Evolving Flexible Beta Operator Neural Trees (FBONT) for Time Series Forecasting

Souhir Bouaziz*, Habib Dhahri, and Adel M. Alimi

REsearch Group on Intelligent Machines (REGIM), University of Sfax, National School of Engineers (ENIS), BP 1173, Sfax 3038, Tunisia
souhir.bouaziz@gmail.com, {habib.dhahri,adel.alimi}@ieee.org

Abstract. In this paper, a new time-series forecasting model based on the Flexible Beta Operator Neural Tree (FBONT) is introduced. The FBONT model which has a tree-structural representation is considered as a special Beta basis function multi-layer neural network. Based on the pre-defined Beta operator sets, the FBONT can be formed and optimized. The FBONT structure is developed using the Extended Genetic Programming (EGP) and the Beta parameters and connected weights are optimized by the Particle Swarm Optimization algorithm (PSO). The performance of the proposed method is evaluated using time series forecasting problems and compared with those of related methods.

Keywords: Flexible Beta Operator Neural Tree model, Extended Genetic Programming, Particle Swarm Optimization algorithm, Time-series forecasting.

1 Introduction

Time series forecasting play a major role in the characterization of time series performance by predicting the future value and understanding fundamental features in systems, so it has been a center of attention of several researches. Recently, various nonlinear time series forecasting methods have been proposed such as artificial neural networks (ANN) [1, 2, 3, 4], SVM [5], adaptive algorithms [6, 7], and have been successfully applied.

A neural network's performance depends mainly on two issues which are the network structure and the parameter's adjustment on the continuous parameter space and these issues are closely coupled. For a given problem, the neural network structure is not unique and also it may be a single hidden layer is not enough. Thus, the design of ANN automatically is required and many important attempts have been developed such as evolutionary programming [8], Neuro Evolution of augmenting topologies [9]. Furthermore, weights and kernel parameters of ANNs can be learned by many methods, i.e., back-propagation algorithm [10], genetic algorithm [11], differential evolution algorithm [1, 3, 4], particle swarm optimization algorithm [2] and so on.

Although conventional representation of ANN has the nonlinear approximation capability, it also presents many weaknesses, for example, the neural network's structure

T. Huang et al. (Eds.): ICONIP 2012, Part III, LNCS 7665, pp. 17–24, 2012.

is difficult to regulate, it suffers from slow convergence characteristics and over-fitting phenomenon leading the decline of its generalization, it is prone to be trapped in local minima [12]. Thus a special multi-layer feedforward ANN has been proposed by Chen [13] and it is called flexible neural tree (FNT). FNT allows over-layer connections, input variables selection and different activation functions for different nodes [12]. Recent studies have begun to explore this representation of neural networks in the context of classification [14], recognition [15], approximation [16] and control [17], etc. To form the flexible neuron model, the most used flexible activation function is the Gaussian function. However, the Beta function [18, 19] shows its performance for standard representation of ANN against the Gaussian function due to its great flexibility and its universal approximation ability [1-4, 11]. For these reasons we adopted in this research the flexible Beta function to establish the flexible neuron model.

In this paper, a flexible Beta operator neural tree (FBONT) model is proposed for time-series prediction problem. Based on the predefined Beta operator sets, a flexible Beta operator neural tree model can be created and evolved. The hierarchical structure is evolved using the Extended Genetic Programming (EGP). The fine tuning of the Beta parameters (centre, spread and the form parameters) and weights encoded in the structure is accomplished using the Particle Swarm Optimization algorithm (PSO).

The paper is planned as follows: Section 2 describes the basic flexible Beta operator neural tree model. A hybrid learning algorithm for evolving the Beta function neural tree models is the subject of Section 3. The set of some simulation results are provided in Section 4. Finally, some concluding remarks are presented in Section 5.

2 Flexible Beta Operator Neural Tree Model

In this work, we have used the tree-based encoding method as it defined by Chen [12-17] for representing a FBONT model. The function node set F and terminal node set T used for generating a FBONT model are described as follows:

$$S = F \cup T = \{+_2, +_3, \dots, +_N\} \cup \{x_1, \dots, x_M\} \tag{1}$$

where $+_n$ (n = 2,. . . , N) denote non-terminal nodes and represent flexible neuron Beta operators with n inputs.

$x_1, x_2, \ldots, x_M$ are terminal nodes and defining the input vector values. The output of a non-terminal node is calculated as a flexible neuron model (fig.1).

