TIME SERIES PREDICTABILITY

By

Minglei Duan, B.S.

A Thesis Submitted to the Faculty of the Graduate School,

Marquette University, in the Requirement of the Degree of Master of Science

Milwaukee, Wisconsin

April 5, 2002

Acknowledgement

I would like to thank Dr. Richard Povinelli for his consistent support and encouragement in the past three years. His initial ideas, insightful suggestions, and wise management have made the completion of this work possible. I have learned a lot from working with him, his active attitude towards research, his earnest, his preciseness, and his humor. I am grateful to my committee members, Drs. Xin Feng, Michael Johnson, and Jeffrey Hock, who have provided great comments and suggestions to this thesis.

I thank my labmates, Felice Roberts, Bin Chen, Xiaolin Liu, and all our Computational Intelligence Seminar members. They have given me a lot of help during my research.

I am grateful to Marquette University for its financial support of this research, and the faculty of the Electrical and Computer Engineering Department for providing a great environment for studying and researching.

I am deeply grateful to my parents, for their constant support, care, and love.

Abstract

A new metric that quantifies the predictability of a time series is introduced. This new time series predictability metric is developed based on the η-metric method introduced by Kaboudan, but overcomes the resolution and stationarity problems presented in the pure η-metric method. It also provides a new feature, which shows how the predictability changes over different subsequences in a time series. The new metric can be built on top of many time series modeling methods and improves their performance in time series forecasting. Successful attempts have been made with Genetic Programming (GP) and Artificial Neural Networks (ANN) in the application of stock time series prediction.

Table of Contents

Chapter 1 Introduction

1.1 Motivation

In general, time series predictability is a measure of how well future values of a time series can be forecasted, where a time series is a sequence of observations $\{y_t, t = 1, 2, \ldots, N\}$. Time series predictability indicates to what extent the past can be used to determine the future in a time series. A time series generated by a deterministic linear process has high predictability, and its future values can be forecasted very well from the past values. A time series generated by an uncorrelated process has low predictability, and its past values provide only a statistical characterization of the future values.

In practice, a given time series is not simply deterministic or stochastic, but rather some combination of both. Predictability can be viewed as the signal strength of the deterministic component of the time series to the whole time series. In this study, the deterministic component is estimated by a given modeling method, whereas the stochastic component is estimated by the corresponding residuals. Thus, the time series predictability under a particular modeling method can be measured.

Measuring the predictability of a time series is useful because it can make a prediction of the model accuracy and thus tell whether a time series can be predicted under this particular model. Therefore prediction of a time series with low predictability, such as a random walk time series, can be avoided. For a low predictability time series, past observations are of little use in predicting future values, and the future values are determined randomly or by unknown factors. An accurate metric of time series predictability provides a measure of confidence in the accuracy of a prediction.

There have been numerous publications in the area of nonlinear time series modeling and prediction [1-5] over the last ten years, but few have studied the predictability of a time series [6]. This thesis presents a new time series predictability metric for use with nonlinear time series modeling techniques. This new metric will be shown to have better characteristics than existing metrics. Use of the metric in conjunction with a time series modeling method in financial modeling applications will show significant performance improvement in comparison to using the time series modeling method alone.

1.2 Problem Statement

Time series analysis builds models that describe the underlying system that generates a time series. Some approaches to time series analysis include Autoregressive Integrated Moving Average (ARIMA) or Box-Jenkins time series analysis, artificial neural networks (ANN), and genetic programming (GP). The focus of this research is to develop quantitative metrics that characterize time series according to their ability to be modeled by a particular method, such as the predictability of a time series using the GP approach or an ANN. Time series predictability provides a measure of how well a time series can be modeled by a particular modeling method, or how well a prediction can be made by this modeling method. Particularly, if time series *A* has higher predictability than time series *B*, the predictions made on *A* should have smaller errors than the predictions made on *B* on average.

In financial applications, knowing the predictability facilitates risk minimization and return maximization for investment decisions. The major application of the developed time series predictability metrics in this thesis will be to address the problem of choosing stocks in which to invest. This stock selection application provides a good example of

the use of time series predictability metric. Assuming an investment goal of maximizing return, while keeping the risk as low as possible, the objective is to identify stocks that are more predictable for a given modeling method. This can be done by evaluating the predictability value for each member of a set of financial time series, and ranking them according to their predictability value. Trading on higher ranked (higher predictability value) financial time series is expected to have better return/risk performance since the predictions made on these time series are on average more accurate.

1.3 Thesis Outline

The thesis is divided into seven chapters. Chapter 2 reviews the background information underlying this research including general concepts of time series analysis and data mining, several time series modeling techniques, and previous work about time series predictability.

Chapter 3 presents the definition of the new time series predictability metric and the methods used for estimating it. Chapter 4 applies the new metric to several sample time series to show the characteristics of the metric.

In Chapter 5, the experimental results from analyzing stock market open price changes using both genetic programming and artificial neural networks are given. Chapter 6 discusses two other alternative modeling approaches. The last chapter summarizes the thesis and discusses future work.

Chapter 2 Historical Review

This chapter reviews the foundation of time series predictability research, which includes the basic concepts of time series analysis [7], linear [7] and nonlinear time series modeling methods [8, 9], and the previous approach of time series predictability [4].

The first task of time series forecasting is to select an appropriate modeling technique. This selection may be application dependent. According to the No Free Lunch (NFL) theorems [10], there is no search algorithm that can outperform all other search algorithms over all possible search problems, however, a particular algorithms may be better suited for a particular problem domain. Kaboudan reported that genetic programming (GP) showed an equivalent or better performance in predicting stock price time series [4, 6] than other modeling methods. Other methods such as artificial neural networks (ANN) are also recognized as effective modeling methods in the problem of financial market forecasting [11]. However the main contribution of this thesis is an extension of these methods. The results presented by other authors using these methods are verified in this thesis. A good modeling method helps achieve the first investment goal, maximizing the expect return.

