Appears in Mason, C. (Ed.) The First International Workshop on AI and the Environment, International Joint Conference on Artificial Intelligence, Montreal, Canada, 1995.

Water Pollution Prediction with Evolutionary Neural Trees*

Byoung-Tak Zhang, Peter Ohm, and Heinz Mühlenbein

German National Research Center for Computer Science (GMD)

Schloss Birlinghoven, D-53754 Sankt Augustin, Germany

E-mail: {zhang, ohm, muehlenbein}@borneo.gmd.de

Abstract

An evolutionary learning method for modeling and prediction of complex systems is described and applied to an environmental system. The method is based on tree-structured neural networks whose node type, weight, size and topology are dynamically adapted by genetic algorithms. Since the genetic algorithm used for training does not require error derivatives, a wide range of neural models can be identified. The application of this method to the prediction of water pollution shows comparable results to those achieved by well-engineered, conventional system-identification methods.

*This research was supported in part by the Real-World Computing Program under the project SIFOGA.

Appears in Mason, C. (Ed.) The First International Workshop on AI and the Environment, International Joint Conference on Artificial Intelligence, Montreal, Canada, 1995.

1 Introduction

Modeling and predicting the behaviors of many environmental or ecological systems is difficult because these systems are often complex. They are generally characterized by a large number of variables, parameters, interactions, and limited amounts of collected data.

In this paper we present an evolutionary method for learning such models. Although several genetic algorithms [1] have been proposed for constructing neural networks (see, for example, [2] and [8] for a review of recent developments), our method is different from them in that we use a tree representation of the neural network, called neural trees. Unlike most conventional neural models, neural trees employ different types of neurons in a single network. The set of different types is defined by the application domain, and the specific type of each unit is determined during the evolutionary learning process. Any arbitrary units can be employed since the training algorithm we use, i.e., the breeder genetic algorithm [7], makes no assumptions as to the differentiability of the activation function.

The structure and size of the network is also automatically adapted during the evolutionary learning process. With sigma and pi units, for example, the method can build a higher-order functional structure of partially connected polynomial units, resembling but different from the architecture constructed by the Group Method of Data Handling (GMDH) [5] and its descendants. The explicit use of product neurons has been very useful for solving problems which are difficult for multilayer perceptrons [4, 11].

In section 2 we briefly describe the tree encoding scheme and the genetic algorithm for evolving problem-specific neural trees. Section 3 reports the application results on the prediction of an environmental system. Implications of current work are discussed in section 4.

Appears in Mason, C. (Ed.) The First International Workshop on AI and the Environment, International Joint Conference on Artificial Intelligence, Montreal, Canada, 1995.

2 Evolutionary Neural Trees

A neural tree consists of a number of artificial neurons connected with weights in a tree structure. The leaves of the tree are elements of the terminal set X of n variables, $X = \{x_1, x_2, \cdots, x_n\}$. The root node of the tree is called the output unit. All the nodes except the input units are non-terminal units. Each non-terminal node i is characterized by the unit type u_i, the squashing function f_i, the receptive field $R(i)$ and the weight vector $\mathbf{w}_i$. $R(i)$ is the index set of incoming units to unit i and can be different from unit to unit.

In the experiments described in the next section, we use sigma (S) and pi (P) units, i.e., $u_i \in \{S, P\}$, mixed into the same network. The sigma units compute the weighted sum of the input values y_j from other units, $\sum_{j \in R(i)} w_{ij} y_j$, and the pi units compute the product of their weighted inputs, $\prod_{j \in R(i)} w_{ij} y_j$. Each squashing function is either a sigmoid function, $f(x) = \frac{1}{1+e^{-x}}$, or a threshold function, $f(x) = +1$ if $x > 0$ and $f(x) = -1$ otherwise.

An instance of the sigma-pi neural tree is shown in Figure 1. Notice that although the set of neuron types and external inputs is finite, any arbitrarily large trees can be generated from them. This encoding scheme can represent any feed-forward network with local receptive fields and direct connections between non-neighboring layers (see [9] for more details) and thus extends the tree representation used in [6]. This is contrasted with the more commonly used perceptron architecture of fully connected feedforward networks.

