ESANN'2006 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2006, d-side publi., ISBN 2-930307-06-4.

Optimal Design of Hierarchical Wavelet Networks for Time-series Forecasting

Yuehui Chen[1], Bo Yang[1] and Ajith Abraham[1,2] *

1- School of Information Science and Engineering
Jinan University, Jinan 250022, P.R. China

2- IITA Professorship Program, School of Computer Science
Chung-Ang University, Seoul, Republic of Korea

Abstract. The purpose of this study is to identify the Hierarchical Wavelet Neural Networks (HWNN) and select important input features for each sub-wavelet neural network automatically. Based on the pre-defined instruction/operator sets, a HWNN is created and evolved using tree-structure based Extended Compact Genetic Programming (ECGP), and the parameters are optimized by Differential Evolution (DE) algorithm. This framework also allows input variables selection. Empirical results on benchmark time-series approximation problems indicate that the proposed method is effective and efficient.

1 Introduction

There are three basic multilevel structures for hierarchical models, namely, incremental, aggregated and cascaded. Designing of these hierarchical models faces many difficulties including determination of the hierarchical structure, parameter identification and input variables selection for each sub-models. To investigate the hybrid technique further, a HWNN framework is proposed in this paper. Based on the pre-defined instruction/operator sets, a HWNN network can be created and evolved. HWNN allows input variables selection. In our previous studies, in order to optimize Flexible Neural Tree (FNT) the hierarchical structure was evolved using Probabilistic Incremental Program Evolution algorithm (PIPE) [1][2] and Ant Programming [3] with specific instructions.

In this research, the hierarchical structure is evolved using the Extended Compact Genetic Programming. The fine tuning of the parameters encoded in the structure is accomplished using Differential Evolution (DE) [4]. The proposed method interleaves both optimizations. Starting with random structures and corresponding parameters, it first tries to improve the structure and then as soon as an improved structure is found, it fine tunes its parameters. It then goes back to improving the structure again and, fine tunes the structure and parameters. This loop continues until a satisfactory solution is found or a time limit is reached. The novelty of this paper is in the automatic design of HWNN models by using evolutionary algorithm.

*This research was partially supported by the Natural Science Foundation of China under grant No. 60573065, and the Provincial Science and Technology Development Program of Shandong under grant No. SDSP2004-0720-03.

2 Wavelet Neural Network

In terms of wavelet transformation theory, wavelets are expressed in the following form: $\Psi = \{\Psi_i = |\mathbf{a_i}|^{-\frac{1}{2}}\psi(\frac{\mathbf{x}-\mathbf{b_i}}{\mathbf{a_i}}) : \mathbf{a_i}, \mathbf{b_i} \in R^n, i \in Z\}$, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, $\mathbf{a_i} = (a_{i1}, a_{i2}, \ldots, a_{in})$, $\mathbf{b_i} = (b_{i1}, b_{i2}, \ldots, b_{in})$ are a family of functions generated from one single function $\psi(x)$ by the operation of dilation and translation. $\psi(x)$, which is localized in both the time space and the frequency space, is called a mother wavelet and the parameters a_i and b_i are named the scale and translation parameters and $\mathbf{x}$ represents the inputs to the WNN model.

In the standard form of wavelet neural network, the output of a WNN is given by

$$f(x) = \sum_{i=1}^{M} \omega_i \Psi_i(x) = \sum_{i=1}^{M} \omega_i |a_i|^{-\frac{1}{2}} \psi(\frac{x - b_i}{a_i}) \tag{1}$$

where ψ_i is the wavelet activation function of ith unit of the hidden layer and ω_i is the weight connecting the ith unit of the hidden layer to the output layer unit. Note that for the n-dimensional input space, the multivariate wavelet basis function can be calculated by the tensor product of n single wavelet basis functions as $\psi(x) = \prod_{i=1}^{n} \psi(x_i)$.