The other investment goal, reducing the risk, is what time series predictability is applicable to. There are hundreds of different stocks that can be traded in the stock market. To reduce the risk, one needs to identify those stocks with low levels of unpredictable variations, i.e., with high level of predictability. A good metric of time series predictability would make this work straightforward - simply evaluate the

predictability for each stock time series and select those with highest predictability metric values.

By design, the computed metric should approach zero for a complex signal that is badly distorted by noise. Alternatively, the computed metric should approach one for a time series with low complexity and strongly deterministic signal. An η-metric, which follows the above design objective, was introduced by Kaboudan [6]. Kaboudan's η-metric measures the level of GP-predictability of a time series. A review of this metric is given in the next chapter.

The goal of this research is to investigate new time series predictability metric with better behavior than Kaboudan's (see section 2.5 and 3.1) so that it can be applied to the real world applications. This will be an original contribution to the field of time series analysis and data mining. This thesis provides an explicit measure of time series predictability. Based on this predictability metric, it proposes new thoughts in using the forecasting results more effectively and thereby improving the efficiency of real world applications.

2.1 ARIMA Time Series Analysis

Traditional time series analysis techniques such as the Box-Jenkins [12] or Autoregressive Integrated Moving Average (ARIMA) [7] method have been well developed and widely used in the area of time series modeling. However, the ARIMA method is limited by the requirement of stationarity of the time series. The statistical characteristics of a stationary time series remain constant though time. Additionally, the residuals, the differences between the time series and the ARIMA model, are independent and normally distributed.

The general Box-Jenkins or ARIMA model of order (*p, P, q, Q*) is

$$\phi_p(B)\phi_P(B^L)z_t = \delta + \theta_q(B)\theta_Q(B^L)a_t .$$

Here

- $\phi_p(B) = (1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p)$ and is called the *nonseasonal autoregressive operator of order p*.
- $\phi_P(B^L) = (1 - \phi_{1,L} B^L - \phi_{2,L} B^{2L} - \cdots - \phi_{P,L} B^{PL})$ and is called the *seasonal autoregressive operator of order P*.
- $\theta_q(B) = (1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q)$ and is called the *nonseasonal moving average operator of order q*.
- $\theta_Q(B^L) = (1 - \theta_{1,L} B^L - \theta_{2,L} B^{2L} - \cdots - \theta_{Q,L} B^{QL})$ and is called the *seasonal moving average operator of order Q*.
- $\delta = \mu\phi_p(B)\phi_P(B^L)$ is a *constant term*, where μ is the true mean of the stationary time series being modeled.
- $\phi_1, \phi_2, \cdots, \phi_p; \phi_{1,L}, \phi_{2,L}, \cdots, \phi_{P,L}; \theta_1, \theta_2, \cdots, \theta_q; \theta_{1,L}, \theta_{2,L}, \cdots, \theta_{Q,L}$; and δ are *unknown parameters* that must be estimated from sample data.
- $a_t, a_{t-1}, \cdots$ are *random shocks* that are assumed to be statistically independent of each other; each is assumed to have been randomly selected from a normal distribution that has mean zero and a variance that is the same for each and every time period *t*.
- The symbol *B* is called the *backshift operator*. It shifts the subscript of a time series observation backward in time. That is, $By_t = y_{t-1}$, and $B^k y_t = y_{t-k}$

Bowerman [7] suggests three steps to identify the particular form of the ARIMA model that describes a particular stationary time series $\{Z_t\}$.

1. Whether the constant term δ should be included in the model.
2. Which of the operators $\phi_p(B)$, $\phi_P(B^L)$, $\theta_q(B)$, and $\theta_Q(B^L)$ should be included in the model.
3. The order of each operator that is included in the model.

Assuming that all observations in the time series are normally distributed, the δ should be included if

$$\left| \frac{\mu_Z}{\sigma_Z / \sqrt{N_Z}} \right| > 2,$$

where μ_Z is the mean of the time series, σ_Z is the standard deviation of the time series, and N_Z is the number of time series observations. Two statistical functions, the sample autocorrelation function (SAC) and sample partial autocorrelation function (SPAC), are used in step 2 and 3. The detailed procedures are included in [7].

2.2 Genetic Programming (GP)

Artificial evolutionary processes, such as genetic algorithms (GA) [13], adapt concepts from evolutionary biology to fields of engineering, optimization, and machine learning. These concepts include reproduction, recombination, mutation, survival of the fittest, and populations. Such algorithms evolve populations of candidate solutions to a problem with the goal of finding near optimal candidates. Koza [14] extended this genetic model of learning into the space of programs and thus introduced the concept of genetic programming (GP). Each candidate solution in the search space is represented by a genetic program. Genetic programming is now widely recognized as an effective

search paradigm in artificial intelligence, databases, classification, robotics and many other areas [2, 4, 14-17].

Genetic programming lets a computer learn programs by itself. The top-level process of genetic programming follows a similar evolutionary approach as a genetic algorithm. The major difference between genetic programming and genetic algorithms is that a genetic program's structures are not encoded as linear genomes, but as terms or simple symbolic expressions. The units being mutated and recombined do not consist of characters or command sequences but of functional modules, which can be represented as tree-structured chromosomes.

2.2.1 Basic Evolutionary Algorithm

1. Generate initial population.
2. Evaluate fitness for each individual in the population.
3. Selection.
4. If solution is sufficient, end the process and present the best individual in the population as the output from the algorithm.
5. Do variations by mutation, crossover and other genetic operators on the selected individuals.
6. Form the new population using the result of the genetic operations.
7. Go to step 2.

