Bayesian Evolutionary Algorithms for Evolving Neural Tree Models of Time Series Data

Dong-Yeon Cho
Artificial Intelligence Lab (SCAI)
School of Computer Science and Engineering
Seoul National University
Seoul 151-742, Korea
dycho@scai.snu.ac.kr

Byoung-Tak Zhang
Artificial Intelligence Lab (SCAI)
School of Computer Science and Engineering
Seoul National University
Seoul 151-742, Korea
btzhang@scai.snu.ac.kr

Abstract- Model induction plays an important role in many fields of science and engineering to analyze data. Specifically, the performance of time series prediction whose objectives are to find out the dynamics of the underlying process in given data is greatly affected by the model. Bayesian evolutionary algorithms have been proposed as a method for automatic model induction from data. In this paper, we apply Bayesian evolutionary algorithms (BEAs) to evolving neural tree models of time series data. The performances of various BEAs are compared on two time series prediction problems by varying the population size and the type of variation operations. Our experimental results support that population-based BEAs with unlimited crossover find good models more efficiently than single individual BEAs, parallelized individual-based BEAs, and population-based BEAs with limited crossover.

1 Introduction

A time series is a sequence of observations taken sequentially in time [3]. Many time series data sets can be found in our life: changes of the stock index or exchange rate, hourly observed output values of some systems, average temperature of a certain place, and so on. The basic characteristic of a time series is that adjacent values are dependent. Generally speaking, past values in the sequence influence future values. The aim of time series prediction (also called forecasting) is to analyze this dependence which is generally believed to be represented by nonlinear relationships and to find out the model which accurately predicts the evolution of the system [14]. It usually involves searching the structure and parameters of models which minimize the error for the test data as well as the training data. Bayesian inference can be applied to these problems [10]. However, integrations of high dimensional functions should often be calculated to estimate the desired probability. This is the most difficult part in Bayesian inference.

Markov chain Monte Carlo (MCMC) methods [5] have been used to perform Bayesian inference without calculating the complex integration. In this method, samples are drawn from the required distribution by constructing a Markov chain and then averaged to approximate the integration. The previous MCMC methods could only be used to find the fittest parameters of a fixed-structure model. Green [6] extended the application of MCMC to the variable-structure models, for example, searching the optimal number of components in the Gaussian mixture [12] or hidden units of feed-forward neural networks [13] and radial basis function (RBF) networks [1].

Evolutionary computation has been used to find the appropriate structure and parameters of models for data in scientific and engineering communities. Several authors have used evolutionary algorithms to design good models for time series prediction. Harrald and Kamstra [7] used evolutionary programming to evolve single hidden-layer perceptrons for combining forecasts of stock price volatility. Kreutz et al. [9] employ an evolutionary algorithm to the optimization of a mixture of densities model for time series forecasting with dynamical noise and missing data. Angeline [2] presents an evolutionary method that evolves a class of dynamic systems with a form similar to neural networks. Chen and Lu [4] empirically show that the evolutionary design of neural networks is helpful in forecasting foreign exchange rates. Recently, Zhang [16] proposed a Bayesian framework for evolutionary computation and derived specific examples of the Bayesian evolutionary algorithms (BEAs) for solving model induction problems.

In this paper, we apply BEAs to time series prediction by using neural trees. Neural tree models represent neural networks as a tree structure [15]. They have heterogeneous neuron types in a single network and the connectivity of the neurons are irregular and sparse. We present a new method to build the neural tree model with the appropriate structure and weights for given data by BEAs.

The generality and uniformity of the Bayesian evolutionary framework allows existing simulation-based methods to be viewed as special cases of BEAs. BEAs can be implemented in various ways. For example, MCMC methods can be regarded as the BEA whose population size is one and thus there is no information exchange among the individuals by crossover, while general evolutionary algorithms can be regarded as a BEA with many individuals exchanging information by crossover. In this paper, we study the performance of BEAs on time series prediction in terms of the population size, the types of variation operators, and the prior probabilities.

0-7803-6375-2/00/$10.00 ©2000 IEEE.

The paper is organized as follows. In Section 2 we describe the structure of neural trees. Section 3 presents the Bayesian evolutionary algorithms for evolving neural tree models. Section 4 reports the experimental results for the laser and sunspot data. Section 5 summarizes our findings from this study.

2 Neural Tree Models

A neural tree is composed of terminal nodes, nonterminal nodes, and weights of connecting links between two nodes [15]. 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, where node i is parent of node j and node j is child of node i. There is also a value w_{ij} which is associated with each link. In this neural tree, the root node is a output unit and the terminal nodes are input units. The depth of a neural tree d_{max} is defined as the longest path length from the root node to any terminal node of the tree.

Each nonterminal node gets input signals from maximum b_{max} child nodes and has a single output. Different neuron types are distinguished in the way that the net inputs are computed. One of the most popular neuron types is the sigma unit, which computes the sum of weighted inputs from the lower layer by

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

where y_j are the inputs to the ith neuron. Another useful neuron type is the pi unit, which calculates the product of weighted inputs from the lower layer as

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

where y_j are the inputs to i. The output of a neuron is computed by the sigmoid transfer function

$$y_i = f(net_i) = \frac{1}{1 + e^{-net_i}}, \tag{3}$$

where net_i is the net input to the unit computed by Equation (1) or (2).