In the creation process of Beta operator neural tree, if a non-leaf node, i.e., $+_n$ is selected, n real values are randomly created to represent the connected weight between the node $+_n$ and its offspring. In addition, seen that the flexible activation function used in this study is the beta function, four adjustable parameters (the center c_n, width σ_n and the form parameters p_n, q_n) are randomly generated as flexible Beta operator parameters. For each non-terminal node, i.e., $+_n$, its total excitation is calculated by:

$$y_n = \sum_{j=1}^{n} w_j * x_j \tag{2}$$

where x_j(j = 1, …, n) are the inputs to node $+_n$. The output of node $+_n$ is then calculated by:

$$out_n = \beta(y_n, c_n, \sigma_n, p_n, q_n) = \begin{cases} \left[1 + \frac{(p_n+q_n)(y_n - c_{\overline{n}})}{\sigma_n p_n}\right]^{p_n} \left[1 - \frac{(p_n+q_n)(c_n - y_n)}{\sigma_n q_n}\right]^{q_n} \\ \quad if\ y_n \in \left]c_n - \frac{\sigma_n p_n}{p_n+q_n}, c_n + \frac{\sigma_n q_n}{p_n+q_n}\right[\\ 0 \qquad else \end{cases} \quad (3)$$

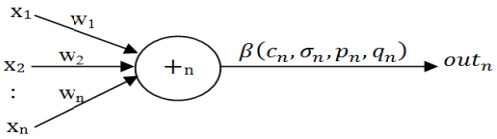

Fig. 1. A flexible neuron Beta operator

A typical flexible Beta operator neural tree model is shown as Fig. 2. The overall output of flexible Beta operator neural tree can be computed recursively by depth-first method from left to right.

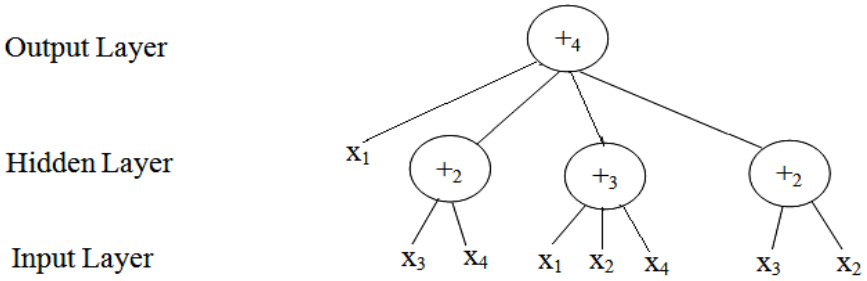

Fig. 2. A typical representation of FBONT: function node set F = {$+_2$, $+_3$, $+_4$}, and terminal node set T = {x_1, x_2, x_3, x_4}

3 The Hybrid FBONT Evolving Algorithm

The optimization of FBONT includes the tree-structure and parameter optimization. In this study, finding an optimal Beta operator neural tree structure is achieved by using Extended Genetic Programming algorithm and the parameters implanted in a FBONT are optimized by PSO.

3.1 Structure Optimization

A number of flexible neural tree variation operators, which are an extension of standard GP, are developed in this work as following:

Selection: In this study firstly a truncation selection is used by ranking all individuals according to their fitness. Then, a threshold *T* (between 0 and 1) is applied such that

the *(1-T)%* best individuals are selected to survive to the next generation and the remaining individuals are removed and replaced with new ones.

Crossover: the tree structure crossover operation is implemented by taking randomly selected sub-trees in the individuals and selecting randomly one non-leaf node in the hidden layer for each chromosome, and then swapping the selected sub-trees.

Mutation: four different mutation operators were used to generate offspring from the parents. These mutation operators are as follows:

1. *Changing one leaf node*: select one leaf node randomly in the neural Beta operator tree and replace it with another leaf node;
2. *Changing all the leaf nodes:* select all leaf nodes in the neural Beta operator tree and replace it with another leaf node;
3. *Growing:* select a random leaf node in hidden layer of the neural Beta operator tree and replace it with a randomly generated sub-tree;
4. *Pruning:* randomly select a Beta operator node in the neural tree and replace it with a random leaf node.

After each mutation or crossover operator, a redundant terminals pruning operator will be applied, if it is possible; i.e. if a Beta operator node has more than two terminals, the redundant terminals should be deleted.