The flowchart of this process is shown below in Figure 2.1.

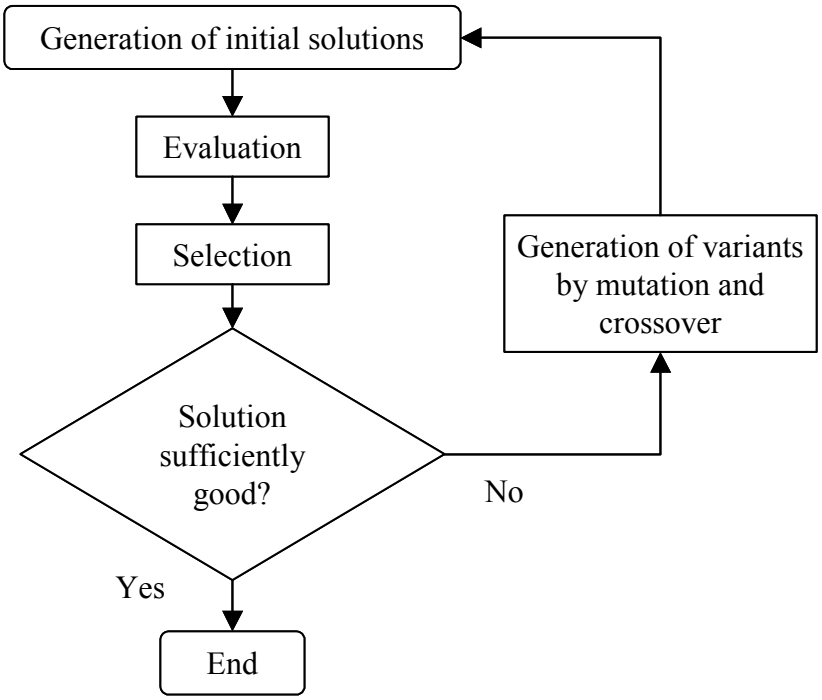

Figure 2.1: Flowchart of evolutionary algorithm

2.2.2 Simple Description of Genetic Programming

GP represents a problem as the set of all possible computer programs or a subset thereof that are less than a designated length. It maintains a population of solutions and evolves it. It uses crossover and mutation as the transformation operators to change candidate solutions into new candidate solutions. A user-defined fitness function is used to select and keep better candidate solutions in the population. GP typically is implemented as a form of supervised machine learning.

2.2.3 GP Fundamentals

2.2.3.1 Terminals and Functions

The terminal set is comprised of the inputs to the GP program, the variable and constants supplied to the GP program. The function set is composed of the statements, operators, and functions available to the GP system. A simple example of function set and terminal set is as follows.

Function set: {+, - ,*, /, sin, cos, exp},

Terminal set: {a, b, 1}.

2.2.3.2 GP's Tree Structure

The tree structure is the most frequently used representation in GP. The nodes of the tree are selected from the function set while the leaves are from the terminal set. Each GP tree represents a single individual (genotype) in the population. See Figure 2.2 for an example. The genetic program (phenotype) represented by this tree is

$$((a+b)/c)*(d+e).$$

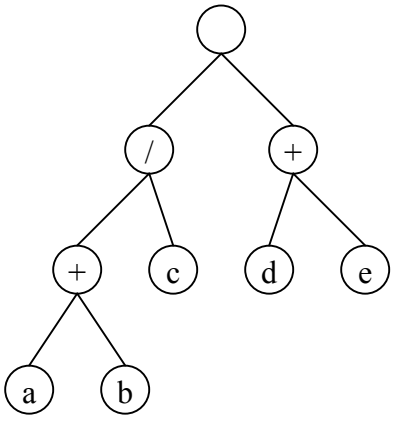

Figure 2.2: Example of a simple GP tree

2.2.3.3 Genetic Operators

Crossover: combines the genetic material of two parents by swapping a part of one parent with a part of the other. An example of the crossover operation is shown in Figure 2.3.

