Eur. Phys. J. B **20**, 569–572 (2001)

THE EUROPEAN PHYSICAL JOURNAL B

EDP Sciences

Rules extraction in short memory time series using genetic algorithms

L.Y. Fong and K.Y. Szeto[a]

Department of Physics, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, PR China

Received 29 August 2000

Abstract. Data mining is performed using genetic algorithm on artificially generated time series data with short memory. The extraction of rules from a training set and the subsequent testing of these rules provide a basis for the predictions on the test set. The artificial time series are generated using the inverse whitening transformation, and the correlation function has an exponential form with given time constant indicative of short memory. A vector quantization technique is employed to classify the daily rate of return of this artificial time series into four categories. A simple genetic algorithm based on a fixed format of rules is introduced to do the forecasting. Comparing to the benchmark tests with random walk and random guess, genetic algorithms yield substantially better prediction rates, between 50% to 60%. This is an improvement compared with the 47% for random walk prediction and 25% for random guessing method.

PACS. 89.65.Gh Economics, business, and financial markets – 05.45.Tp Time series analysis – 87.23-n Ecology and evolution

1 Introduction

After the introduction of Genetic algorithm (GA) by Holland [1] as a classifier and the later development by Goldberg [2], GA has found many applications in time series forecasting [3,4]. The problem of prediction is translated into one of pattern matching, learning, and generalization. Recently, this forecasting method, called *genetic algorithm optimizer* by Szeto and Cheung [5], has been extended to multiple time series. Contrary to the approach of stochastic dynamics, this new formulation of time series prediction does not require the knowledge of the equation of motion for the evolution of patterns. Rather, the forecasting problem is treated as one in image analysis [6]. In the applications, the forecasting problem is mapped onto an optimization problem with an objective function describing the probability of correct prediction, a quantity that one tries to maximize. In order that the prediction tool has some real utility, meaningful constraints are implemented. These constraints are to maintain certain level of the power of generalization and proximity of the distribution of guess to the distribution of outcomes from the training set [5]. Szeto and Luo [7] have discussed these constraints in the context of the self-organization of rules. Although these works provide many interesting insights to the use of genetic algorithms in forecasting, we still want to get a better understanding of the relation between the mechanism of the extraction of rules and the nature of the time series [8]. In order to prevent unnecessary bias of using real time series, we artificially generate a time series with controllable memory function and use simple genetic algorithm to perform extraction of prediction rules from the data. It is the aim of this paper to compare the result of genetic algorithm for the mining of rules in controlled time series with benchmark test like the random walk model and the random guess model.

2 Generation of time series

We generate time series with controllable memory measured by the autocorrelation function using the method of the inverse whitening transformation: $T : Y = TX$, to relate the uncorrelated data Y to the correlated X [9]. First let X be an $n \times n$ matrix with eigenvalues $\{\lambda_1, \lambda_2, ...\lambda_n\}$, and eigenvectors $\{\phi_1, \phi_2, ...\phi_n\}$. Here ϕ_j is a $n \times 1$ column vector. In matrix notation, this is an eigenvalue equation $X\Phi = \Phi\Lambda$, where Φ denotes the eigenvector matrix of X and Λ the eigenvalue matrix that is a diagonal matrix with entries being the nonzero eigenvalue Λ_l for the corresponding column of eigenvector matrix. Using $T = \Lambda^{-\frac{1}{2}}\Phi^{\mathrm{T}}$ as the transformation matrix, the correlation matrix of the resultant Y can be shown to be the identity matrix. Our interest is to generate a time series with correlation matrix that is X, a given matrix that can be diagonalized with nonzero eigenvalues.

Our first step is to generate an independent, normally distributed random variable Y with zero mean and unit variance: $Y = N(0, 1)$. The next step is to construct a

[a] e-mail: phszeto@ust.hk

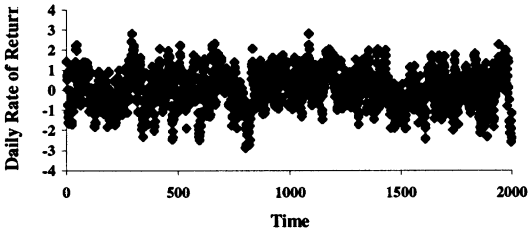

Fig. 1. A plot of the values for the daily rate of return (= fractional change) of the short memory times series with $\tau = 5$ *vs.* time.

