Evolutionary Design of Neural Trees for Heart Rate Prediction

Byoung-Tak Zhang[1] **Je-Gun Joung**[2]

[1] Dept of Computer Engineering, Seoul National University
Seoul 151-742, Korea. E-mail: btzhang@comp.snu.ac.kr
[2] Dept of Computer Science and Engineering, Konkuk University
Seoul 143-701, Korea. E-mail: jgjoung@ai.konkuk.ac.kr

Keywords: Neural network design, Evolutionary computation, *Neural tree induction, Time series prediction, Heart rate data.*

Abstract

Some classes of neural networks are known as universal function approximators. However, their training efficiency and generalization performance depend highly on the structure which is usually determined by a human designer. In this paper we present an evolutionary computation method for automating the neural network design process. We represent networks as tree structures, called neural trees, in genotype and apply genetic operators to evolve problem-dependent network structures and their weights. Experimental results are provided on the prediction of a heart rate time-series by evolving sigma-pi neural trees.

1 Introduction

Recently, several evolutionary methods have been proposed for constructing and training neural networks. Existing representation methods for neural networks can be roughly divided into two categories [1]: direct and indirect encoding [1]. Direct encodings use a fixed structure, such as connection matrix or bitstrings that precisely specifies the architecture of the corresponding neural network. This encoding scheme requires little effort to decode. However, matrix structures have limited flexibility in expressing topologies of the network structure with variable layers. Bitstrings are not flexible enough to represent various partial connectivity without further annotation. Genetic operators need to be applied carefully to preserve the topological constraints of networks.

Indirect encoding schemes use rewrite rules to specify a set of construction rules that are recursively applied to yield the phenotype. Examples include graph generation grammars and cellular encoding. This approach is interesting in that it simulates in some sense the developmental process. Subtree crossover applies well to these representations. In addition, experimental evidence has shown that the cellular encoding scheme is effective in evolving modular structures consisting of similar substructures. However, the grammatical encoding does not seem appropriate for exploring a huge number of partial interaction possibilities as required in our application. In addition, grammatical encoding requires execution of rewrite-rules for every conversion from genotype to phenotype. This makes network training an expensive phase since training of neural networks requires a large number of evaluations and each evaluation needs a separate decoding.

In this paper we present an alternative representation scheme, called neural trees [9, 10]. 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 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.

The neural tree representation combines the advantages of direct and indirect encoding schemes. It is

powerful and flexible in expressing a broad class of neural architectures. The representation is decoding-efficient and convenient for genetic operations. 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. The explicit use of product neurons has been very useful for solving problems which are difficult for multilayer perceptrons [2].

The paper is organized as follows. Section 2 describes the neural tree encoding scheme in more detail. Section 3 is devoted to the description of the heart rate time-series data and the evolutionary computing method for designing neural trees for the prediction of the data. Section 4 reports the experimental results. Sectoin 5 contains conclusions.

2 Neural Trees

Let $\mathcal{NT}(d, b)$ denote the set of all possible trees of maximum depth d and maximum b branches for each node. The nonterminal nodes represent neural units and the neuron type is an element of the basis function set $\mathcal{F} = \{\text{neuron types}\}$. Each terminal node is labeled with an element from the terminal set $\mathcal{T} = \{x_1, x_2, ..., x_n\}$, where x_i is the ith component of the external input $\mathbf{x}$. Each link (j, i) represents a directed connection from node j to node i and is associated with a value w_{ij}, called the synaptic weight. The members of $\mathcal{NT}(d, b)$ are referred to as neural trees. In case of $\mathcal{F} = \{\Sigma, \Pi\}$, the trees are specifically called sigma-pi neural trees. The root node is also called the output unit and the terminal nodes are called input units. Nodes that are not input or output units are hidden units. The layer of a node is defined as the longest path length to any terminal node of its subtree.