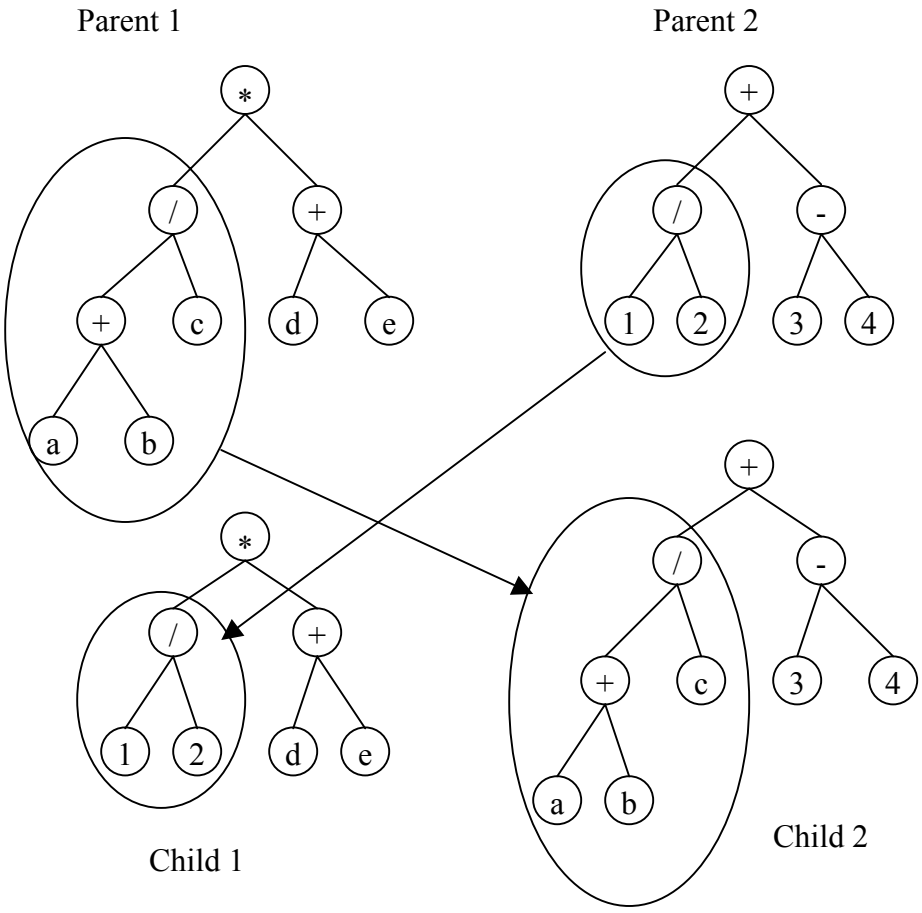

Figure 2.3: Example of the crossover operation

Mutation: select a point in the tree randomly and replaces the existing subtree at that point with a new randomly generated subtree.

Selection: decide whether to apply genetic operators to a particular individual and whether to keep it in the population or allow it to be replaced, based on the fitness of that individual.

2.2.3.4 Fitness

Fitness is the measure used by GP during simulated evolution of how well a genetic program (individual) has learned to predict the output from the input.

2.2.4 Previous Work of GP in Time Series Predictability

Fogel and Fogel [3] added noise to data generated by the Lorenz system and the logistic system. Using GP and Akaike's information criterion (AIC) [18], they found that signals with no noise are more predictable (measured by average prediction error) than noisy ones. Their results suggest the potential for evolving models of chaotic data, even in background noise. Evolutionary programming can be used to optimize parameter estimates associated with models of chaotic time series in light of observed data.

Kaboudan [4] applied GP to estimate the predictability of stock price time series. He tried to find the best-fit model for a time series using GP by minimizing the sum of squared error (SSE). His predictability metric was defined based on comparing the SSE between the original time series and its reshuffled version. Kaboudan's η-metric is the only clearly defined predictability metric found in the literature. This research extends his metric and applies the newly developed metric to a set of financial time series experiments.

The advantages of GP include its ability to evolve arbitrarily complex equations without requiring a model with an *a priori* structure, and the flexibility in selecting the terminal set and function set to fit different kind of problems.

2.3 Fast Evolutionary Programming (FEP) and Reduced Parameter Bilinear Model (RPBL)

Rao and Chellapilla [19] proposed an alternative modeling approach called fast evolutionary programming (FEP) [20] to optimize the parameters of a reduced parameter bilinear model (RPBL). The RPBL approach [21] is capable of effectively modeling nonlinear time series with fewer parameters than a conventional bilinear model. FEP,

which can be used to determine RPBL model structure, is shown here in this thesis to have reasonable optimization performance. FEP evolves RPBL models with lower normalized mean squared error (NMSE) and also lower model order than evolved with conventional evolutionary programming [19, 22]. This approach will be shown to have less computational cost and less model complexity when compared with GP.

2.3.1 Fast Evolutionary Programming

Fast evolutionary programming (FEP) is a variation of evolution strategies (ES) [23]. FEP should not be confused with Fogel's evolutionary programming [24], which evolves finite state machines. Yao and Liu [20] 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 [20], using an $(\mu + \lambda)$ 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 $(\mathbf{x_i}, \boldsymbol{\eta_i})$, $i = 1, \ldots, \mu$, where $\mathbf{x_i}$ includes the values of the solution vector elements and $\boldsymbol{\eta_i}$ includes the mutation parameter values. Typically the elements of $\mathbf{x_i}$ are selected randomly following a uniform distribution over the search space.
2. Evaluate the error score for each individual, in terms of the objective function, $f(\mathbf{x_i})$.
3. Mutate each parent $(\mathbf{x_i}, \boldsymbol{\eta_i})$ to create a single offspring $(\mathbf{x_i'}, \boldsymbol{\eta_i'})$ 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 $j = 1, \cdots, n$, where $x_i(j)$, $x_i'(j)$, $\eta_i(j)$ and $\eta_i'(j)$ denote the j-th component of the vectors $\mathbf{x_i}$, $\mathbf{x_i'}$, $\boldsymbol{\eta_i}$ and $\boldsymbol{\eta_i'}$, respectively. $N(0,1)$ is a normally distributed one-dimensional random variable with mean zero and standard deviation one. $C(0,1)$ is a random variable satisfying the standard Cauchy distribution. The probability density function for $C(t,s)$ is $f(x) = \dfrac{1}{s\pi(1 + ((x-t)/s)^2)}$, where t is the median of the distribution. The mean and the standard deviation of the Cauchy distribution are undefined. $N_j(0,1)$ indicates that the random variable is generated for each value of j. The factors τ and τ' are commonly set to be $\left(\sqrt{2\sqrt{n}}\right)^{-1}$ and $\left(\sqrt{2n}\right)^{-1}$ [25].

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.

2.3.2 Reduced Parameter Bilinear Model

The reduced parameter bilinear model (RPBL) [21] 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 random variables having a $N(0,1)$ distribution,

$$\phi_p(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p,$$
$$\theta_q(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q,$$
$$\xi_m(B) = B + \xi_2 B^2 + \cdots + \xi_m B^m, \text{ and}$$
$$\zeta_k(B) = \zeta_1 B + \zeta_2 B^2 + \cdots + \zeta_k B^k.$$

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 used in FEP 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].