An instance of the neural tree is shown in Figure 1, where $\mathcal{F} = \{\Sigma, \Pi\}$, $\mathcal{T} = \{x_1, x_2, x_3, x_4\}$, and $b_{max} = 4$.

3 Bayesian Evolutionary Algorithms for Evolving Neural Trees

3.1 Bayesian Inference

Bayes theorem provides a direct method for computing the posterior probability $P(A|D)$ of each model A given the observed training data D [11]. By Bayes theorem, we have

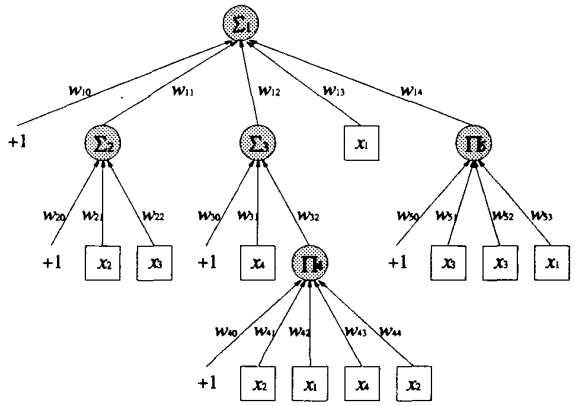

Figure 1: The structure of a neural tree.

$$P(A|D) = \frac{P(D|A)P(A)}{P(D)}, \tag{4}$$

where $P(A)$ is the prior probability for the model, $P(D|A)$ is the likelihood of the model for the data, and $P(D)$ is a normalizing constant and computed as

$$P(D) = \int P(D|A)P(A)dA. \tag{5}$$

With this posterior probability, we can compute the expected value of output for the unknown data $\mathbf{x}$ as follows

$$E[f_A(\mathbf{x})] = \int f_A(\mathbf{x})P(A|D)dA, \tag{6}$$

where f_A is the function implemented by a model A. In most applications, however, we cannot easily evaluate the integration in Equation (6) and numerical calculation is impossible especially for the high dimensional function f_A. This value can be approximated by the Bayesian evolutionary algorithms.

3.2 Defining the Probability Distributions of Neural Trees

To find the fittest model by the BEAs, we first define the probability distributions of neural tree models for data. The posterior probability of a neural tree A is defined as

$$\begin{aligned} P(A|D) \propto P(D|A)P(A) &= P(D|\mathbf{w}, k)P(\mathbf{w}, k) \\ &= P(D|\mathbf{w}, k)P(\mathbf{w}|k)P(k), \end{aligned} \tag{7}$$

where k is the number of nodes in the neural tree (including bias terms) and $\mathbf{w}$ is the weight vector. Given the training data as

$$D = \{(\mathbf{x}_c, y_c)\}_{c=1}^N, \tag{8}$$

the model A can describe many time series problems by the following input-output mapping

$$y_c = f_A(\mathbf{x}_c) + \epsilon, \tag{9}$$

where the noise ϵ is assumed to be zero-mean Gaussian with the standard deviation σ. If we assume that data are independent of each other, then the likelihood of the neural tree can be expressed as follows

$$P(D|\mathbf{w},k) = \prod_{c=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\Big(-\frac{(y_c - f_{(\mathbf{w},k)}(\mathbf{x}_c))^2}{2\sigma^2}\Big)$$

$$= \Big(\frac{1}{\sqrt{2\pi}\sigma}\Big)^N \exp\Big(-\frac{\sum_{c=1}^{N}(y_c - f_{(\mathbf{w},k)}(\mathbf{x}_c))^2}{2\sigma^2}\Big). \quad (10)$$

We define the following prior probability for weights of the neural tree

$$P(\mathbf{w}|k) = \prod_{j=1}^{k-1} \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{w_j^2}{2}\Big)$$

$$= \Big(\frac{1}{\sqrt{2\pi}}\Big)^{k-1} \exp\Big(-\frac{\sum_{j=1}^{k-1} w_j^2}{2}\Big), \quad (11)$$

where the components of the weight vector are assumed to be independent of each other and distributed according to zero-mean Gaussian with the standard deviation 1. We also assume that the number of nodes in the neural tree is distributed according to following Poisson distribution

$$P(k-3) = \frac{\lambda^{k-3}\exp(-\lambda)}{(k-3)!}, \quad (12)$$

where $k = 3, 4, ...$ since the neural tree which consists of one terminal node was not considered. Substituting Equation (10), (11), and (12) into Equation (7), we obtain the following posterior probability for the neural tree

$$P(A|D) \propto P(D|\mathbf{w},k)P(\mathbf{w}|k)P(k)$$

$$= \Big(\frac{1}{\sqrt{2\pi}\sigma}\Big)^N \exp\Big(-\frac{\sum_{c=1}^{N}(y_c - f_{(\mathbf{w},k)}(\mathbf{x}_c))^2}{2\sigma^2}\Big)$$

$$\times \Big(\frac{1}{\sqrt{2\pi}}\Big)^{k-1} \exp\Big(-\frac{\sum_{j=1}^{k-1} w_j^2}{2}\Big) \frac{\lambda^{k-3}\exp(-\lambda)}{(k-3)!}. \quad (13)$$