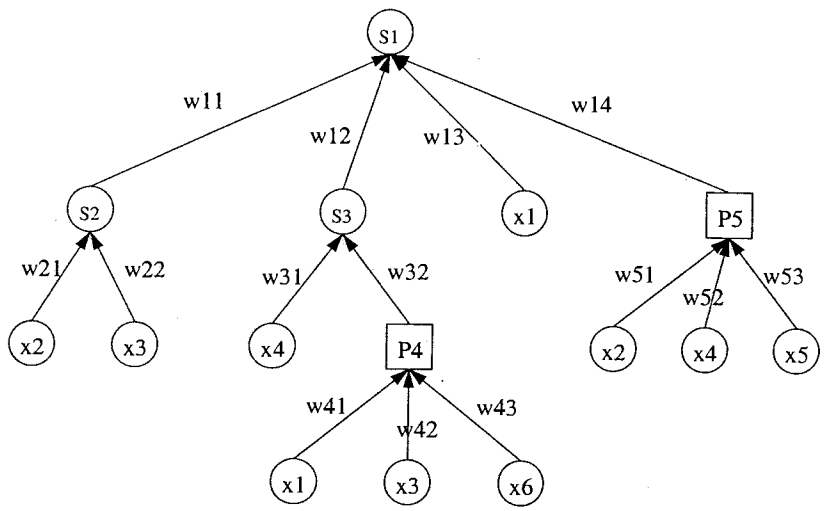

Figure 1: An example of neural tree structure.

Different neuron types are distinguished in the way of computing net inputs. For the evolution of higher-order networks, we consider two types of units. Sigma units compute the sum of weighted inputs from the lower layer:

$$net_i = \sum_j w_{ij} y_j \tag{1}$$

where y_j are the inputs to the ith neuron. Pi units compute the product of weighted inputs from the lower layer:

$$net_i = \prod_j w_{ij} y_j \tag{2}$$

where y_j are the inputs to i. The output of a neuron is computed either by the threshold response function

$$y_i = \sigma(net_i) = \begin{cases} 1 & : \quad net_i \geq 0 \\ -1 & : \quad net_i < 0 \end{cases} \tag{3}$$

or the sigmoid transfer function

$$y_i = f(net_i) = \frac{1}{1 + e^{-net_i}} \tag{4}$$

where net_i is the net input to the unit computed by Eqn. 1 or Eqn. 2.

A higher-order network with m output units can be represented by m sigma-pi neural trees. That is, the genotype A_i of ith individual in our evolutionary framework consists of m neural trees:

The neural tree representation does not restrict the functionality since any feedforward network can be represented with a forest of neural trees:

$$A_i = (A_{i,1}, A_{i,2}, ..., A_{i,m}) \quad \forall k \in \{1, ..., m\}, \; A_{i,k} \in \mathcal{NT}(d, b) \tag{5}$$

The connections between input units to arbitrary units in the network is also possible since input units can appear more than once in the neural tree representation. The output of one unit can be used as input to more than one unit. The duplication does not necessarily mean more space requirements in trees than network representations since frequently-used fit submodules can be stored and multiply reused. This leads to the construction of modular structures and reduces memory requirements for representing the population [10].

Neural trees do not require decoding for their fitness evaluation. Training and evaluation of fitness can be performed directly on the genotype since both the genotype and phenotype are equivalent. Since subtree crossover used in genetic programming [3] applies without modification to this representation, we can use genetic programming as the main evolutionary engine.

3 Predicting Heart Rates by Evolving Neural Trees

Physiological systems are often considered to have machinelike properties, and a common objective in physiology is to discover how some system senses its current state and uses this information to respond automatically [6]. For example, it has long been known that heart rate is regulated by the baro-reflex, in which specialized nerves in the aorta and other blood vessels sense the blood pressure and convey this information to the brainstem, which in turn sends signals to the pacemaker region of the heart that determines the heart rate. The physiological signals are ordinarily nonstationary due, for example, to the difficulty of controlling voluntary behavior or to circadian influences. Furthermore, a variety of nonlinearities abound in the interaction between the components of physiological systems.

We used a multivariate physiological data recorded from a patient in the sleep laboratory. The signals were recorded from a 49-year-old male. He had been tentatively diagnosed as suffering from sleep apnea, a potentially life-threatening disorder in which the subject stops breathing during sleep. It was a part of the data used in the Santa Fe Institute Time Series Prediction and Analysis Competition [7]. This data set provides simultaneous measurements using the extra information to learn about how potentially variables interact. If one-variable time series is deterministic, there exists a scalar d (which is called the embedding dimension) and a function f such that for every $t > d$:

$$x_t \quad = \quad f(x_{t-1}, x_{t-2}, ..., x_{t-d}). \tag{6}$$

We have used neural trees to model and predict the heart rate time series. For the construction of neural models, we maintain a population $\mathcal{A}$ consisting of M individuals of variable size:

$$\mathcal{A}(g) = (A_1, A_2, ..., A_M). \tag{7}$$

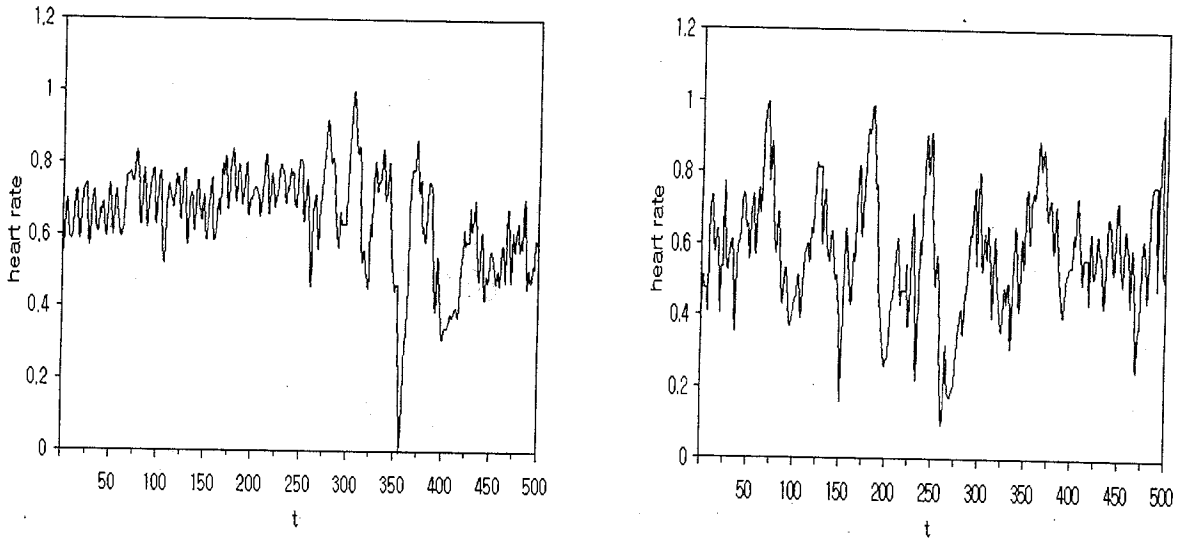

Figure 2: The heart rate time series used for training (left) and test (right).

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 population performance does not decrease 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 [4], in which the step size Δw is determined with a random value $\epsilon \in [0, 1]$:

$$\Delta w \quad = \quad R \cdot 2^{-\epsilon \cdot K}, \tag{8}$$

where R and K are constants specifying the range and slope of the exponential curve. In the experiments, the values were $R = 2$ and $K = 3$. 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

$$F_i \quad = \quad F(D|A_i) = \frac{E(D|A_i)}{m \cdot N} + \frac{C(A_i)}{N \cdot C_{best}}, \tag{9}$$

where m is the number of outputs, N is the number of training examples, and C_{best} is the complexity of the best individual at generation $g-1$. 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, defined as the sum of the units and the number of layers in the tree. This evaluation measure prefers parsimonious networks to complex ones and turned out to be important for achieving good generalization [9].

4 Experimental Results

Since we do not know the embedding dimension for the time series, we varied the number of inputs for the prediction. In particular, we experimented with two different embedding dimensions, $d = 3, 5$:

- One-step ahead prediction from three inputs

$$x_t \quad = \quad f(x_{t-1}, x_{t-2}, x_{t-3}) \tag{10}$$

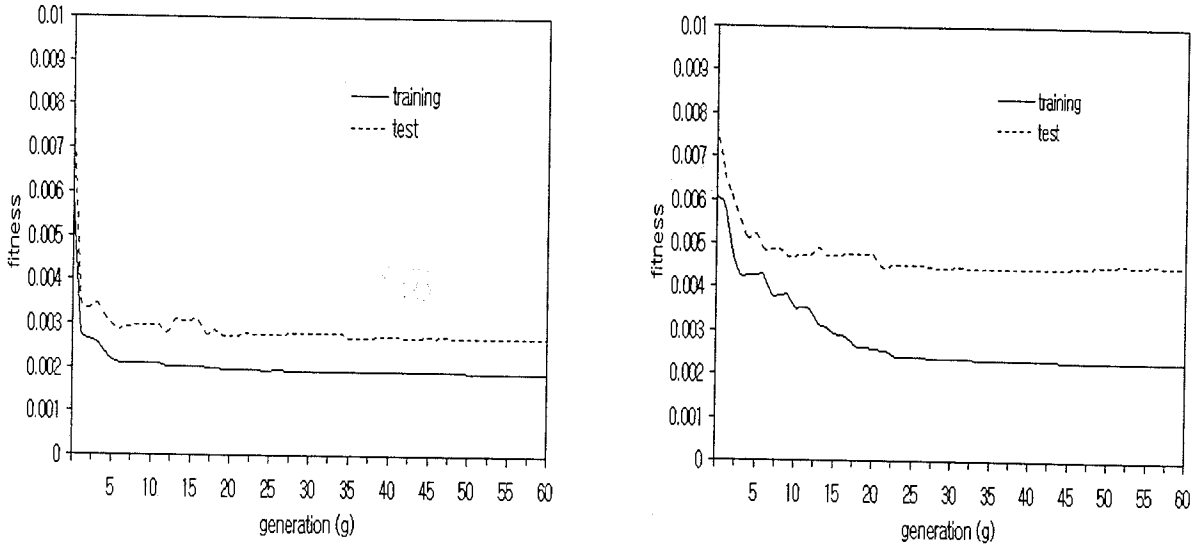

Figure 3: Fitness vs. generation plots for heart rate prediction: (left) three inputs and (right) five inputs.

- One-step ahead prediction from five inputs

$$x_t = f(x_{t-1}, x_{t-2}, x_{t-3}, x_{t-4}, x_{t-5}) \tag{11}$$

A time series of 500 points was used to generate the training set. The next 500 points of time series were used for generating the test set. The algorithm parameters used for the experiments are summerized in Table 1. Each run consists of 60 generations with a population size of 200.

Algorithm parameters	Values used
population size	200
max generation	60
crossover rate	0.95
mutation rate	0.1
no. of iterations	100
training set size	500
test set size	500

Table 1: Parameter values used in the experiments.

Figure 3 shows the change of fitness values of the best-of-generation neural trees. A fast reduction of errors can be observed during early generations. This seems attributed to the fact that the value x_{t-1} is usually important for the prediction of x_t and the evolutionary algorithm can find this variable in an early generation. Other authors, for example [5], also have found similar phenomenon in one-step ahead predictions of time series. The rest of the evolution attempts to find a better neural tree structure by both structural and parametric modification.

Figure 4 plots the evolution of tree size during the run. The graphs show a flexible change of network size. Close relationship between the change of the network structure and the change of the fitness value was observed. The use of complexity penalty in the fitness function was observed very useful; without it the network size usually grows without bound, resulting in poor prediction performance for unseen time series.

Figures 5 and 6 show the one-step ahead prediction results for the training and test data, respectively. For comparison the figures also contain plots for the difference between the measured and predicted values.