2.3.3 Model Identification for FEP

The identification procedure consists of determining the orders *p, q, m* and *k* of the model and estimating the corresponding parameters. The model order is determined as the order that minimizes the Minimum Description Length (MDL) criterion defined as [19]

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

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

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

The predicted output $\hat{z}_t$ at time t is obtained using the model with order (p,q,m,k). This criterion tries to minimize both model order and squared error at the same time. Using FEP, the model order is estimated following Rao and Chellapilla's method [19]: 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. The model orders are then rounded to the nearest integer to obtain the new model orders. The model orders and parameters are selected according to the MDL fitness criterion. The best vector in the final generation contains the desired model order and the model parameters.

2.4 Time Series Data Mining (TSDM) Method

Povinelli introduced a new framework for analyzing time series data called Time Series Data Mining (TSDM) [5, 26, 27]. This framework adapts and innovates data mining concepts to analyzing time series data. It creates a set of methods that reveal hidden temporal patterns that are characteristic and predictive of time series events. Unlike traditional time series analysis methods which attempt to characterize and predict all time series observations, TSDM methods focus on characterizing and predicting events, and therefore overcome the limitations of requiring stationarity of the time series and normality and independence of the residuals. The possibility of combining this method with the predictability metric will be discussed in Chapter 6.

2.4.1 Key Concepts in Time Series Data Mining

An *event* is defined as an important occurrence in time. The associated event characterization function $g(t)$, defined *a priori*, represents the value of future "eventness" for the current time index.

Defined as a vector of length Q or equivalently as a point in a Q-dimensional space, a temporal pattern is a hidden structure in a time series that is characteristic and predictive of events.

A *phase space* is a Q-dimensional real metric space into which the time series is unfolded. The *augmented phase space* is defined as a Q+1 dimensional space formed by extending the phase space with the additional dimension of $g(\cdot)$.

The *objective function* represents a value of fitness of a temporal pattern cluster or a collection of temporal pattern clusters. Finding optimal temporal pattern clusters that characterize and predict events is the key of the TSDM framework [5].

2.4.2 Time Series Data Mining Method

The first step in applying the TSDM method is to define the TSDM goal, which is specific to each application, but may be stated generally as follows. Given an observed time series, the goal is to find hidden temporal patterns that are characteristic of events in the time series, where events are specified in the context of the TSDM goal.

Given a TSDM goal, an observed time series to be characterized, and a testing time series to be predicted, the steps in the TSDM method are:

1. Training Stage
    1) Frame the TSDM goal in terms of the event characterization function, objective function, and optimization formulation.

a. Define the event characterization function *g*.

b. Define the objective function *f*.

c. Define the optimization formulation, including the independent variables over which the value of the objective function will be optimized and the constraints on the objective function.

2) Determine *Q*, i.e., the dimension of the phase space and the length of the temporal pattern.

3) Transform the observed time series into the phase space using the time-delayed embedding process.

4) Associate with each time index in the phase space an eventness represented by the event characterization function. Form the augmented phase space.

5) In the augmented phase space, search for the optimal temporal pattern cluster, which best characterizes the events.

6) Evaluate training stage results. Repeat training stage as necessary.

2. Testing Stage

1) Embed the testing time series into the phase space.

2) Use the optimal temporal pattern cluster for predicting events.

3) Evaluate testing stage results.

2.4.3 Optimization Method – Genetic Algorithm

The basic genetic algorithm is adapted to the TSDM framework [5]. These adaptations include an initial random search and hashing of fitness values. The adapted genetic algorithm is as follows.

1. Created an elite population
    1) Randomly generate a large population (n times normal population size)
    2) Calculate fitness
    3) Select the top $1/n$ of the population to continue
2. While all fitnesses have not converged
    1) Selection
    2) Crossover
    3) Mutation
    4) Reinsertion

Initializing the genetic algorithm with the results of a Monte Carlo search has been found to help the optimization's rate of convergence and in finding a good optimum. The hashing modification reduces the computation time of the genetic algorithm by 50%.

2.5 Existing Time Series Predictability metric (η-metric)

An η-metric was introduced by Kaboudan [6], which measures the probability that a time series is GP-predictable. By design, the computed metric should approach zero for a complex signal that is badly distorted by noise. Alternatively, the computed metric should approach one for a time series with low complexity and strongly deterministic signal.

This metric is based on comparing two outcomes: the best fit model generated from a single data set before shuffling with the best fit model from the same set after shuffling. The shuffling process is done by randomly re-sequencing an observed data set using Efron's bootstrap method [28]. Figure 2.4 shows an example of the shuffling process.

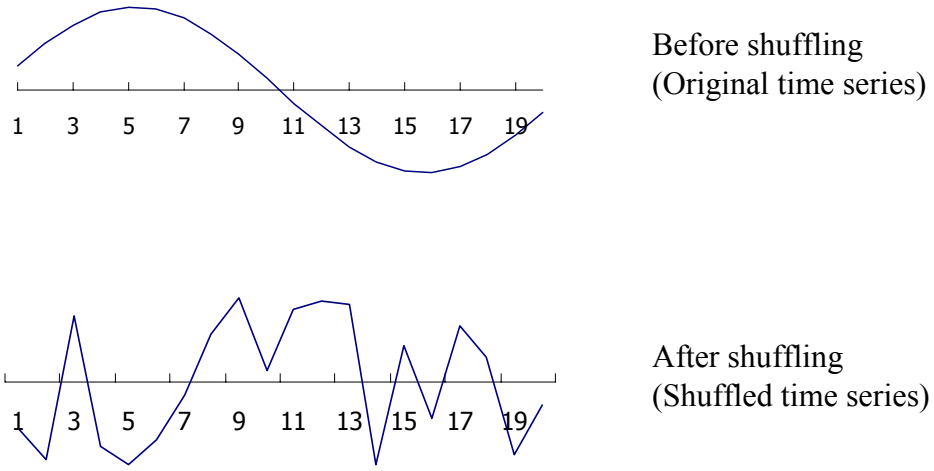

Figure 2.4: Shuffling a time series

The 1-step prediction error (measured by the sum of squared error (SSE) between the original time series and the modeled time series) before and after shuffling of a time series $Y = \{y_t, t = 1,2,..., N\}$, are compared. The 1-step prediction error in Y before shuffling is