3 Hierarchical Wavelet Neural Network

In order to generate and optimize a mutilevel HWNN model, a tree-structural representation is adopted. For generating the tree, a function set F and a terminal instruction set T are described as $S = F \bigcup T = \{+_2, +_3, \ldots, +_N\} \bigcup \{x_1, \ldots, x_n\}$, where $+_i (i = 2, 3, \ldots, N)$ denote non-leaf nodes' instructions and taking i arguments. x_1,x_2,...,x_n are leaf nodes' instructions and taking no other arguments. The output of a non-leaf node is calculated as a wavelet neural network model by Eqn.(1). From this point of view, the instruction $+_i$ is also called a WNN operator with i inputs.

The WNN operator is shown in Figure 1 (left). In this research, the mother wavelet $\psi(x) = -xexp(-\frac{x^2}{2})$ is used for both experiments, and the number of wavelet basis functions in hidden layer is same with the number of inputs, that is, $m = n$.

In the creation process of HWNN tree, if a nonterminal instruction, i.e., $+_i (i = 2, 3, 4, \ldots, N)$ is selected, i real values are randomly generated and used for representing the connection strength between the node $+_i$ and its children. In addition, $2 \times n^2$ adjustable parameters a_i and b_i are randomly created as dilation and translation parameters of the wavelet basis functions. The output of the node $+_i$ can be calculated by using Eqn. (1). The overall output of HWNN tree can be computed from left to right by depth-first method, recursively.

3.1 Tree structure optimization

Finding an optimal or near-optimal HWNN is formulated as a product of evolution. The Extended Compact Genetic Programming (ECGP) [6] is employed

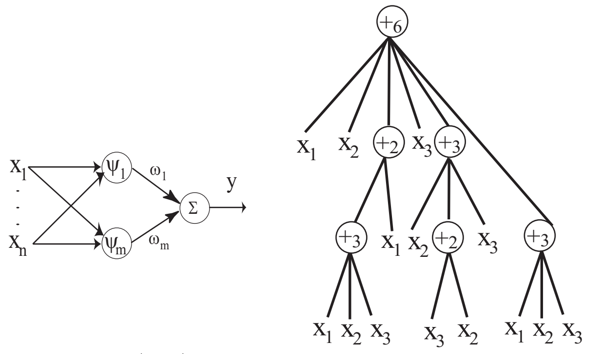

Fig. 1: A WNN operator (left), and a tree-structural representation of a HWNN with instruction set $F \bigcup T = \{+_2, \ldots, +_6\} \bigcup \{x_1, x_2, x_3\}$ (right)

to find an optimal or near-optimal HWNN structure.

ECGP is a direct extension of ECGA to the tree representation which is based on the PIPE prototype tree. In ECGA, Marginal Product Models (MPMs) are used to model the interaction among genes, represented as random variables, given a population of Genetic Algorithm individuals [8]. MPMs are represented as measures of marginal distributions on partitions of random variables. ECGP is based on the PIPE prototype tree, and thus each node in the prototype tree is a random variable. ECGP decomposes or partitions the prototype tree into sub-trees, and the MPM factorizes the joint probability of all nodes of the prototype tree, to a product of marginal distributions on a partition of its sub-trees. A greedy search heuristic is used to find an optimal MPM mode under the framework of minimum encoding inference. ECGP can represent the probability distribution for more than one node at a time. Thus, it extends PIPE in that the interactions among multiple nodes are considered.

3.2 Parameter optimization with DE algorithm

The DE algorithm was first introduced by Storn and Price in 1995 [4] [5]. It resembles the structure of an evolutionary algorithm (EA), but differs from traditional EAs in its generation of new candidate solutions and by its use of a 'greedy' selection scheme. DE works as follows: First, all individuals are randomly initialized and evaluated using the fitness function provided. Afterwards, the following process will be executed as long as the termination condition is not fulfilled: For each individual in the population, an offspring is created using the weighted difference of parent solutions. The offspring replaces the parent if it is fitter. Otherwise, the parent survives and is passed on to the next iteration of the algorithm. In generation k, we denote the population members by x_1^k, x_2^k, $\ldots$, x_N^k. The DE algorithm is given as follows [5]:

1) Set $k = 0$, and randomly generate N points x_1^0, x_2^0, $\ldots$, x_N^0 from search space to form an initial population;

2) For each point $x_i^k (1 \leq i \leq N)$, execute the DE offspring generation scheme to generate an offspring $x_i^(k+1)$;

3) If the given stop criteria is not met, set $k = k + 1$, goto step 2).