3.3 Bayesian Evolution of Neural Trees

To search the structure and parameters of neural trees, we maintain a population $\mathcal{A}$ of individuals A_i at gth generation

$$\mathcal{A}(g) = \{A_1, A_2, ..., A_M\}, \quad (14)$$

where M is the population size. The initial population $\mathcal{A}(0)$ is created according to the prior probability of models, that is, each number of nodes k_i is given by the Poisson distribution (12) and then $\mathbf{w}_i$ is set by the Gaussian distribution (11). In each generation g, the error $E_i(g)$ of neural trees are evaluated as follows

$$E_i(g) = \sum_{c=1}^{N}(y_c - f_{A_i}(\mathbf{x}_c)). \quad (15)$$

Using this value, we can calculate the likelihood of each individual in the population. Finally, the posterior probability of each model is computed by Equation (13).

For constructing the next generation $\mathcal{A}(g+1)$, candidate model A_i' is first created from the parent model A_i in the current population. The candidate model is then accepted with the following probability

$$\alpha(A_i, A_i') = \min\Big\{1, \frac{P(A_i'|D)}{P(A_i|D)}\Big\} \quad (16)$$

which is called acceptance probability. The candidate model is always accepted when the posterior probability of the candidate model is higher than that of the parent; it is accepted according to the ratio of two probabilities otherwise. If the candidate model is accepted, A_i' is copied into the next generation. If candidate is rejected, then A_i is copied into the next generation.

Two major variation operators are applied to the parent models for generating candidate models. First, a crossover operator swaps two subtrees chosen at random from the parent tree A_i and another tree $A_j, (i \neq j)$ which is selected randomly from the current population to create the candidate model A_i'. Second, a mutation operator changes the type of nonterminal nodes and the index of incoming units in the subtree which is also chosen randomly from the parent tree. The probabilities for applying these operators are p_c and p_m respectively. These mating steps are performed iteratively until L individuals are produced.

Weights of a neural tree are adjusted through a stochastic hill-climbing. All components of weight vector $\mathbf{w}$ are changed just one time in random order with the following expression

$$w_j' = w_j + N(0,1) \quad j = 1, 2, ..., k_i - 1, \quad (17)$$

where k_i is the number of nodes in tree A_i and $N(0,1)$ is a normal distribution with mean 0 and variance 1. Each change of the weight is also accepted by Equation (16).

The offspring population $\mathcal{A}'(g)$ is obtained through the above procedure and we finally generate the parent population $\mathcal{A}(g+1)$ of the next generation by selecting the best M individuals from $\mathcal{A}'(g)$.

3.4 Various Implementations of the BEAs

3.4.1 Individual-Based Exploitative BEA

The crossover operator cannot be applied since there is just one model ($M = 1$) in the individual-based exploitative BEA. However, candidate model can be created by the insert and delete operator. These operators limit the modification of structure to one node for exploitative search, which

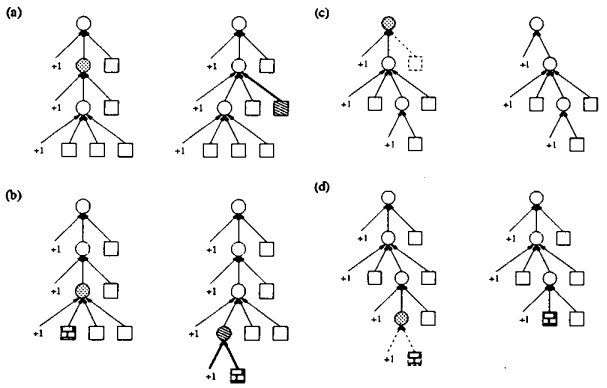

Figure 2: Insert and delete operation ($b_{max} = 3$).

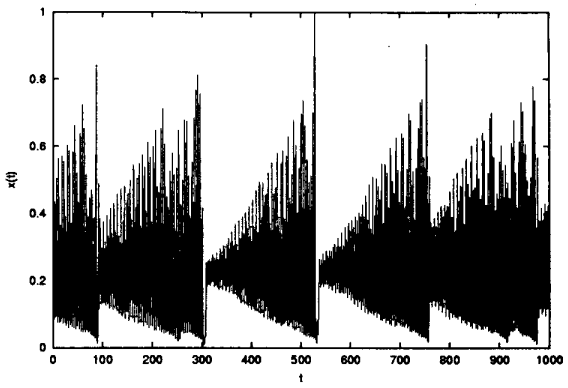

Figure 3: Laser data.

means that the candidate model is located close to the current model in the search space. The Insert operator adds a random terminal node to a randomly chosen nonterminal node (Figure 2a). If the nonterminal node has b_{max} branches, one terminal node of the children nodes is changed into a nonterminal node and the terminal node becomes the child node of the new nonterminal node (Figure 2b). The delete operator removes a random terminal node from the children nodes of randomly selected nonterminal node (Figure 2c). If the nonterminal node has only one child node, the nonterminal is removed and the child node is linked to the parent node of the nonterminal node (Figure 2d). The mutation operator is applied to not a random subtree but a random node. For example, Π node can be changed into Σ node or x_1 can be changed into x_3.

