NONLINEAR MODELING: GENETIC PROGRAMMING VS. FAST EVOLUTIONARY PROGRAMMING

MINGLEI DUAN
Department of Electrical and Computer Engineering
Marquette University
Milwaukee, Wisconsin

RICHARD J. POVINELLI
Department of Electrical and Computer Engineering
Marquette University
Milwaukee, Wisconsin

ABSTRACT
Both Genetic Programming (GP) and Fast Evolutionary Programming (FEP) combined with a Reduced Parameter Bilinear (RPBL) model have been recognized as effective time series modeling methods. This study compares the performance of these two methods for their ability to model time series data in terms of their accuracy and time efficiency. A brief review of GP and FEP are presented. Then the accuracy and time efficiency of these two methods are evaluated on several different time series. The performances of the two methods are compared against each other.

INTRODUCTION

Artificial evolutionary processes, such as genetic algorithms (GA), are based on reproduction, recombination, and selection of the fittest members in an evolving population of candidate solutions. Koza [1] extended this genetic model of learning into the space of programs and thus introduced the concept of genetic programming (GP). Sathyanarayan and Chellapilla [5] proposed an alternative modeling approach called fast evolutionary programming (FEP) to optimize the parameters of a reduced parameter bilinear model (RPBL). The RPBL model [6] is capable of effectively representing nonlinear models with the additional advantage of using fewer parameters than a conventional bilinear model. This paper applies both approaches to model several time series, including Mackey-Glass, sunspot, and stock price time series.

GENETIC PROGRAMMING

Genetic programming (GP) lets a computer learn programs. The top-level process of GP follows a similar evolutionary approach as a GA. The major difference between GPs and GAs is that GP structures are not encoded as linear genomes, but rather as terms or symbolic expressions. The units being mutated and recombined do not consist of characters or command sequences but rather functional modules, which are generally represented as tree-structured chromosomes.

The basic algorithm of GP is as follows:

1. Generate an initial population and evaluate the fitness for each individual in the population.
2. Select individuals from the population, typically using roulette or tournament methods.

3. Perform mutation, crossover and other genetic operators on the selected individuals, and form the new population using the result.
4. If the solution is sufficient, end the process and present the best individual in the population as the result. Otherwise go to step 2.

Adil Qureshi's GPsys release 2b [7] is used. The configuration used in this study is given in Table 1.

Generations	100
Populations	2000
Function set	+, -, /, *, sin, cos, exp, sqrt, ln
Terminal set	$\{x(t-1), x(t-2), \ldots, x(t-10), R\}$
Fitness	Sum of squared error
Max depth of new individual	9
Max depth of new subtrees for mutation	7
Max depth of individuals after crossover	13
Mutation rate	0.01
Generation method	Ramped half-and-half

Table 1: GP configuration

FAST EVOLUTIONARY PROGRAMMING

Fast evolutionary programming (FEP) is a variation of evolutionary strategies (ES) [9]. FEP should not be confused with Fogel's evolutionary programming [8], which evolves finite state machines. Yao and Liu [10] have shown empirically that FEP, which uses a Cauchy mutation operator, has better convergence properties than ES, which uses a Gaussian mutation operator. This was demonstrated on several multimodal functions with many local minima. Further it is comparable to ES in performance for unimodal and multimodal functions with only a few local minima.

FEP is implemented as follows [10], using a $(\mu + \lambda)$ evolution strategy.

1. Generate the initial population of μ randomly selected individuals, and set the generation number, *k* to one. Each individual is taken as a pair of real-valued vectors, (x_i, η_i), $\forall i \in \{1, \cdots, \mu\}$, where x_i is the vector elements and η_i is the corresponding variance.
2. Evaluate the error score for each individual, in terms of the objective function, $f(x_i)$.
3. Mutate each parent (x_i, η_i) to create a single offspring (x_i', η_i') by $x_i'(j) = x_i(j) + \eta_i(j)C(0,1)$, $\eta_i'(j) = \eta_i(j)\exp[\tau' N(0,1) + \tau N_j(0,1)]$ for $j = 1, \cdots, n$.
4. Calculate the fitness of each offspring.
5. Conduct pairwise comparison over the union of parents and offspring. For each individual, *q* opponents are chosen randomly from all the parents and offspring with equal probability. For each comparison, if the individual's error is no greater than the opponent's, the individual receives a "win".
6. Select the μ individuals that have the most wins to be parents of the next generation.
7. Stop if the halting criterion is satisfied; otherwise, increment the generation number and go to Step 3.