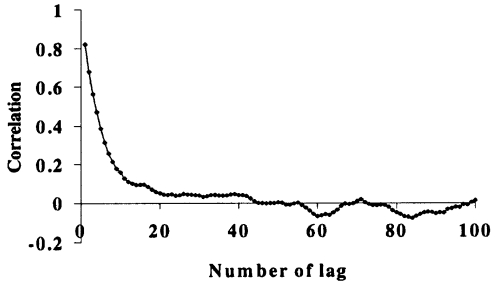

Fig. 2. A plot of correlation function *vs.* the number of lag for the above short memory time series.

Toeplitz matrix C with entries $C_{ij} = C_{ji} = C(|i-j|)$.

$$C = \begin{bmatrix} 1 & C_2 & C_3 & \dots & \dots \\ C_2 & 1 & C_2 & & \\ C_3 & C_2 & 1 & & \\ \vdots & & & \ddots & \\ \vdots & & & & 1 \end{bmatrix}. \quad (1)$$

It is a symmetric matrix with unit diagonal. The matrix elements C_2, C_3... C_n define the memory of the resultant time series. If the required time series has a short memory, one assumes an exponentially decaying function for these C,

$$C(n) = \mathrm{e}^{-\frac{n}{\tau}}. \quad (2)$$

Here τ is the range of correlation and $n = |i-j|$ is the number of days in the past. After finding the eigenvalues and corresponding eigenvectors of C, a controlled time series can be obtained by multiplying Y with the transformation $\Phi\Lambda^{1/2}Y$ to obtain X,

$$X = \Phi\Lambda^{1/2}Y. \quad (3)$$

Here X is a correlated time series with autocorrelation function C. We illustrate the artificial time series with the raw data in Figure 1, and its associated correlation function in Figures 2 and 3.

3 Training and testing of rule

Three sets of short memory time series with 2000 data points are produced. Their correlation functions are with

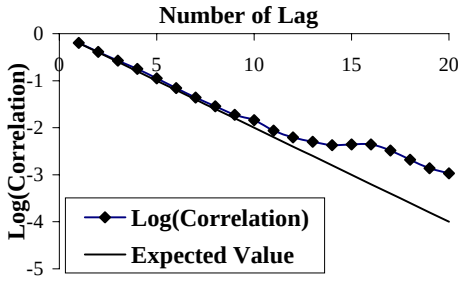

Fig. 3. A plot of correlation function *vs.* the number of lag for the above short memory time series.

$C(t) = \mathrm{e}^{-\frac{t}{\tau}}$ with $\tau = 5$, 10, and 15. The time series is then separated into two disjoint sets: the first 1000 data points form the training set and the next 1000 points form the test set.

3.1 Data preprocessing

In the training set, points are put into four categories. For instance the cut-off values of the time series with short memory with $\tau = 5$ are -0.615, 0.030 and 0.677. These cut-off values are chosen to ensure that the number of data points of each class in the training set is approximately the same. Using the cut-off values of the training set, the data points in the test set are also labeled into four classes: $(1, 2, 3, 4)$. (For $x(t) \leq -0.615$, x is in class 1; for $-0.615 \leq x(t) \leq 0.030$, x is in class 2; for $0.030 \leq x(t) \leq 0.667$, x is in class 3 and for $0.667 \leq x(t)$, x is in class 4.)

3.2 Initialization of rules

A group of 100 rules is produced randomly. A rule is defined by an "if" part (conditional unit) and a "then" part (resultant unit). The conditional unit has fixed number (=4 in this paper) of sub-units. Each sub-unit can assume one value from the set $\{1,\ 2,\ 3,\ 4, *\}$. The sub-units are linked by one of the two logical operators: (AND, OR). Now for quantized values $\{X_t\}$ of the time series, a symbolic representation of a rule with four sub-units is

$$\begin{aligned} \langle \text{If } [((X_t = I) \text{ and/or } (X_{t+1} = J)) \text{ and/or} \\ ((X_{t+2} = K)) \text{ and/or } (X_{t+3} = L))], \\ \text{then } (X_{t+4} = M)\rangle. \end{aligned} \quad (4)$$

For example, a typical rule reads

$$\begin{aligned} \langle \text{If } [((X_t = 1) \text{ and } (X_{t+1} = *)) \text{ and} \\ ((X_{t+2} = 3) \text{ or } (X_{t+3} = 2))], \\ \text{then } (X_{t+4} = 4)\rangle. \end{aligned} \quad (5)$$

Note that the star *= *don't care* symbol does not appear in the "then" part, as we want definite prediction. In the initialization, the "then" part has four different classes, so that we generate 25 rules for each class. The evolution of each subset of rules (25 for class 1, 25 for class 2, etc.) only involves changes on the if part. This separation of the 100 rules into 4 subsets is aimed at specialization in the training process for each class.