3.4.2 Individual-Based Explorative BEA

There is just one individual like the individual-based exploitative BEA but different operators are used. A randomly chosen subtree whose depth does not exceed d_s is replaced by the randomly generated tree whose depth is d_s or less ($d_s \geq 1$). Mutation can be applied to the random subtree whose depth is also d_s or less. This method can explore the search space more widely from the current model than the exploitative BEAs.

3.4.3 Parallelized Individual-Based BEA

This method maintains a large number of individuals. However, crossover is not used and thus the information is not exchanged between the individuals. That is, each individual is evolved independently and in parallel.

3.4.4 Population-Based BEAs

A population-based explorative BEA is the most general implementation of BEAs, where the crossover and mutation operators are applied to any subtree of a neural tree. This is in contrast to the population-based exploitative BEA where variation operators are limited to be applied to subtree of depth d_s or lower.

4 Experimental Results

4.1 Laser data

This data set was generated by sampling every other data point from far-infrared NH_3 laser data in a physics laboratory and have been used as a benchmark problem in the 1992 Santa Fe time series competition [8]. Following [15], the even points from the original data set were used in our experiments. The input attributes of all data set were linearly rescaled into the interval [0,1] (Figure 3).

We used the first 500 data points for evolving the neural tree models and the remaining 500 data points for testing the predictive accuracy. Other experimental setup was as follows: maximum branches of a nonterminal node was $b_{max} = 3$, which is the same as the input size, the standard deviation of the noise was $\sigma = 0.05$, the average number of nodes in a tree was $\lambda = 30$, and the maximum number of evaluations was $E_{max} = 10^6$. The candidate population size was identical to the parent population size ($M = L$) in all experiments. The sum of insert probability and delete probability was the same as the crossover probability ($p_c = p_i + p_d = 2/3$).

Figures 4 and 5 show the change in mean squared error (MSE) values of the best model for individual-based BEA (iBEA) and population-based BEA (pBEA), respectively. Tables 1 and 2 summarize the normalized MSE (NMSE) value of the best model for the test data set. All results are averaged over ten runs for each method. Both iBEA and pBEA have a better performance as the population size gets larger, but their performance degrades for very large populations. This is because the weight searching time per individual is reduced as the population size grows for the fixed evaluation time. Our best performance for the test data set is also comparable to the NMSE of 0.097 achieved by Angeline [2].

The exploitative method which uses the insert and delete operations is better than the explorative methods in iBEAs while the explorative method with unlimited crossover is bet-

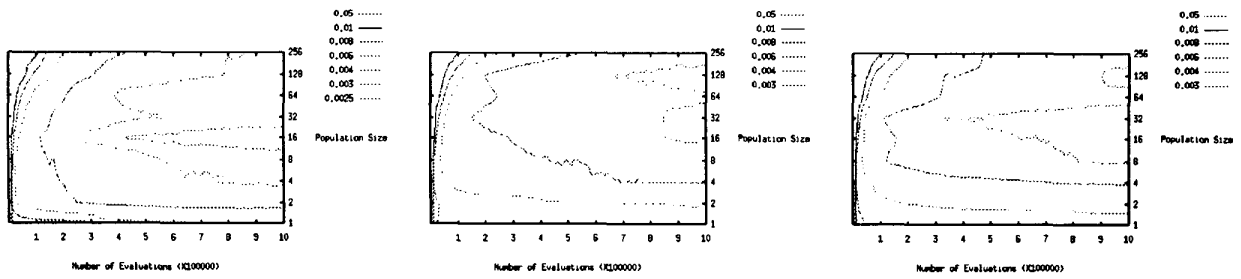

Figure 4: Evolution of MSE values for exploitative (left) and explorative (middle and right) iBEAs on the laser data. In the explorative methods, the depth d_s of subtrees replaced by variation operators does not exceed 1 (middle) or 2 (right).