$$SSE_Y = \sum_{t=1}^{N} (y_t - \hat{y}_t)^2 ,$$

where $\hat{y}_t$ is the predicted value of y_t. Shuffling increases the 1-step prediction error in Y. This is

$$SSE_S = \sum_{t=1}^{N} (S_t - \hat{S}_t)^2 ,$$

where S is the shuffled Y. Define

$$\eta = 1 - \frac{SSE_Y}{SSE_S}.$$

Thus, if time series Y is a totally deterministic signal and can be modeled perfectly, then $SSE_Y = 0$ and $\eta = 1$. If it is totally unpredictable, the reshuffling shouldn't affect the learned GP model accuracy, hence $SSE_Y = SSE_S$ and $\eta = 0$.

While applying Kaboudan's η-metric to estimate stock price predictability, two main problems have been observed.

First, the value of the metric depends on the length of the time series. The larger the sample size is, the higher the predictability Kaboudan's metric gives. Specifically, the η calculated for a 50-day stock price time series will be much larger than the η calculated from a 20-day stock price time series that is a subsequence of the 50-day series. This is inconsistent with prior work [2] that provides evidence that longer stock price time series are closer to a random walk than shorter ones. The source of this effect is mainly due to the nonstationarity of financial time series, and the nonstationarity becomes more evident as the sample size increases. This is because a longer financial time series is more likely to have a larger variance, and GP tends to give more "bad predictions" for a shuffled time series with larger variance. This way, a longer time series would yield larger SSE_S while the SSE_Y remains approximately unchanged, and therefore increases

$$\eta = 1 - \frac{SSE_Y}{SSE_S}.$$

The second problem is a derivation of the first one. Since the η increases when the time series is longer, and its value has an upper bound of one, the value of the η-metric will be distributed in a very narrow range, especially for a long-term stock price time series. Hence, the resolution of the η-metric is reduced. This can be clearly seen by examining a long random walk time series, which has an η close to 0.9 (See Chapter 4).

Since the random walk time series are expected to have lower predictability than most stock price time series (main experimental data in this research), the η-metric over stock price time series will be distributed in the approximate range of [0.9,1.0].

Chapter 3 Methods

As stated in the previous chapter, there are two main problems with Kaboudan's η-metric. First, the value of the metric largely depends on the length of the time series. Second, for a long-term stock series, the value of the η-metric will be distributed in a very narrow range. Hence, the resolution of the metric is limited. These two problems are resolved by the new η-metric presented in this chapter.

3.1 New η-metric

For a long-term time series

$$Y = \{y_t, t = 1,2,\cdots,N\},$$

the η-metric is calculated on the first Q points, that is, a sample series

$$\{y_t, t = 1,2,\cdots,Q\}.$$

Then, the sample series is shifted by τ, and the η-metric is calculated again on the new sample

$$\{y_t, t = 1+\tau,2+\tau,\cdots,Q+\tau\}.$$

Continuing this process, a series of η's is generated, which are the local predictability estimations of the subsequences of the time series. Generally, η_s^Q is defined as the η-metric over the sample

$$\{y_t, t = s-Q+1, s-Q+2,\cdots,s-1,s\}.$$

Thus, the η-series is represented by

$$\{\eta_Q^Q, \eta_{Q+\tau}^Q, \eta_{Q+2\tau}^Q, \cdots, \eta_{Q+m\tau}^Q, \cdots\}.$$

Since all the η's are estimated over same sample size Q, they are comparable, and by selecting appropriate values of window length Q, they can be made to distributed in a reasonable range. This completely solves the first problem (η depends on the length of time series) and partially solves the second problem (badly scaled and low resolution). Additionally, by examining the resulting η-series, the variation of the predictability over time can be observed, and the overall predictability of a specific time series can be estimated by calculating the average η over all windows.

To completely address the second problem, Kaboudan's definition of η is examined. His definition

$$\eta = 1 - \frac{SSE_Y}{SSE_S}$$

uses squared error, which makes the ratio of the prediction error between the original time series Y and the reshuffled version S fall into a narrow range for most financial time series. Since the original metric compared squared error, apply the square root operator to the error is a reasonable, and as will see later a successful approach. Simply modify the definition to

$$\eta = 1 - \sqrt{\frac{SSE_Y}{SSE_S}}$$

partially solves the low resolution problem.

A comparison of the new metric and Kaboudan's metric is shown in Figure 3.1 and Figure 3.2. It can be seen that using Kaboudan's metric, all of the 30 stocks fall into the 0.85 to 1 range. This makes the resolution of the metric very low, and it's hard to distinguish a stock price time series with a random walk time series (with predictability

value 0.875, see Chapter 4). Using the new metric, all the stocks are distributed in a wider range (0.3 – 0.7), and more predictable than a random walk time series (with predictability value 0.255, see Chapter 4).