This algorithm was coded in java to make it comparable to GPsys with the same population size (2000) and number of generations (100) as used in GP.

REDUCED PARAMETER BILINEAR MODEL

The reduced parameter bilinear model (RPBL) is defined as

$$\phi_p(B)z_t = \theta_q(B)a_t + [\xi_m(B)z_t][\zeta_k(B)a_t]$$

where $\{z_t\}$ is the sequence of time series observations, $\{a_t\}$ is a sequence of independent zero mean random variables, $\phi_p(B) = 1-\phi_1 B-\phi_2 B^2 -\cdots-\phi_p B^p$, $\theta_q(B) = 1-\theta_1 B-\theta_2 B^2 -\cdots-\theta_q B^q$, $\xi_m(B) = B+\xi_2 B^2 +\cdots+\xi_m B^m$, and $\zeta_k(B) = \zeta_1 B+\zeta_2 B^2 +\cdots+\zeta_k B^k$. The variables $\phi_1,\phi_2,\cdots,\phi_p$; $\theta_1,\theta_2,\cdots,\theta_q$; $\xi_1,\xi_2,\cdots,\xi_m$; and $\zeta_1,\zeta_2,\cdots,\zeta_k$ are unknown parameters to be estimated from the time series data. The backshift operator *B* shifts the subscript of a time series observation backward in time, that is, $B^k y_t = y_{t-k}$. As can be seen, the autoregressive moving average (ARMA) model is a special case of the bilinear model where ξ_i and $\zeta_i = 0$ for all *i*.

The RPBL model is evolved by FEP using the following configuration. The individual vectors of the population consist of the model orders followed by the model parameters, as given by $x_i = [p,q,m,k,\{\phi_j\},\{\theta_j\},\{\xi_j\},\{\zeta_j\}]$. In the initial population, *p*, *q*, *m*, and *k* parameters were selected randomly from $\{1,2,\cdots,20\}$ and the model coefficients were selected uniformly from [-1, 1].

MODEL IDENTIFICATION FOR FEP

The identification procedure consists of determining the orders *p, q, m* and *k* of the model, and estimating the parameters. The model order is determined as the order that minimizes the Minimum Description Length (MDL) criterion defined as [10]

$(N-\gamma)\log(\sigma_e^2) + (1/2)(\text{ number of independent parameters})\log(N-\gamma)$,

where *N* is the number of observations of the time-series, $\gamma = \max(p,q,m,k)$

and

$$\sigma_e^2 = \left(\frac{1}{N-\gamma}\right)\sum_{t=\gamma+1}^{N}(z_t-\hat{z}_t)^2$$

$\hat{z}_t$ is the predicted output at time *t*. This criterion tries to minimize both model order and squared error at the same time. Using FEP, the model order is estimated following Yao and Liu's method [10]: Each individual in the population is a vector of the model order followed by the model parameters. In each generation, the model orders and model parameters are perturbed with continuous Cauchy random numbers and selected according to the MDL fitness criterion. The model orders are rounded to the nearest integer to obtain new model orders. The fittest vector after there is no more improvement in the fitness contains the desired model order and the model parameters.

EXPERIMENTS AND RESULTS

The time series used in the following experiments are scaled to lie between –1 and 1 before modeling. The mean square errors (MSEs) and times are all averaged over 10 runs, and σ is the standard deviation.