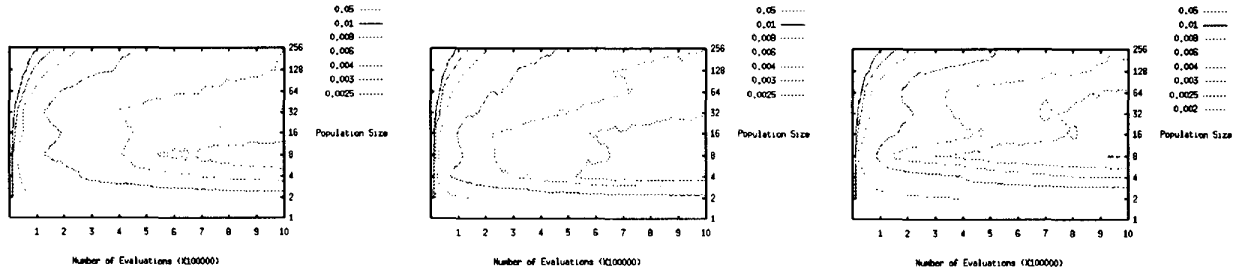

Figure 5: Evolution MSE values for exploitative (left and middle) and explorative (right) pBEAs on the laser data. In exploitative methods, the depth d_s of subtrees replaced by variation operators dose not exceed 1 (left) or 2 (middle).

Table 1: NMSE of individual-based BEAs for laser data.

	exploitative			explorative ($d_s = 1$)			explorative ($d_s = 2$)		
popsize	Mean ± Stdev	Min	Max	Mean ± Stdev	Min	Max	Mean ± Stdev	Min	Max
1	0.24647 ± 0.07857	0.11482	0.43451	0.25695 ± 0.07135	0.12948	0.38385	0.27592 ± 0.10973	0.10044	0.42620
2	0.15672 ± 0.04153	0.10917	0.22679	0.23722 ± 0.08431	0.10811	0.40822	0.20650 ± 0.07256	0.11753	0.30024
4	0.13409 ± 0.03942	0.10703	0.24168	0.17196 ± 0.03317	0.13325	0.25252	0.16807 ± 0.05415	0.11706	0.25733
8	0.12962 ± 0.02663	0.01473	0.19283	0.16436 ± 0.05175	0.11462	0.26635	0.13516 ± 0.02163	0.10900	0.17012
16	0.11929 ± 0.00700	0.10350	0.13050	0.13261 ± 0.02746	0.11585	0.21286	0.13454 ± 0.01414	0.11242	0.16075
32	0.13449 ± 0.02030	0.11634	0.17976	0.13403 ± 0.01854	0.10877	0.16529	0.12831 ± 0.01735	0.11293	0.17258
64	0.12505 ± 0.00882	0.11571	0.14449	0.14056 ± 0.01785	0.11889	0.16302	0.14739 ± 0.01865	0.11937	0.18928
128	0.12890 ± 0.00833	0.11394	0.14143	0.12918 ± 0.01102	0.11347	0.15495	0.12965 ± 0.01142	0.11580	0.15130
256	0.13487 ± 0.01546	0.11746	0.16799	0.15044 ± 0.02120	0.12176	0.19125	0.14221 ± 0.01401	0.12420	0.17256

Table 2: NMSE of population-based BEAs for laser data.

	exploitative ($d_s = 1$)			exploitative ($d_s = 2$)			explorative		
popsize	Mean ± Stdev	Min	Max	Mean ± Stdev	Min	Max	Mean ± Stdev	Min	Max
2	0.18987 ± 0.06524	0.11207	0.29105	0.18965 ± 0.07245	0.10958	0.32761	0.22444 ± 0.05837	0.10982	0.33205
4	0.13425 ± 0.02700	0.10731	0.19897	0.12047 ± 0.01171	0.10577	0.14484	0.14834 ± 0.05560	0.10784	0.26261
8	0.12222 ± 0.01154	0.10410	0.14444	0.11905 ± 0.01232	0.10627	0.14173	0.11850 ± 0.00692	0.11074	0.13449
16	0.12705 ± 0.01736	0.10698	0.16315	0.11661 ± 0.01020	0.10437	0.14099	0.12806 ± 0.01701	0.11175	0.16837
32	0.12834 ± 0.01122	0.11050	0.15154	0.12824 ± 0.00980	0.11212	0.14778	0.12358 ± 0.00997	0.11005	0.14293
64	0.12751 ± 0.01434	0.10964	0.16282	0.12825 ± 0.01115	0.11048	0.14840	0.12533 ± 0.00883	0.10929	0.13947
128	0.13843 ± 0.01687	0.11787	0.17168	0.12973 ± 0.01097	0.11192	0.15195	0.13218 ± 0.00811	0.12347	0.14827
256	0.13905 ± 0.01025	0.12357	0.15887	0.13687 ± 0.01082	0.12338	0.16160	0.13969 ± 0.01719	0.11829	0.18385

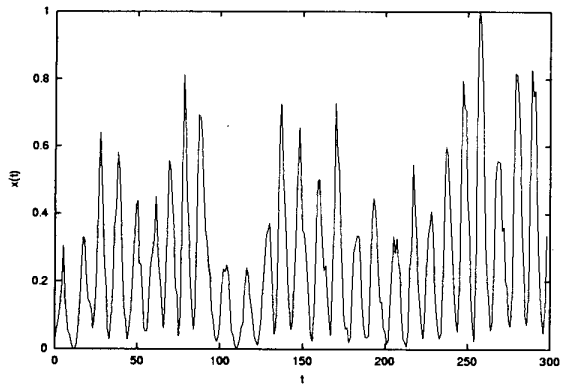

Figure 6: Sunspot data.

ter in pBEAs. Compared with iBEAs, pBEAs find better solution more efficiently, although there is little difference in the small population. This implies that the subtrees obtained from other individuals by crossover are more useful than the randomly generated subtrees.