3.3 Fitness evaluation

Initially all rules are assigned to be zero fitness. In each training step, the rules for class k is trained by comparing the patterns in the training set, and three possible cases can arise:

Case 1: the "if" part of the rule does not match the data point pattern. In this case, we cannot make prediction.

Case 2: the "if" part of the rule matches the data points in the training set. We make a prediction. When the "then" part of the rule also matches the class of the corresponding data point, it is counted as a correct guess; otherwise it is counted as a wrong guess. The fitness value of this rule i will be

$$F_{\mathrm{i}} = N_{\mathrm{c}}/N_{\mathrm{g}} = N_{\mathrm{c}}/(N_{\mathrm{c}} + N_{\mathrm{w}}). \tag{6}$$

Here N_{c} is the number of correct guess and N_{w} is the number of wrong guess, so that

$$N_{\mathrm{g}} = N_{\mathrm{c}} + N_{\mathrm{w}}. \tag{7}$$

Case 3: there are more than one rules with the "if" part which match the data points in the training set. We then have to decide how to make a prediction. We make use of the specificity of the rule. The most specific rule is the one that has no "don't care" and all 3 logical operator are "AND". Assuming the occurrence of each possible pattern is the same, the probability of matching the "if" part of this rule is $1/(4 \times 4 \times 4 \times 4)$=1/256. On the other hand, if there are three "don't care" appearing in the rule, the probability is just 1/4. Hence the more specific rule has a lower probability to appear and contain more information, and its prediction should be considered with more weight than the less specific rule. Therefore, in this case, we will choose the prediction of the more specific rule. Note that both kinds of rules co-exist in the rule set, as they are complementary: rule with low specificity predicts more kinds of patterns, while rule with high specificity gives a more precise prediction. Note also that there are only 1024 rules without "don't care" symbol and no "or" operator, and they form a set with highest specificity.

3.4 Evolution of rules

We used 100 groups, each containing 100 rules. 50 groups with fitness higher than medium are selected to survive. There is no need to re-evaluate the fitness of the rules in these groups in the next generation, thereby increasing its efficiency compared to the roulette wheel selection method. The remaining 50 groups of the new generation is produced by crossover (25) and mutation (25) so that the population size ($= 100 \times 100$) remains unchanged throughout training. The mutation and crossover are performed only on the conditional parts, so that the number of rules for each class in each group remained unchanged. After 1000 generations, the fittest group is selected for real test. In Figures 4–5, we observe steps in fitness, corresponding to sudden improvement in percentage of correct guess. In time series prediction, we want to maximize the correct percentage AND the guessing percentage [10]. This difficult requirement is partially accomplished by including both specific rules and general rules. The former aims to increase the correct percentage N_{c} while the latter to increase total number of guess N_{g}.

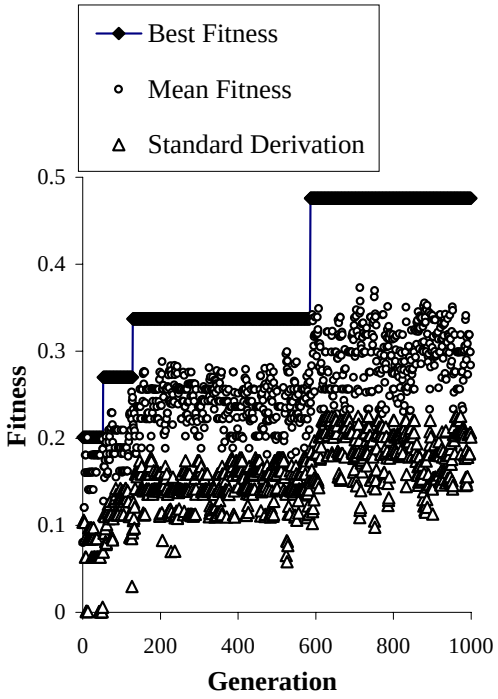

Fig. 4. Fitness of the best chromosome, average fitness and standard derivation *vs.* generation.