The Mackey-Glass equation

The first time series considered in this study is generated by the Mackey-Glass equation. The equation for the discretized Mackey-Glass map is

$$x(t+1) = x(t) + \frac{bx(t-\tau)}{1+x^c(t-\tau)} - ax(t),$$

where a=0.1, b=0.2, c=10, and τ=16. The Mackey-Glass map is seeded with 17 pseudo-random numbers, creating a 1200 points series. The first 1000 points are discarded to remove the initial transients. The next 100 points are used as the training set and the last 100 points are used as the test set, see Figure 1. Results from GP and FEP are shown in Table 2.

	GP	FEP
Training MSE	5.687×10^{-5}	4.735×10^{-5}
σ_{Train}	2.333×10^{-5}	2.613×10^{-5}
Test MSE	5.038×10^{-5}	4.648×10^{-4}
σ_{Test}	2.123×10^{-5}	1.851×10^{-4}
Time (sec)	254.5	76.1
σ_{Time}	159.8	3.7

Table 2: Results for the Mackey-Glass time series

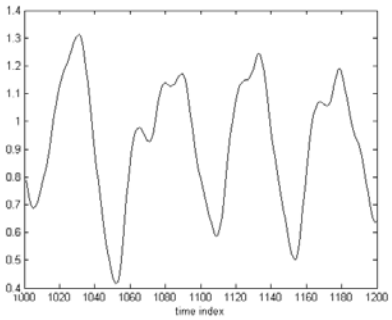

Figure 1: MackDey-Glass map

It can be seen that the models evolved by GP give much smaller MSE than FEP in the test data, although they have similar MSE for the training stage. Since the Mackey-Glass series is a totally deterministic time series, this result may imply that GP is more suitable for modeling those series with strong signals and weak noise than FEP. Even though GP takes about four times longer time, it would be the preferred method due to its better accuracy.

The Sunspot time series

The second experiment was conducted on the yearly sunspot series for the year 1800-1999 [11], see Figure 2. Once again, the first 100 data points are used for training and the next 100 are used for testing. See Table 3 for the results.

	GP	FEP
Training MSE	2.409×10^{-2}	4.019×10^{-2}
σ_{Train}	6.11×10^{-3}	1.34×10^{-3}
Test MSE	4.582×10^{-2}	5.765×10^{-2}
σ_{Test}	1.582×10^{-2}	4.23×10^{-3}
Time (sec)	205.4	70.1
σ_{Time}	28.1	4.4

Table 3: Results for the sunspot time series

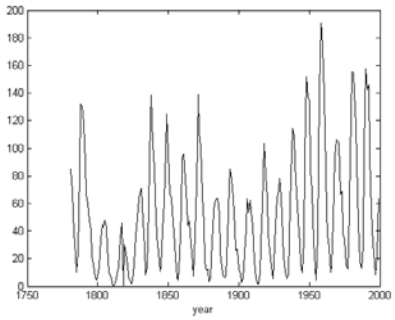

Figure 2: Sunspot time series

In modeling the sunspot time series, the accuracy performance between the two methods is not significantly different. The GP gives slightly better accuracy, but again, it takes three times as long to compute as the FEP method.

Stock prices time series

Two arbitrarily selected stocks, Compaq Computers (CPQ) on the NY Stock Exchange, and Microsoft (MSFT) on the NASDAQ, are used as the third experimental time series. The closing prices of the first 210 trading days in 1999 are used. The first 10 points are need for modeling the first prediction, and the next 200 points are divided into training and test set in the same manner as before, see Figure 3. Tables 4 and 5 present the modeling results.

	GP	FEP
Training MSE	6.597×10^{-3}	6.951×10^{-3}
σ_{Train}	3.65×10^{-4}	2.30×10^{-5}
Test MSE	7.076×10^{-3}	6.456×10^{-3}
σ_{Test}	2.19×10^{-3}	3.12×10^{-4}
Time (sec)	126	64
σ_{Time}	87.3	6.8