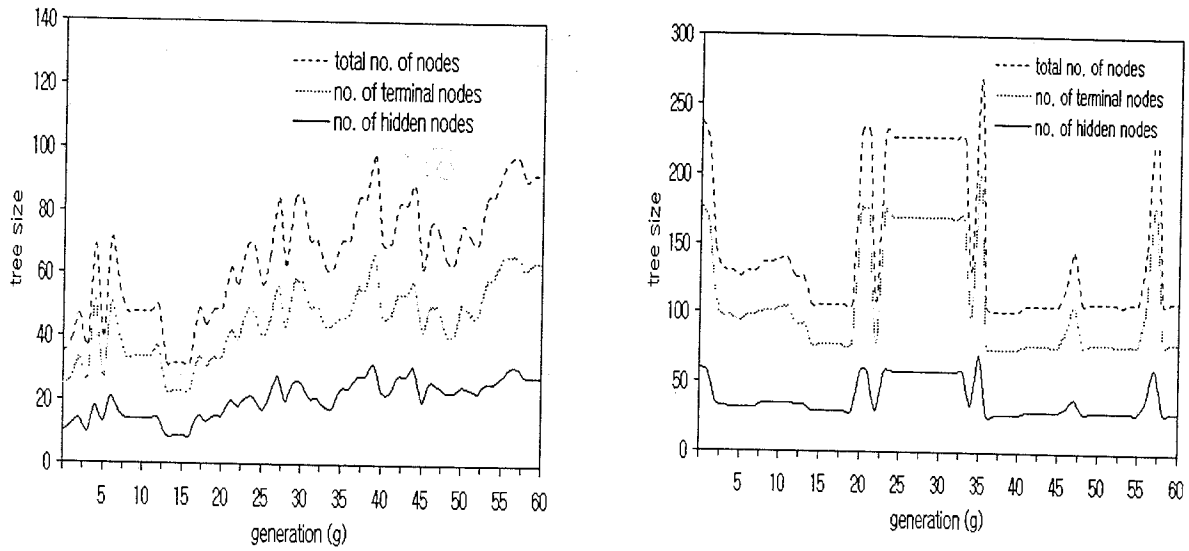

Figure 4: Tree size vs. generation plots for heart rate prediction: (left) three inputs and (right) five inputs.

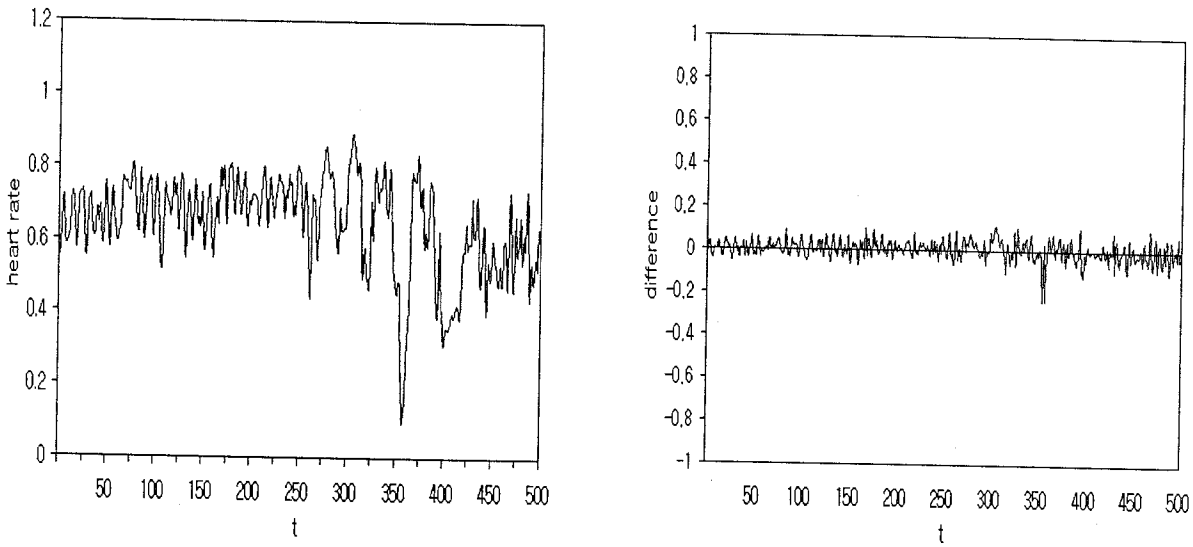

Figure 5: Performance for the training data: one-step ahead prediction from three inputs.

