. IEEE*, **83**(1995), 94.

[6] J. C. Principle, A. Ratheie, J. M. Kuo, *Int. J. Bifur. and Chaos*, **2**(1990), 989.

[7] L. Y. Cao, Y. G. Hong, H. Fang, G. W. He, *Physica*, **D85**(1995), 225.

[8] M. Casdali, *Physica*, **D35**(1989), 335.

[9] J. S. Zhang, X. C. Xiao, *Chin. Phys. Lett.*, **17**(2000), 88.

[10] L. P. Maguire, B. Roche, T. M. McGinnity, L. J. McDaid, *Inform. Sci.*, **112**(1998), 125.

[11] K. Hornik, M. Stinchombe, H. White, *Neural Networks*, **3**(1990), 535.

[12] D. B. Fogel, *IEEE Trans. Neural Networks*, **5**(1994), 3.

[13] S. S. Rao, A. Ramasubrahmanyan, *IEEE Signal Proc. Lett.,* **3**(1996), 137.

[14] G. W. Greenwood, *IEEE Trans. Evolutionary Computation*, **1**(1997), 244.

[15] J. R. McDonnell, D. Waagen, *IEEE Trans. Neural Networks*, **5**(1994),24.

[16] K. Chellapilla, S. S. Rao, *IEEE Signal Proc. Lett.*, **5**(1998), 39.