3.2 Parameter Optimization with PSO

PSO was proposed by Kennedy and Eberhart [20] and is inspired by the swarming behavior of animals. The initial population of particles is randomly generated. Each particle has a position vector denoted by x_i. A swarm of particles 'flies' through the search space; with the velocity vector v_i of each particle. Each particle records its best position corresponding to the best fitness in a vector p_i. Moreover, the best position among all the particles obtained in a certain neighborhood of a particle is recorded in a vector p_g. At each iteration, a new velocity for particle *i* is updated by*:*

$$v_i(t+1) = w\, v_i(t) + c_1\varphi_1\big(p_i(t) - x_i(t)\big) + c_2\varphi_2\Big(p_g(t) - x_i(t)\Big) \tag{4}$$

where c_1,c_2 (acceleration) and w (inertia) are positive constant and φ_1 and φ_2 are randomly distributed number in [0,1]. The velocity v_i is limited in[-v_{max} ,+v_{max}]. Based on the calculated velocities, each particle changes its position:

$$x_i(t+1) = x_i(t) + (1-w)v_i(t+1) \tag{5}$$

3.3 Fitness Function

To find an optimal FBONT, the Root Mean Squared Error (RMSE) is employed as a fitness function:

$$Fit(i) = \sqrt{\frac{1}{P}\sum_{j=1}^{P}(y_t^j - y_{out}^j)^2} \quad (6)$$

where P is the total number of samples, y_t^j and y_{out}^j are the actual time-series and the FBONT model output of jth sample. $Fit(i)$ denotes the fitness value of ith individual.

3.4 The Learning Algorithm for FBONT Model

To find an optimal or near-optimal FBONT model, architecture and parameters optimization are used alternately. Combining of the EGP and PSO algorithms, a hybrid algorithm for evolving FBONT model is described as follows and is depicted:

(a) Randomly create an initial population (FBONT trees and its parameters);
(b) Structure optimization is achieved by the Extended Genetic Programming (EGP) as described in section 3.1;
(c) If a better architecture is found or a maximum number of generation is attained, then go to step (d), otherwise go to step (b);
(d) Parameter optimization is achieved by the PSO algorithm. The architecture of FBONT model is fixed, and it is the best tree found by the structure search. The parameters (weights and flexible Beta function parameters) encoded in the best tree formulate a particle;
(e) If the maximum number of iterations is attained, or no better parameter vector is found for a fixed time then go to step (f); otherwise go to step (d);
(f) If satisfactory solution is found, then the algorithm is stopped; otherwise go to step (b).

4 Experimental Results

To evaluate its performance, the proposed FBONT model is submitted to time-series prediction problems: Mackey-Glass chaotic and the Jenkins–Box time series.

4.1 Mackey–Glass Time Series Prediction

A time-series prediction problem can be constructed based on the Mackey–Glass [21] differential equation:

$$\frac{d(x(t))}{dt} = \frac{ax(t-\tau)}{1+x^{10}(t-\tau)} - bx(t) \quad (7)$$

The setting of the experiment varies from one work to another. In this work, the same parameters of [2] and [14], namely a = 0.2, b = 0.1 and $\tau \geq 17$, were adopted, since the results from these works will be used for comparison. 500 data pairs of the series were used as training data, and 500 were used to validate the model identified. The used Beta operator sets to create an optimal FBONT model is $S = F \cup T = \{+_2, +_3, +_4, +_5\} \cup \{x_1, x_2, x_3, x_4\}$, where x_i (i = 1, 2, 3, 4) denotes $x(t), x(t -$

$6), x(t-12)\ and\ x(t-18)$, respectively. After 16 generations, an optimal FBONT model was obtained with RMSE 0.007401. The RMSE value for validation data set is 0.007623. The evolved FBONT and its output are shown in Fig. 3. The proposed system is essentially compared with beta basis function neural network's system using particle swarm optimization (BBFN-PSO) [2] and the FNT model with Gaussian function as flexible neuron operator [13] and also with other systems in Table 1.