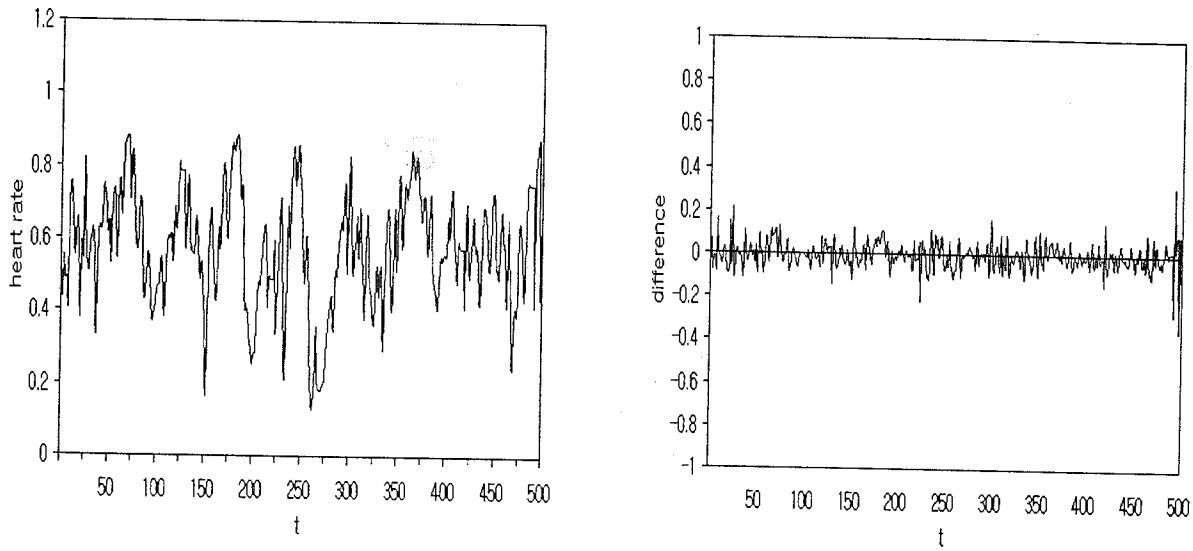

Figure 6: Performance for the test data: one-step ahead prediction from three inputs.

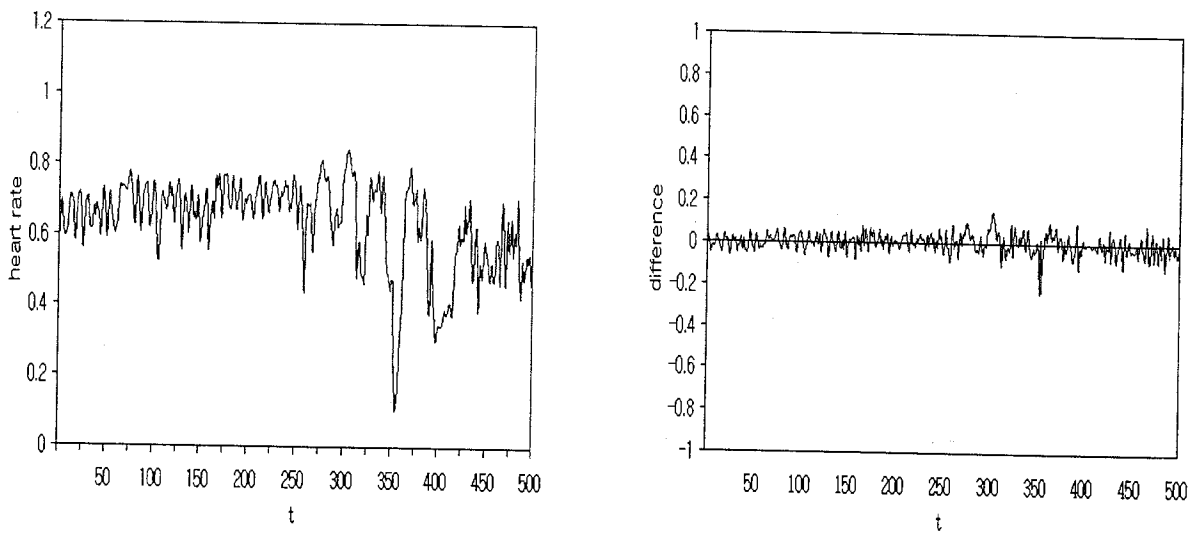

Figure 7: Performance for the training data: one-step ahead prediction from 5 inputs.