For the construction of neural models, we maintain a population $\mathcal{A}$ consisting of M individuals of variable size. Each individual A_i is a neural network represented as neural trees. The initial population $\mathcal{A}(0)$ is created at random. In each generation g, the fitness values $F_i(g)$ of networks are evaluated and the upper $\tau\%$ are selected to be in the mating pool $\mathcal{B}(g)$. The next generation $\mathcal{A}(g+1)$ of M individuals are then created by exchanging subtrees and thereby adapting the size and shape of the network. Mutation changes the node type and the index of incoming units. The best individual is always retained in the next generation so that the

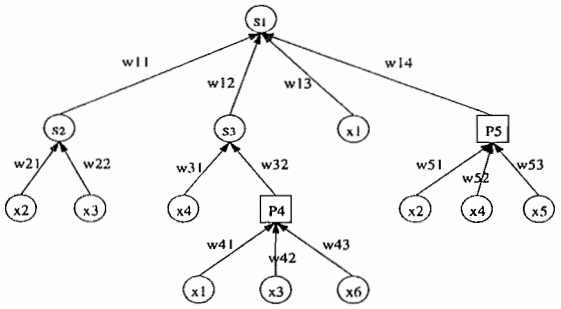

Figure 1: *Tree representation of a sigma-pi neural network with six inputs and one output. Each unit has a local receptive field.*

population performance does not descrease as generation goes on (elitist strategy).

Between generations the network weights are adapted by a stochastic hill-climbing search. This search method is based on the breeder genetic algorithm [7], in which the step size Δw is determined with a random value $\epsilon \in [0, 1]$:

$$\Delta w \quad = \quad R \cdot 2^{-\epsilon \cdot K}, \tag{1}$$

where R and K are constants specifying the range and slope of the exponential curve. This method proved very robust for a wide range of parameter-optimization problems.

The fitness F_i of the individuals A_i is defined as

$$F_i = F(D|A_i) = \frac{E(D|A_i)}{m \cdot N} + \frac{C(A_i)}{N \cdot C_{max}}, \tag{2}$$

where m is the number of outputs, and N is the number of training examples. The first term expresses the error penalty $E(D|A_i)$ for the training set. The second term penalizes the complexity $C(A_i)$ of the network. This evaluation measure prefers simple networks to complex ones and turned out to be important for achieving good generalization [9, 11].

Appears in Mason, C. (Ed.) The First International Workshop on AI and the Environment, International Joint Conference on Artificial Intelligence, Montreal, Canada, 1995.

3 Predicting an Environmental Time Series

In recent years, system identification and time-series prediction have received much attention as promising application areas of neural networks. The task we have studied involves an environmental system in the Sangamon River, Illinois [3]. Our objective here is to predict nitrate levels a week ahead in the watersheds of the river from the previous values. The original study in the literature [3] also aims at giving some indication of the biochemical and physical relationships among the variables and of the controllability of the system.

The training data is based on the nitrate-nitrogen levels during the period from January 1, 1970, to December 31, 1971, graphed in Figure 2 (left). The sampling interval is one week. The training set is generated from this series using a time lag of 4. The initial population of the genetic algorithm was created by randomly generating neural trees with a branching factor up to four and maximum depth of four also. 50% of the units were randomly chosen as sigma units and the rest as pi units. During evolution, however, sigma units usually survived more often than the pi units did. Using the weight interval of $[-10, +10]$, the best solutions after 300 generations contained on average 10 hidden units in three layers.

An example of one-step ahead prediction performance for the training data is shown in Figure 2 (right) whose mean square error is 0.0104. To test the predictive accuracy of the neural tree models constructed by the evolutionary algorithm, unseen data for the same watershed for 1972 was used. The measured and predicted outputs for this test data are plotted in Figure 3. As can be seen in the figure, the nitrate levels for the following week was predicted relatively well, considering the sparseness of the training data. This result is as good as that obtained by the well-engineered GMDH algorithm [3].

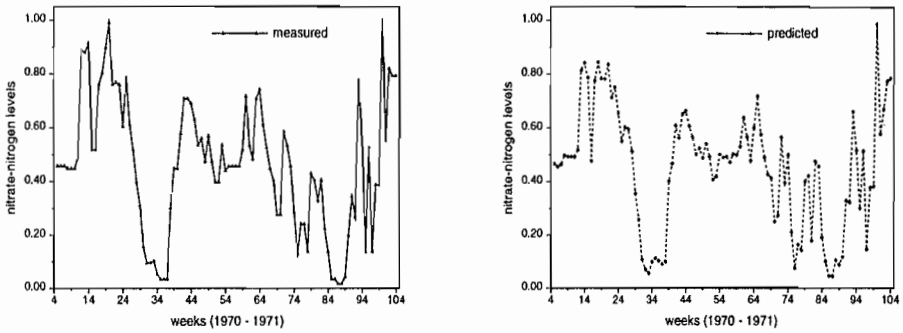

Figure 2: *Performance for the training data: (left) measured, (right) one-step ahead prediction.*