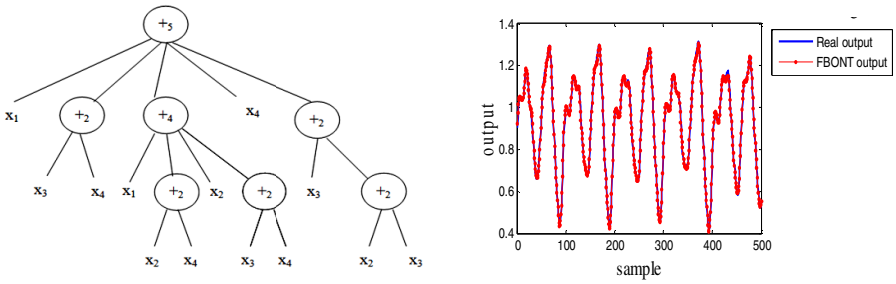

Fig. 3. The evolved FBONT and its output for prediction of the Mackey-Glass time-series

Table 1. Comparison of different methods for the Mackey-Glass time-series

Method	Prediction error (RMSE)
PSO-BBFN [2]	0.027
Habib [4]	0.013
Aouiti [11]	0.013
FNT [13]	0.0069
FBONT	0.0076

4.2 Box and Jenkins' Gas Furnace Problem

The gas furnace data of Box and Jenkins [22] was saved from a combustion process of a methane–air mixture. It is frequently used as a benchmark example for testing prediction algorithms. The input $u(t)$ is the gas flow into the furnace and the output $y(t)$ is the CO_2 concentration in outlet gas. In this work, 200 data samples are used for training and 96 data samples are used for testing the performance of the evolved model. The used instruction set for creating a FBONT model $S = F \bigcup T = \{+_2, +_3, +_4\} \bigcup \{_1, x_2\}$, where x_i (i = 1, 2) denotes $u(t-4), y(t-1)$, respectively. After 16 generations, the optimal Beta operator neural tree model was obtained with the MSE 0.000023. The MSE value for validation data set is 0. 0.000135. The evolved FBONT and its output are shown in Fig. 4. A comparison result of different methods for forecasting Jenkins–Box data is shown in Table 2.

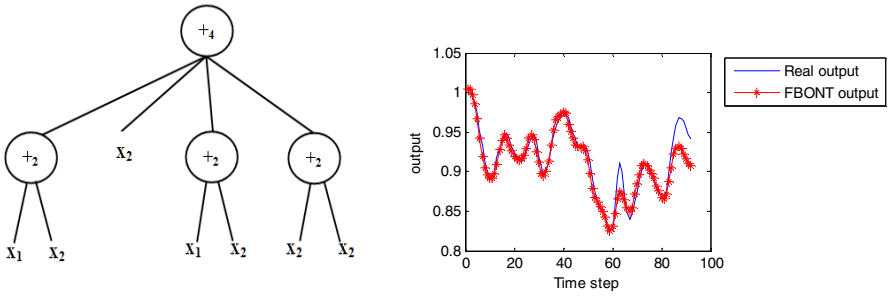

Fig. 4. The evolved FBONT and its output for forecasting Jenkins–Box data

Table 2. Comparison of testing errors of Box and Jenkins

Method	**Prediction error (MSE)**
ANFIS model [23]	0.0073
FuNN model [24]	0.0051
FNT [14]	0.00066
HMDDE [3]	0.0581
FBONT	**0.000135**

5 Conclusion

In this paper, a Flexible Beta Operator Neural Tree model and its design and optimization algorithm are proposed for time-series forecasting problems. The work demonstrates that the FBONT model can successfully evolve the structure and parameters of artificial neural networks simultaneously by using a tree representation. In fact, the FBONT structure is developed using Extended Genetic Programming (EGP) and the Beta parameters and connected weights are optimized by Particle Swarm Optimization algorithm (PSO). The experiment results show that the FBONT model can effectively predict the time-series problem such as Mackey-Glass chaotic time series and the Jenkins–Box time series.

Acknowledgments. The authors would like to acknowledge the financial support of this work by grants from General Direction of Scientific Research (DGRST), Tunisia, under the ARUB program.