4.2 Sunspot data

This data represents the yearly sunspot numbers from 1700 to 1998[1] (Figure 6). In our experiments, the period from 1700 to 1920 was used as the training set (221 data) and the period from 1921 to 1998 as the test set (78 data). Experimental setup is same as that of the sunspot data except for $b_{max} = 4$ and $\lambda = 40$.

Figure 7 and 8 also show the change in MSE values of the best model. The results which was obtained in the same way as the laser data were shown in Table 3 and 4. The MSE values decrease fastly as the population grow up but many individuals slow down this decreasing rate like in the laser data.

Though the difference between each methods of iBEAs is not clear, the explorative methods is a little faster than the exploitative one. In the pBEAs, the explorative methods with unlimited operations is also more robust. Unlike the laser data, however, the pBEAs does not outperform the iBEAs and exploitative methods of pBEAs are worse than the iBEAs. From this result, we can deduce that the rough models for the sunspot data is found more easily than for the laser data.

4.3 Analysis of the effects by the prior

It is difficult to add the prior knowledge about the model explicitly in general evolutionary algorithms. But it is possible to specify this knowledge through the prior probability in BEAs thus the local search can be done more effectively. Of course, general evolutionary algorithms are able to search locally by using hill-climbing for weights and elitist strategy for the structure where the best individual is always retained in the next generation. However it tends to increase the complexity of models as the evolution goes by so that models are overfitted to the training data. The prior probability prevents this misfortune by controlling the model complexity (for more formal description, see the section 3 in [16]). Figure 9 (left, middle) and 10 (left, middle) show the complexities of the best model in terms of the number of nodes and the squared sum of weight values. Most neural trees have similar complexity which is smaller than the prior complexity λ.

In general, we have no idea about the optimal models for a given problem. In this case, BEAs are especially useful since they are relatively robust with respect to prior probability. To demonstrate this, we selected different values for λ. The obtained NMSE values for the test data are shown in Figures 9 (right) and 10 (right). The whole results are similar except for pBEA starting with very small trees. Due to the lack of the diversity in the population, in this case, the model space cannot be explored well by the crossover.

[1]This data is available from the Sunspot Index Data Center in Belgium. http://www.astro.oma.be/SIDC/

5 Conclusions

In this paper, we presented Bayesian evolutionary algorithms which do Bayesian inference through evolutionary computation for evolving neural tree models. The prior probability and likelihood for the neural tree models were defined and Bayes theorem was used to estimate the posterior probabilities of individuals.

The performance of BEAs is affected by the population size and the type of variation operators. We compared the evolution speed and predictive accuracy of different BEA methods on two scientific times series prediction problems. It turned out that multiple individuals (i.e. pBEAs or parallelized iBEAs), contrary to conventional MCMC methods, is of benefit to searching the solution. Furthermore, population-based BEAs with unlimited crossover generally find good models more efficiently.

BEAs allow background knowledge about the given data to be incorporated in the procedure thus the local search can be done more effectively without overfitting. Our experimental results show that BEAs are relatively robust with respect to prior specification.

Acknowledgements

This research was supported in part by the Korea Ministry of Science and Technology through KISTEP under grant BR-2-1-G-06 and by the BK21-IT Program.

Bibliography

[1] Andrieu, C., de Freitas, N., and Doucet, A., (2000) "Robust full Bayesian methods for neural networks," *Advances in Neural Information Processing Systems*, vol. 12, pp. 379-385, MIT Press.

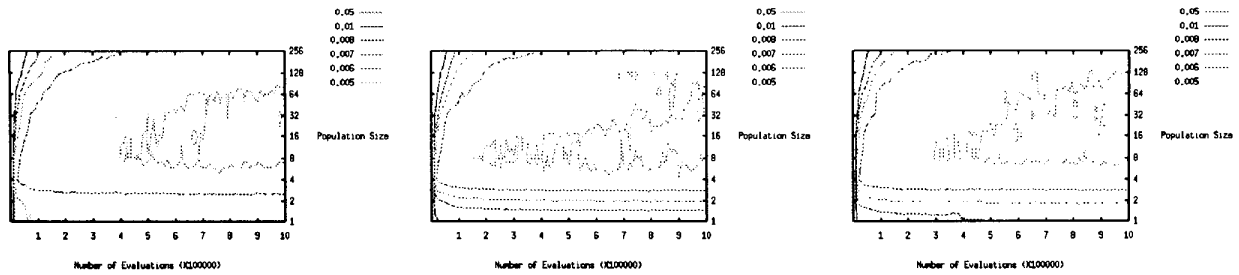

Figure 7: Evolution of MSE values for exploitative (left) and explorative (middle and right) iBEAs on the sunspot data. In the explorative methods, the depth d_s of subtrees replaced by variation operators does not exceed 1 (middle) or 2 (right).