The DE offspring generation approach used is given as follows,

1) Choose one point x_d randomly such that $f(x_d) \leq f(x_i^k)$, another two points x_b, x_c randomly from the current population and a subset $S = \{j_1, \ldots, j_m\}$ of the index set $\{1, \ldots, n\}$, while $m < n$ and all j_i mutually different;

2) Generate a trial point $u = (u_1, u_2, \ldots, u_n)$ as follows:
DE Mutation. Generate a temporary point z as follows, $z = (F+0.5)x_d + (F-0.5)x_i + F(x_b - x_c)$, where F is a give control parameter;
DE Crossover. for $j \in S$, u_j is chosen to be z_j; otherwise u_j is chosen a to be $(x_i^k)_j$;

3) If $f(u) \leq f(x_i^k)$, set $x_i^{k+1} = u$; otherwise, set $x_i^{k+1} = x_i^k$.

3.3 Procedure of the general learning algorithm.

The general learning procedure for constructing the HWNN network can be described as follows. (1) Create an initial population randomly (HWNN trees and its corresponding parameters); (2) Structure optimization is achieved by using ECGP algorithm; (3) If a better structure is found, then go to step (4), otherwise go to step (2); (4) Parameter optimization is achieved by the DE algorithm. In this stage, the architecture of HWNN model is fixed, and it is the best tree developed during the end of run of the structure search. (5) If the maximum number of local search is reached, or no better parameter vector is found for a significantly long time then go to step (6); otherwise go to step (4); (6) If satisfactory solution is found, then the algorithm is stopped; otherwise go to step (2).

4 Simulation studies

In this work, the fitness function used for the ECGP and PSO is given by mean square error, $MSE = \frac{1}{P}\sum_{j=1}^{P}(y_1^j - y_2^j)^2$, and/or root mean squared error, $RMSE = \sqrt{\frac{1}{P}\sum_{j=1}^{P}(y_1^j - y_2^j)^2}$, where P is the total number of samples, y_1^j and y_2^j are the actual time-series and the HWNN model output of j-th sample.

4.1 Prediction of Mackey-Glass time-series

The mackey-Glass chaotic time series is generated from the following equation:

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1 + x^{10}(t-\tau)} - bx(t). \tag{2}$$

Where $\tau > 17$, the equation shows chaotic behavior.

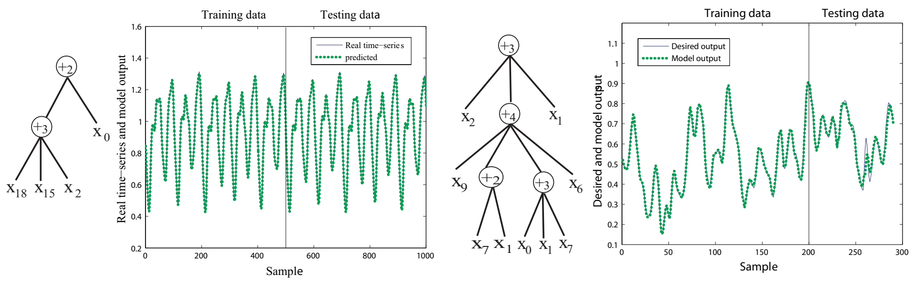

Fig. 2: The evolved HWNN structures and actual time-series and predicted one: left for Mackey-Glass time-series and right for Jenkins-Box data.

Model	BPNN	ANFIS [7]	FNT[2]	HWNN
RMSE	0.02	0.007	0.0027	0.0043

Table 1: Comparison of prediction error for Mackey-Glass time-series.