References

1. Dhahri, H., Alimi, A.M.: Opposition-based Differential Evolution for Beta Basis Function Neural Network. In: IEEE Congress on Evolutionary Computation, Barcelona, Spain, pp. 1–8 (2010)
2. Dhahri, H., Alimi, A.M., Karray, F.: Designing Beta Basis Function Neural Network for Optimization Using Particle Swarm Optimization. In: IEEE International Joint Conference on Neural Networks, Hong Kong, China, pp. 2564–2571 (2008)

3. Dhahri, H., Alimi, A.M., Abraham, A.: Hierarchical multi-dimensional differential evolution for the design of beta basis function neural network. Neurocomputing 79, 131–140 (2012)
4. Dhahri, H., Alimi, A.M.: The Modified Differential Evolution and the RBF (MDE-RBF) Neural Network for Time Series Prediction. In: Proc. of the International Conference, pp. 5245–5250 (2006)
5. Liu, H., Liu, D., Deng, L.-F.: Chaotic Time Series Prediction Using Fuzzy Sigmoid Kernel-based Support Vector Machines. Chin. Phys. 15(6), 1196–1200 (2006)
6. Li, H., Zhang, J., Xiao, X.: Neural Volterra Filter for Chaotic Time Series Prediction. Chin. Phys. 14(11), 2181–2188 (2005)
7. Meng, Q., Zhang, Q., Mu, W.: A Novel Multi-step Adaptive Prediction Method for Chaotic Time Series. Acta Phys. Sin. 55(4), 1666–1671 (2006)
8. Yao, X., Liu, Y., Lin, G.: Evolutionary Programming Made Faster. IEEE Trans. Evolut. Comput. 3, 82–102 (1999)
9. Stanley, K.O., Miikkulainen, R.: Evolving Neural Networks through Augmenting Topologies. Evolut. Comput. 10, 99–127 (2002)
10. Stepniewski, S.W., Keane, A.J.: Pruning Back-propagation Neural Networks Using Modern Stochastic Optimization Techniques. Neural Comput. Appl. 5, 76–98 (1997)
11. Aouiti, C., Alimi, A.M., Maalej, A.: A Genetic Designed Beta Basis Function Neural Networks for Approximating of Multi-variables Functions. In: Proc. Int. Conf. Artificial Neural Nets and Genetic Algorithms. Springer Computer Science, Prague, Czech Republic, pp. 383–386 (2001)
12. Chen, Y., Yang, B., Meng, Q.: Small-time Scale Network Traffic Prediction Based on Flexible Neural Tree. Appl. Soft Comput. 12, 274–279 (2012)
13. Chen, Y., Yang, B., Dong, J., Abraham, A.: Time-series Forecasting Using Flexible Neural Tree Model. Inf. Sci. 174, 219–235 (2005)
14. Chen, Y., Abraham, A., Yang, B.: Feature Selection and Classification using Flexible Neural Tree. Neurocomput. 70, 305–313 (2006)
15. Chen, Y., Jiang, S., Abraham, A.: Face Recognition Using DCT and Hybrid Flexible Tree. In: Proc. of the International Conference on Neural Networks and Brain, pp. 1459–1463 (2005)
16. Chen, Y., Peng, L., Abraham, A.: Exchange Rate Forecasting Using Flexible Neural Trees. In: Wang, J., Yi, Z., Żurada, J.M., Lu, B.-L., Yin, H. (eds.) ISNN 2006. LNCS, vol. 3973, pp. 518–523. Springer, Heidelberg (2006)
17. Chen, Y., Meng, Q., Zhang, Y.: Optimal Design of Hierarchical B-Spline Networks for Nonlinear System Identification. J. Dynam. Continu. Discrete Impul. Systems Series B (2006)
18. Alimi, A.M.: The Beta System: Toward a Change in Our Use of Neuro-Fuzzy Systems. Int. J. Manag. Invited Paper, 15–19 (2000)
19. Alimi, A.M.: The Beta Fuzzy System: Approximation of Standard Membership Functions. In: Proc. 17eme Journees Tunisiennes d'Electrotechnique et d'Automatique: JTEA 1997, Nabeul, Tunisia, pp. 108–112 (1997)
20. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, pp. 1942–1948. IEEE Press, Piscataway (1995)
21. Lzvbjerg, M., Krink, T.: Extending particle swarms with self-organized criticality. In: Proceedings of the Fourth Congress on Evolutionary Computation (CEC 2002), pp. 1588–1593. IEEE Press, Piscataway (2002)
22. Box, G.E.P., Jenkins, G.M.: Time Series Analysis-Forecasting and Control. Holden Day, San Francisco (1976)
23. Nie, J.: Constructing fuzzy Model by Self-organising Counter Propagation Network. IEEE Trans. Systems Man Cybern. 25, 963–970 (1995)
24. Jang, J.S.R., Sun, C.T., Mizutani, E.: Neuro-fuzzy and Soft Computing: a Computational Approach to Learning and Machine Intelligence. Prentice-Hall, Upper Saddle River (1997)