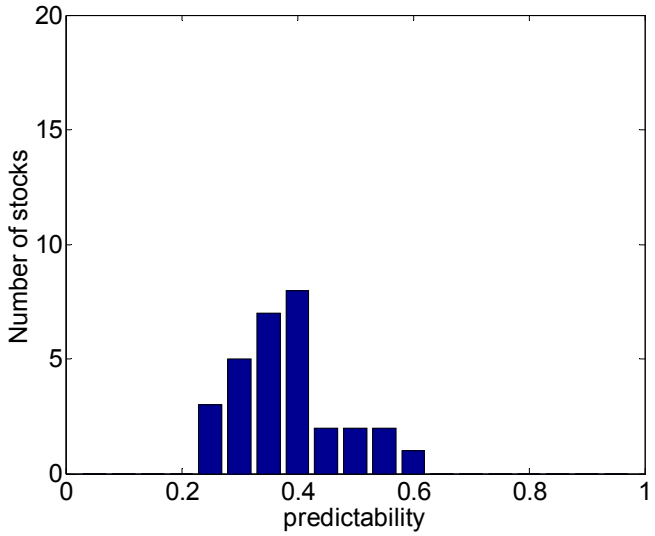

Figure 3.1: Distribution of new metric on the 30 DJI stocks

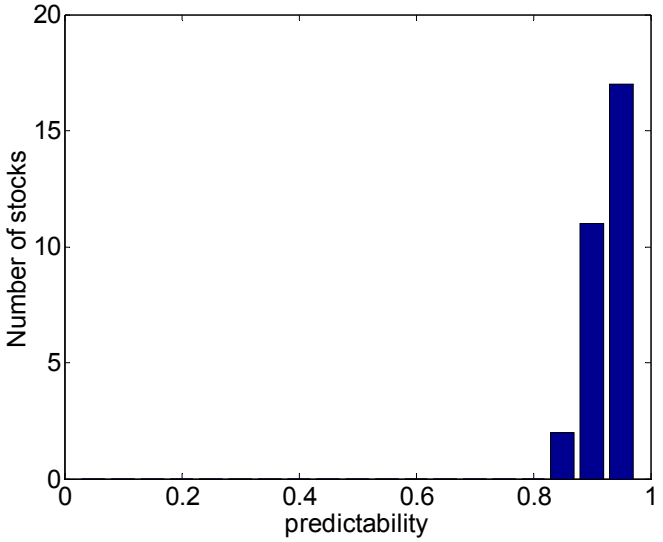

Figure 3.2: Distribution of Kaboudan's metric on the 30 DJI stocks

3.2 Combining η-metric with Genetic Programming

Two different evolutionary approaches, Genetic Programming (GP) and Fast Evolutionary Programming (FEP), are considered for use as modeling methods. A detailed comparison of these two methods is given in Chapter 6. It shows that GP has a better search ability than FEP, especially when dealing 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, its accuracy performance is similar to GP, but with much less computational effort.

However, in the financial applications, to which the η-metric is applied in this thesis, the accuracy performance is of much more concerned than is the time performance, as long as the results can be worked out within acceptable amount of time (for example, the time between today's closing and tomorrow's opening of the market). Therefore, GP is considered to be better in this particular application. FEP may be more useful in some other applications where the computation time is more important.

3.2.1 Model

The forecasting model is a regressive expression that takes the past values in a time series as the input and future values as the output. For example, Kaboudan concluded that stock prices p_t are mostly explained by the following ten variables (selected from five related time series):

$$p_{t-1}, p_{t-2}, p_{t-3}, hp_{t-1}, hp_{t-2}, lp_{t-1}, lp_{t-2}, vol_{t-1}, dji_{t-1}, dji_{t-2}$$

where *p* is the daily close price, *hp* and *lp* are the daily highest and lowest stock prices, respectively, *vol* is the daily traded volume of that stock, *dji* is the daily Dow Jones Industrial Average, and *t* is the time index.

Following this suggestion, these ten variables are used for GP to evolve the forecasting model using 1-step predicting shown in the following equation.

$$p_t = f(p_{t-1}, p_{t-2}, p_{t-3}, hp_{t-1}, hp_{t-2}, lp_{t-1}, lp_{t-2}, vol_{t-1}, dji_{t-1}, dji_{t-2})$$

GP searches for an optimal function *f* that gives the minimum prediction error over the training data. The function set provides all the mathematical operators used in *f* that combine those terminals. The *R* in the terminal set represents a random constant, which can form random floating point numbers between –1 and 1 in the function *f*. Following is an example of the resulting model

$$p_t = p_{t-1} - 0.1357\, p_{t-2} + 0.5365 vol_{t-1} / (hp_{t-1} - lp_{t-1}) + dji_{t-1}.$$

In most cases, the resulting GP equations are very complex and almost impossible to translate into humanly understandable relations between variables [4]. A complete GP configuration is given in the next section.

Another evolutionary algorithm called Fast Evolutionary Programming is considered as an alternative approach to model the time series in this thesis. The result of comparing this method and GP is discussed in Chapter 6.

3.2.2 Genetic Programming Settings

Adil Qureshi's GPsys release 2b [29] was used to perform all the GP runs. The configuration used in this study is given in Table 3.1.

Parameter	Value
Generations	100
Populations	1000
Function set	+, -, /, *, sin, cos, exp, sqrt, ln
Terminal set	$\{p_{t-1}, p_{t-2}, p_{t-3}, hp_{t-1}, hp_{t-2}, lp_{t-1}, lp_{t-2}, vol_{t-1}, dji_{t-1}, dji_{t-2}, R\}$
Fitness	Sum of squared error between predicted and actual points

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 3.1: GP configuration

3.3 Combining η-metric with Artificial Neural Network

Artificial Neural Networks have been widely recognized as an effective modeling method in financial market forecasting [1, 30-35]. It is used as an alternative modeling method in this thesis. This section describes the configurations of the network to be used.