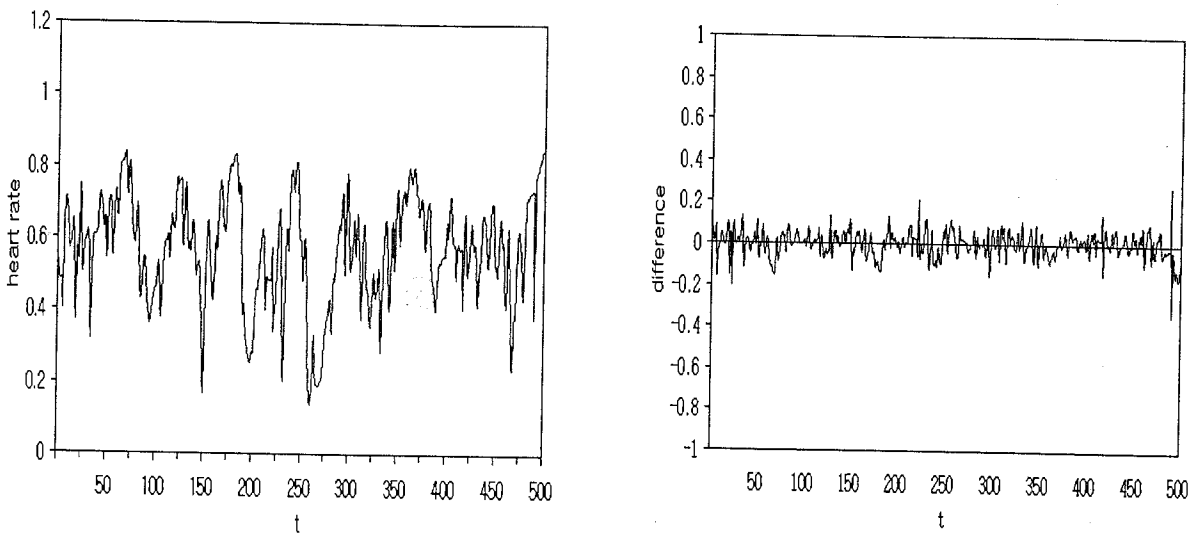

Figure 8: Performance for the test data: one-step ahead prediction from 5 inputs.

Considering the nonlinearity and nonstationarity of the heart rate, the method finds a reasonable model of the underlying structure within a few dozen generations.

Figures 7 and 8 show the results for the five inputs rather than three. Several runs have shown the tendency that using five points as input results in poorer performance than using three inputs. This suggests a possibility of the embedding dimension of this problem less than five. We did not try to analyze the underlying structure of this data.

5 Conclusions

We presented an evolutionary computation method for evolving neural trees and demonstrated its performance on a real-life physiological data. 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.

Since evolutionary search does not require restrictive assumptions such as differentiability of neural activation functions, the method can be used to explore a wide range of novel neural architectures. The control of tree size was found important to make the approach practical faced with real-life data.

The physiological data seems to provide an interesting benchmark for the test of neural modeling techniques in several reasons. First, they contain much noise and information loss caused during the preprocessing of the signals. Second, the signals are ordinarily nonstationary due, for example, to the difficulty of controlling behavior (e.g., respirator or activity) or to circadian influences. Third, a variety of nonlinearities abound in the interaction between the components of physiological systems. Experiments are in progress to achieve a prediction accuracy in multi-step ahead prediction problems which is comparable to that of one-step ahead prediction.

Acknowledgements

This research was supported in part by the Korea Science and Engineering Foundation (KOSEF) under grant number 961-0901-001-2.

References

[1] Balakrishnan, K. and Honavar, V., 1995, *Evolutionary design of neural architectures*, CS-TR-95-01, AI Lab, Dept. of Computer Science, Iowa State University.

[2] Durbin, R. and Rumelhart, D. E., 1989, Product units: a computationally powerful and biologically plausible extension to backpropagation networks, *Neural Computation*, **1**, 133–142.

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

[4] Mühlenbein, H. and Schlierkamp-Voosen, D. 1994, The science of breeding and its application to the breeder genetic algorithm, *Evolutionary Computation*, **1**(4), 335-360.

[5] Mulloy, B. S., Riolo, R. L., and Savit, R. S. 1996, Dynamics of genetic programming and chaotic time series prediction, *Proc. First Annual Conf. on Genetic Programming*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and E. L. Riolo (eds.), Cambridge, MA: MIT Press, pp. 166-174.

[6] Rigney, D. R. et al., 1994, Multi-channel physiological data: description and analysis (data set B), *Time Series Prediction: Forecasting the Future and Understanding the Past*, Addison-Wesley.

[7] Weigend, A. S. and Gershenfeld, N. A. (Eds.), 1994, *Time Series Prediction: Forecasting the Future and Understanding the Past*, Addison-Wesley.

[8] Whitley, D. Starkweather, T. and Bogart, C. 1990, Genetic algorithms and neural networks: optimizing connections and connectivity, *Parallel Computing*, **14**, 347–361.

[9] Zhang, B. T. and Mühlenbein, H. 1995, Balancing accuracy and parsimony in genetic programming, *Evolutionary Computation*, **3**, 17–38.

[10] Zhang, B. T. Ohm, P. and Mühlenbein, H. 1997, Evolutionary induction of sparse neural trees, *Evolutionary Computation*, **5**(3), to appear.