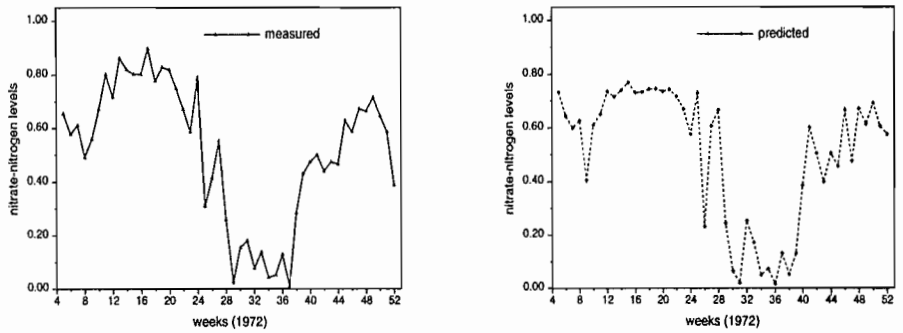

Figure 3: *Performance for the unseen data: (left) measured, (right) one-step ahead prediction.*

Appears in Mason, C. (Ed.) The First International Workshop on AI and the Environment, International Joint Conference on Artificial Intelligence, Montreal, Canada, 1995.

4 Concluding Remarks

Most existing evolutionary methods for neural-network optimization represent the network as a linear string or matrix of fixed size whose connectivity and weights are changed in isolation. In contrast, our method is based on a tree-structured representation of the network, which allows dynamic adaptation of the neuron type, weight, and topology in a natural way using genetic operators.

The neural tree representation takes advantages of both the direct encoding scheme (little effort in decoding) and the indirect encoding scheme (easy to build modular structures) and is able to generate a very general class of feed-forward networks of partially connected heterogeneous neurons. One primary difficulty with this approach is that the network size may become very large without any improvement in generalization performance. The use of complexity penalty in fitness evaluation was very useful for solving this problem, as was indicated by the relatively good generalization performance in the water pollution experiments.

In contrast to conventional learning algorithms for neural networks, the evolutionary learning method described in this paper makes relatively few assumptions as to the architecture space in which the models for data are constructed. This method is especially effective in identifying the important variables and structures of the systems whose functional structures are unknown or ill-defined. Examples of the most promising application areas of the evolutionary neural trees include prediction, monitoring and diagnosis of complex systems, such as environmental processes.

References

[1] T. Bäck and H.-P. Schwefel, An overview of evolutionary algorithms for parameter optimization, *Evolutionary Computation*, 1(1):1-23, 1993.

[2] K. Balakrishnan and V. Honavar, Evolutionary design of neural architectures, CS-TR-95-01, AI Lab, Dept. of Computer Science, Iowa State University, Jan-

uary 1995.

[3] J. J. Duffy and M. A. Franklin, A learning identification algorithm and its application to an environmental system, *IEEE Trans. on Sys. Man and Cyb.*, 5(2):226-240, 1975.

[4] C. L. Giles and T. Maxwell, Learning, invariance, and generalization in high-order neural networks, *Applied Optics*, 26(23):4972-4978, 1987.

[5] A. G. Ivakhnenko, Polynomial theory of complex systems, *IEEE Trans. on Sys. Man, and Cyb.*, 1(4):364-378, 1971.

[6] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[7] H. Mühlenbein and D. Schierkamp-Voosen, Predictive models for the breeder genetic algorithm I: Continuous parameter optimization, *Evolutionary Computation*, 1(1):25-49, 1993.

[8] J. D. Schaffer, D. Whitley, and L. J. Eshelman, "Combinations of genetic algorithms and neural networks: A survey of the state of the art," in *Proc. Int. Workshop on Combinations of Genetic Algorithms and Neural Networks*, IEEE, 1992, pp. 1-37.

[9] B. T. Zhang and H. Mühlenbein, Evolving optimal neural networks using genetic algorithms with Occam's razor, *Complex Systems*, 7(3):199-220, 1993.

[10] B. T. Zhang and H. Mühlenbein, "Synthesis of sigma-pi neural networks by the breeder genetic programming," in *Proc. ICEC-94*, IEEE World Congress on Computational Intelligence, IEEE Computer Society Press, 1994, pp. 318–323.

[11] B. T. Zhang, Effects of Occam's razor in evolving sigma-pi neural networks, in *Lecture Notes in Computer Science 866*, Springer-Verlag, 1994, pp. 462-471.