We predicted $x(t+6)$ by using the inputs variables $x(t)$, $x(t-1)$, …, $x(t-18)$. 1000 sample points were used in our study. The first 500 data pairs of the series were used as training data, while the remaining 500 were used to validate the model identified. The used instruction sets to create an optimal HWNN model is $S = F \bigcup T = \{+_2, +_3, +_4\} \bigcup \{x_0, x_1, \ldots, x_{18}\}$. Where $x_i (i = 0, 1, \ldots, 18)$ denotes $x(t)$, $x(t-1)$, … $x(t-18)$. After 12 generations of the evolution, an optimal HWNN model was obtained with RMSE 0.0045. The RMSE value for validation data set is 0.0043. The evolved HWNN and the actual time-series data, the output of HWNN model are shown in Figure 2 (left). From the evolved HWNN tree, it can be seen that the optimal inputs variables for constructing a HWNN model are: $x(t)$, $x(t-2)$, $x(t-15)$ and $x(t-18)$. A comparison result of different methods for forecasting Mackey-Glass data is shown in Table 1.

4.2 Prediction of Jenkins-Box time-series

The gas furnace data (series J) of Box and Jenkins (1970) is well known and frequently used as a benchmark example for testing identification and prediction algorithms. For this simulation, 10 inputs variables are used for constructing a HWNN model. The proper time-lags for constructing a HWNN model are finally determined by an evolutionary procedure.

The used instruction sets to create an optimal HWNN model is $S = F \bigcup T = \{+_2, \ldots, +_4\} \bigcup \{x_0, x_1, \ldots, x_9\}$. Where $x_i (i = 0, 1, \ldots, 9)$ denotes $y(t-1)$, $y(t-2)$, $y(t-3)$, $y(t-4)$, and $u(t-1)$, $u(t-2)$, $u(t-3)$, $u(t-4)$, $u(t-5)$ and $u(t-6)$, respectively. After 21 generations of evolution, the optimal HWNN model was obtained with MSE 0.00021. The MSE value for validation data set is 0.00025. The evolved HWNN and the actual time-series data, the output of HWNN model are shown in Figure 2 (right). From the evolved HWNN tree, it

Model	BPNN	ANFIS [7]	FNT[2]	HWNN
MSE	0.009	0.0073	0.00066	0.00025

Table 2: Comparison of prediction errors for the gas furnace data.

can be seen that the optimal inputs variables for constructing a HWNN model are: $u(t{-}3)$, $u(t{-}4)$, $u(t{-}6)$, $y(t{-}1)$, $y(t{-}2)$ and $y(t{-}3)$. It should be noted that the HWNN model with proper selected input variables has accurate precision and good generalization ability. A comparison result for different methods for forecasting Jenkins-Box data is shown in Table 2.

5 Conclusions

Based on a novel representation and computational model, a framework for evolving the HWNN was proposed in this paper. The hierarchical architecture and inputs selection method of the HWNN were accomplished using ECGP algorithm, and the free parameters embedded in the HWNN model were optimized using DE algorithm. Preliminary research results reveal that the evolved HWNN models are effective for function approximation problems. Our future work will concentrate on improving the convergence speed of the proposed method by parallel implementation of the algorithm and by applying the proposed approach for some real world applications.

References

[1] Y. Chen, B. Yang, and J. Dong, Nonlinear System Modeling via Optimal Design of Neural Trees, *International Journal of Neural Systems*, 14:125-137, 2004.

[2] Y. Chen, B. Yang, J. Dong and A. Abraham, Time-series Forecasting using Flexible Neural Tree Model, *Information Science*, 174:219-235, 2005.

[3] Y. Chen, B. Yang, J. Dong, Automatic Design of Hierarchical TS-FS Models using Ant Programming and PSO algorithm, Lecture Notes Computer Science 3192, pages 285-294, 2004.

[4] R. Storn, and K. Price, Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, International Computer Science Institute, Berkley, 1995.

[5] K. Price, Differential Evolution vs. the Functions of the 2nd ICEO. In proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97), Indianapolis, USA, pages 153-157, 1997.

[6] K. Sastry and D.E. Goldberg, Probabilistic model building and competent genetic programming, In R. L. Riolo and B. Worzel, editors, Genetic Programming Theory and Practise, pages 205-220, Kluwer, 2003.

[7] J.-S.R. Jang, C.-T. Sun and E. Mizutani, Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence, Upper saddle River, NJ:prentice-Hall (1997).

[8] G. Harik, F. Lobo, and D.E. Goldberg, The compact genetic algorithm, IlliGAL Report No. 97006, pages 523-528, 1998.