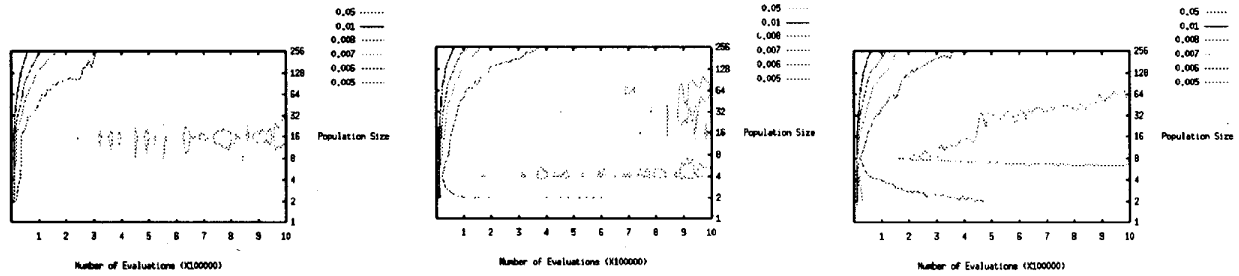

Figure 8: Evolution of MSE values for exploitative (left and middle) and explorative (right) pBEAs on the sunspot data. In the exploitative methods, the depth d_s of subtrees replaced by variation operators dose not exceed 1 (left) or 2 (middle).

Table 3: NMSE of individual-based BEAs for sunspot data.

	exploitative			explorative ($d_s = 1$)			explorative ($d_s = 2$)		
popsize	Mean ± Stdev	Min	Max	Mean ± Stdev	Min	Max	Mean ± Stdev	Min	Max
1	0.30165 ± 0.09516	0.15006	0.37014	0.32533 ± 0.10336	0.14867	0.47269	0.31768 ± 0.09745	0.18348	0.44547
2	0.24992 ± 0.09359	0.15609	0.36935	0.26006 ± 0.11097	0.15031	0.46229	0.26224 ± 0.09643	0.14336	0.41948
4	0.18334 ± 0.02899	0.15351	0.24187	0.19048 ± 0.03291	0.15423	0.25294	0.20234 ± 0.06141	0.14375	0.36553
8	0.18191 ± 0.03722	0.13776	0.23455	0.16932 ± 0.02371	0.13794	0.20983	0.18664 ± 0.02861	0.14509	0.23636
16	0.17858 ± 0.01943	0.15754	0.21690	0.18476 ± 0.01734	0.15589	0.20742	0.17371 ± 0.01968	0.15466	0.20936
32	0.19209 ± 0.03069	0.14944	0.24328	0.18258 ± 0.02707	0.14845	0.25382	0.17538 ± 0.01995	0.14864	0.20784
64	0.18074 ± 0.01306	0.16267	0.21422	0.18550 ± 0.02018	0.15848	0.22998	0.17151 ± 0.02013	0.12772	0.20491
128	0.17739 ± 0.01058	0.16293	0.19650	0.19063 ± 0.01899	0.15885	0.23424	0.18633 ± 0.02897	0.14907	0.22884
256	0.18481 ± 0.02570	0.12916	0.22363	0.20776 ± 0.03014	0.17002	0.25112	0.20531 ± 0.03068	0.15308	0.25358

Table 4: NMSE of population-based BEAs for sunspot data.

	exploitative ($d_s = 1$)			exploitative ($d_s = 2$)			explorative		
popsize	Mean ± Stdev	Min	Max	Mean ± Stdev	Min	Max	Mean ± Stdev	Min	Max
2	0.19125 ± 0.06671	0.14707	0.37528	0.21828 ± 0.07419	0.16396	0.36644	0.22825 ± 0.09014	0.15389	0.37757
4	0.19586 ± 0.05628	0.15451	0.35910	0.17564 ± 0.02319	0.14999	0.21917	0.23420 ± 0.09112	0.14284	0.37314
8	0.20076 ± 0.03009	0.15506	0.23563	0.17760 ± 0.01859	0.15997	0.22859	0.17886 ± 0.01209	0.16371	0.19822
16	0.16718 ± 0.01455	0.15228	0.20197	0.20156 ± 0.02270	0.17604	0.23938	0.17493 ± 0.01780	0.15492	0.21391
32	0.18565 ± 0.03312	0.14785	0.24501	0.19871 ± 0.01965	0.15531	0.22494	0.17991 ± 0.01664	0.15906	0.20886
64	0.19625 ± 0.02389	0.15824	0.24942	0.18090 ± 0.02259	0.15453	0.22347	0.19443 ± 0.03663	0.15429	0.25653
128	0.17864 ± 0.02378	0.15206	0.23116	0.20515 ± 0.03704	0.15522	0.26289	0.20256 ± 0.03258	0.17297	0.25408
256	0.19698 ± 0.02732	0.14884	0.25890	0.20253 ± 0.03656	0.15446	0.26733	0.19996 ± 0.03751	0.15978	0.28250