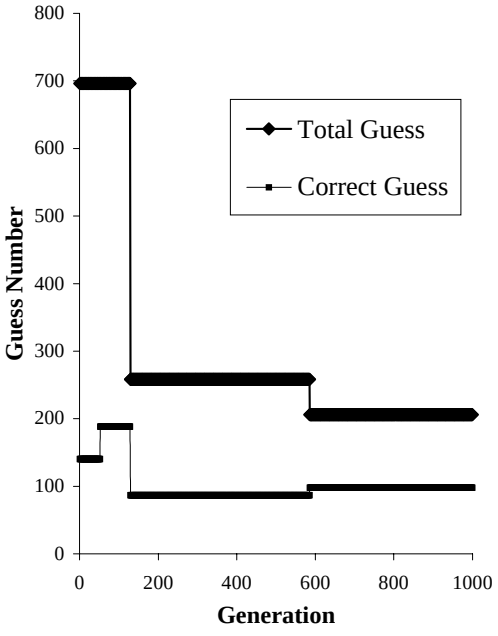

Fig. 5. Total number of guess and correct guess of the best group *vs.* generation

Table 1. Comparison of prediction in terms of percentage of correct prediction.

Correlation, $C(t)$	$e^{-\frac{t}{5}}$	$e^{-\frac{t}{10}}$	$e^{-\frac{t}{15}}$
Genetic algorithm	50%	56%	57%
Random guessing	25%	25%	24%
Random walk	46%	48%	46%

4 Discussion

After training, we choose the group with highest fitness for prediction. In this chosen group, there are 25 rules for each class and the mechanism for making a prediction is the same as in the training set. In general the percentage of correct guess between 50% and 60% can be achieved. For comparison, random guess and random walk predictions are also used as benchmark. Random guess is the simplest method of prediction as a guess is done without previous information. The percentage of correct random guess is only 25% as there are 4 classes in the data. Random walk method uses the value of previous time unit to predict the present unit. (If the last unit is 3, then the prediction is 3 too.) This method is better than random guess and 46% of predictions are correct. A comparison is made in Table 1. We see that genetic algorithm performs better than the random guess and random walk method. Furthermore, besides providing a set of rules for forecasting, GA also performs better in time series with longer memory (larger τ). This is quite reasonable since the rules must exploit the correlation within the time series and the longer memory the series has, the better the data mining. This application has also seen useful application in real financial time series.

K.Y. Szeto acknowledges the support of grants DAG 98/99. SC25 and RGC 6144/00P. L.Y. Fong acknowledges discussion with classmates Mr. Lee Fukay and Mr. Cheung Ho Yin.

References

1. J.H. Holland, *Adaptation in Natural and Artificial Systems* (Ann Arbor, MI: University of Michigan Press, 1975).
2. D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison Wesley, 1989).
3. K.Y. Szeto, K.H. Cheung, *Proceedings of the World Multiconference on Systemic, Cybernetics and Informatics, Caracas* **3**, 390–396 (1997).
4. K.Y. Szeto, K.O. Chan, K.H. Cheung, *Contributed paper of Proceedings of the Fourth International Conference on Neural Networks in the Capital Markets Progress in Neural Processing, Decision Technologies for Financial Engineering*, edited by A.S. Weigend, Y. Abu-Mostafa, A.P.N. Refenes, (World Scientific, NNCM-96, 1997), pp. 95–103.
5. K.Y. Szeto, K.H. Cheung, *Proceedings of the International Symposium on Intelligent Data Engineering and Learning, Hong Kong* (IDEAL'98, 1998) pp. 127–133.
6. T. Froehlinghaus, K.Y. Szeto, *Proceedings of the International Conference on Neural Information Processing, Hong Kong, 1996, Springer Verlag* (ICONIP'96, 1996) Vol. 2, pp. 799–802.
7. K.Y. Szeto, P.X. Luo, *Self Organized Genetic Algorithm in forecasting stock market, Proceeding of the the Sixth International Conference "Forecasting Financial Markets" FFM'99 (London, 26-28 May 1999, CD-ROM).*
8. S. Zemke, Physica A **269**, 177 (1999).
9. Keinosuke Fukunaga, *Introduction to Statistical Pattern Recognition* (Academic Press, 1990).
10. D.G. Papageorgiou, I.N. Demetropoulos, I.E. Lagaris, Computer Physics Communications **109**, 227 (1998).