Table 4: Results for the MSFT time series

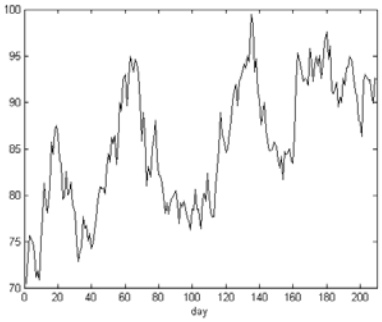

Figure 3: MSFT price time series

	GP	FEP
Training MSE	6.002×10^{-3}	7.003×10^{-3}
σ_{Train}	1.32×10^{-3}	9.15×10^{-5}
Test MSE	2.335×10^{-3}	2.148×10^{-3}
σ_{Test}	4.12×10^{-4}	7.39×10^{-5}
Time (sec)	119.6	68.1
σ_{Time}	115.3	4.0

Table 5: Results for the CPQ time series

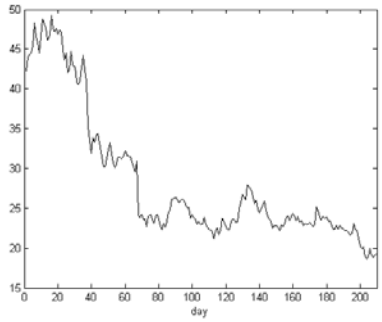

Figure 4: CPQ price time series

The results from the stock time series are similar to the sunspot results. The two methods give similar error in both training and testing, but the GP is much more time consuming. It was noticed that the results generated by FEP in each trial are consistent, but this is not the case for GP. There are larger variances in both GP's MSE and time. One interesting observation in the experiments is that as the generation increases, the models evolved by FEP tend to become simpler

while those evolved by GP always become more complex. This explains why GP is not as consistent as FEP. Because while the model becomes more and more complex, the search space is expanded rapidly, and there are a large number of local minimums into which GP could fall. In the experiments, the best solution is always found by GP. This also suggests that GPs have relatively stronger search ability.

CONCLUSIONS

In this paper, two different nonlinear modeling techniques: Genetic Programming and Fast Evolutionary Programming are applied to solve three different kinds of times series modeling problem. The GP has been shown to have better search ability than FEP, especially when deal with more predictable time series. FEP performs better when applied to noisier time series. For real world time series such as sunspot series and stock price series, it could give predictions not worse than GP, but with much less computational effort.

REFERENCES

[1] Koza, John 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: The MIT Press.

[2] Kaboudan, M. 1998. A GP approach to distinguish chaotic from noisy signals. *Genetic Programming 1998: Proceedings of the Third Annual Conference*, San Francisco. CA: Morgan Kaufmann, pp. 187-192

[3] Kaboudan, M. Genetic Programming Prediction of Stock Prices, *Computational Economics,* to appear.

[4] Fogel, D. and Fogel, L. (1996). Preliminary experiments on discriminating between chaotic signals and noise using evolutionary programming. *Genetic Programming 1996: Proceedings of the First Annual Conference.* Cambridge, MA: The MIT Press, pp. 512-520.

[5] Sathyanarayan, S. and Chellapilla, K. (1996). Evolving reduced parameter bilinear models for time series prediction using fast evolutionary programming. *Genetic Programming 1996: Proceedings of the First Annual Conference.* Cambridge, MA: The MIT Press, pp. 528-535.

[6] Zhang, Y. and Hagan, M. T. (1994), A Reduced Parameter Bilinear Time Series Model, *IEEE Trans. Signal Processing.* Vol. 42, no. 7, pp. 1867-1870

[7] Adil Qureshi's GPsys release 2b in java http://www.cs.ucl.ac.uk/staff/A.Qureshi/gpsys.html.

[8] L. J. Fogel, A. J. Owens and M. J. Walsh (1966), *Artificial Intelligence Through Simulated Evolution*, New York: John Wiley & Sons.

[9] Rechenberg, I. (1989), Evolution strategy: Nature's way of optimization. In Optimization: Methods and Applications, Possibilities and Limitations. Lecture Notes in Engineering 47. Berlin: Springer-Verlag.

[10] Yao, X. and Liu, Y. (1996), Fast evolutionary programming. *Evolutionary Programming V: Proc. of 5th Annual Conf. On Evol. Prog.*, MIT Press, Cambridge, MA, forthcoming.

[11] Yearly sunspot data from SIDC: http://sidc.oma.be/index.php3