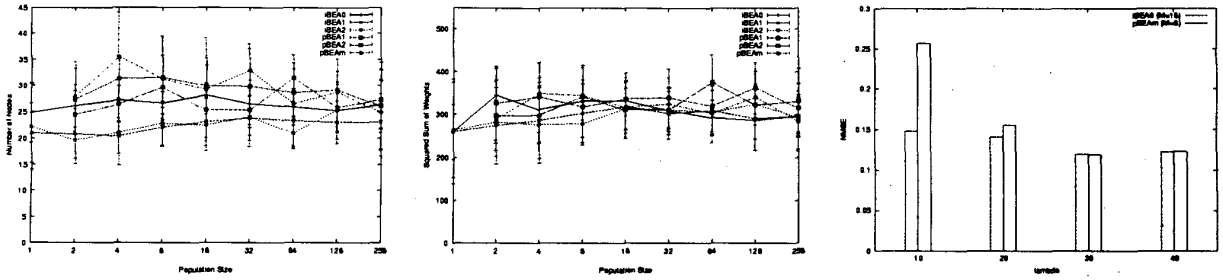

Figure 9: Effects on the number of nodes (left), weights (middle), and NMSE (right) by the prior for the laser data.

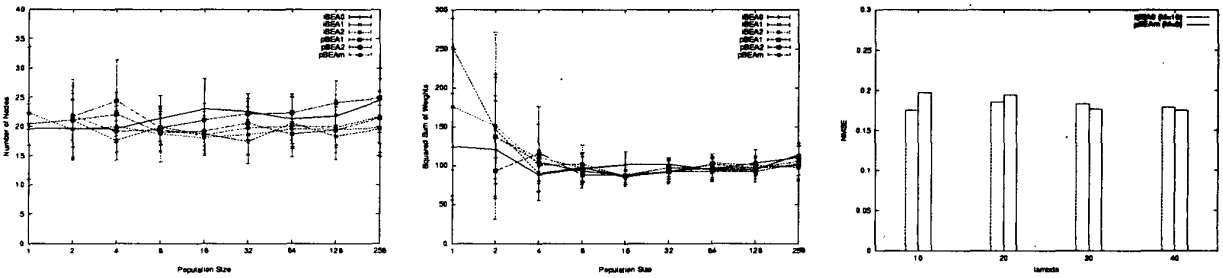

Figure 10: Effects on the number of nodes (left), weights (middle), and NMSE (right) by the prior for the sunspot data.

[2] Angeline, P.J., (1998) "Evolving predictors for chaotic time series," *Proceedings of SPIE: Applications and Science of Computational Intelligence*, vol. 3390, pp. 170-180.

[3] Box, G.E.P, Jenkins, G.M., and Reinsel, G.C., (1994) *Time Series Analysis*, 3rd Ed., Prentice-Hall.

[4] Chen, S.-H. and Lu, C.-F., (1999) "Would evolutionary computation help in designs of ANNs in forecasting foreign exchange rates," *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 1, pp. 267-274.

[5] Gilks, W.R., Richardson, S., and Spiegelhalter, D.J., (1996) *Markov chain Monte Carlo in Practice*, Chapman & Hall.

[6] Green, P.J., (1995) "Reversible jump Markov chain Monte Carlo computation and Bayesian model determination," *Biometrika*, vol. 82, no. 4, pp. 711-732.

[7] Harrald, P.G. and Kamstar, M., (1997) "Evolving artificial neural networks to combine financial forecasts," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 40-52.

[8] Hübner, H., Weiss, C.O., Abraham, N.B., and Tang, D., (1993) "Lorenz-like chaos in NH_3-FIR laser," *Time series prediction: Forecasting the future and understanding the past*, pp. 73-104, Addison-Wesley.

[9] Kreutz, M., Reimetz, A.M., Sendhoff, B., Weihs, C., and von Geelen, W., (1998) "Optimisation of density estimation models with evolutionary algorithms," *Parallel Problem Solving from Nature*, Lecture Notes in Computer Science 1498, pp. 998-1007, Springer.

[10] Pole, A., West, M., and Harrison, J., (1994) *Applied Bayesian Forecasting and Time Series Analysis*, Chapman & Hall.

[11] Press, S.J., (1989) *Bayesian Statistics: Principles, Models, and Applications*, Whiley.

[12] Richardson, S. and Green, P.J., (1997) "On Bayesian analysis of mixtures with an unknown number of components (with discussion)," *Journal of Royal Statistics Society B*, vol. 59, no. 4, pp. 731-792.

[13] Rios Insua, D. and Müller, P., (1998) "Feedforward neural networks for nonparametric regression," *Practical Nonparametric and Semiparametric Bayesian Statistics*, Lecture Notes in Statistics 133, pp. 181-194, Springer.

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

[15] Zhang, B.-T., Ohm, P., and Mühlenbein, H., (1997) "Evolutionary Induction of Sparse Neural Trees," *Evolutionary Computation*, vol. 5, no. 2, pp. 213-236.

[16] Zhang, B.-T., (1999) "A Bayesian Framework for Evolutionary Computation," *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 1, pp. 722-728.