3.3.1 Model

The same inputs and output are used in the neural network model as in the genetic programming model described in last section, i.e.

$$p_t = NN(p_{t-1}, p_{t-2}, p_{t-3}, hp_{t-1}, hp_{t-2}, lp_{t-1}, lp_{t-2}, vol_{t-1}, dji_{t-1}, dji_{t-2}).$$

Again, *p* is the daily close price, *hp* and *lp* are the daily highest and lowest stock prices, respectively, *vol* is the daily traded volume of that stock, *dji* is the daily Dow Jones Industrial Average, and *t* is the time index. The function *NN* represents the neural network system. It takes 10 ten past variables as the inputs and gives one single output as the prediction.

3.3.2 Neural Network Structure

A feed-forward backpropagation neural network is used in our problem. The network is created by using the MATLAB function "newff". For example, the following MABLAB code returns a two-hidden-layer feed-forward backpropagation network.

```
net = newff(PR,[3 3 1],{'logsig' 'logsig' 'purelin'});
```

The first parameter “PR” is a $R \times 2$ matrix of min and max values for R input elements (R equals 10 in our model). The second parameter “[3 3 1]” indicates that both hidden-layers contain 3 neurons, and the output layer contains a single neuron which gives a single output. The third parameter specifies the transfer functions for each layer, respectively.

Chapter 4 Examples

In this Chapter, the η-metric method is tested on several sample time series, including deterministic time series, white noise time series, deterministic plus noise time series, random walk time series and stock price time series. All the experiments conducted in this Chapter use GP as modeling method and use 10-step past window to perform 1-step ahead prediction, i.e., for a time series $\{y_t, t = 1, 2, \ldots, N\}$, GP is used to search the function *f* than minimize the 1-step prediction error for the model $y_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-10})$. Also, a linear predictor is used to evaluate the predictability metric on these sample time series as a comparison to GP. A discussion on this comparison will be given at the end of this chapter.

4.1 Deterministic Time Series

The Mackey-Glass equation is used to generate the deterministic time series in this study. The equation for the discretized map is [36]

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

where a=0.1, b=0.2, c=10, and τ=16. The Mackey-Glass map is seeded with 17 pseudo-random numbers and an 1100 points time series is generated. The first 1000 points are discarded to remove the initial transients. The last 100 points are used as the deterministic time series upon which the predictability metric is tested. The sample size is set to 100 for Kaboudan's method. For the new method, the sample size $Q = 20$ and the shift step $\tau = 5$. Results are shown in Table 4.1 and Table 4.2, where *LP* represents Linear Predictor, and *GP* represents Genetic Programming.

η_{LP}	η_{GP}
0.986	0.999

Table 4.1: Predictability of Mackey-Glass series using Kaboudan's -metric

τ	η_{LP}	η_{GP}
0	0.984	0.998
5	0.935	0.999
10	0.902	0.999
15	0.930	1.000
20	0.946	0.998
25	0.960	0.995
30	0.884	0.995
35	0.858	0.996
40	0.930	0.994
45	0.970	0.996
Average η	0.930	0.997

Table 4.2: Predictability of Mackey-Glass time series using the new metric

Both Kaboudan's metric and the new metric give an average η very close to 1, indicating that the time series is highly predictable. Note that the difference in SSE_S between Kaboudan's method and the new method presented in this paper is due to the length of the respective time series. Recall for Kaboudan's method the time series is 100 observations and for the new method each subsequence is 20 observations.

4.2 Random Walk Time Series

A random walk time series is generated and tested using both the Kaboudan's η-metric and the new metric. The random walk series

$$\{R_t\}, t = 1,2,\cdots,N,$$

is generated by

$$R_t = R_{t-1} + a_t,$$

where a_t is random variable uniformly distributed in [-0.5, 0.5], and the initial value $R_0 = 10$. Again, for Kaboudan's method, the sample size is 100, and for the new method, the sample size $Q = 20$ and the shift step $\tau = 5$. The results are shown in Table 4.3 and Table 4.4.

η_{LP}	η_{GP}
0.956	0.875

Table 4.3: Predictability of random walk series using Kaboudan's metric

τ	η_{LP}	η_{GP}
0	0.019	0.251
5	0.156	0.127
10	0.284	0.211
15	0.481	0.457
20	0.390	0.328
25	0.350	0.124
30	0.323	0.207
35	0.237	0.204
40	0.352	0.391
45	0.442	0.247
Average η	0.303	0.255

Table 4.4: Predictability of random walk series using the new metric

Kaboudan's metric gives $\eta = 0.875$ for a random walk series. As stated in section 2.5 and 3.1, this forces the predictability of most financial time series to be distributed in the narrow range of 0.875 to 1. The new metric gives an average $\eta = 0.255$, which gives a wider range for more predictable time series.

4.3 White Noise Time Series

A white noise time series was generated using a Gaussian random number generator with zero mean and variance 1. Results shown in Table 4.5 and Table 4.6.

η_{LP}	η_{GP}
0.017	0.051

Table 4.5: Predictability of white noise time series using Kaboudan's metric

τ	η_{LP}	η_{GP}
0	-0.292	-0.191
5	0.060	-0.61
10	0.151	-0.333
15	0.207	0.003
20	-0.148	0.291
25	0.085	-0.035
30	0.042	-0.082
35	0.183	0.225
40	-0.253	0.271
45	-0.301	0.184
Average η	-0.026	-0.028

Table 4.6: Predictability of white noise time series using the new metric

Following Kaboudan's suggestion, if $\eta < 0$, it is simple set equal to zero, indicating that the time series is not predictable. The predictability of the white noise time series is close to zero. This result matches the design goal very well.

4.4 Deterministic Plus Noise Time Series

In this example, noise is added to a deterministic time series, which is the Mackey-Glass time series used in section 4.1, with a signal-to-noise ratio (SNR) equal to 10 dB. See Table 4.8 for the results. As expected, the predictability of this time series is some value between 0 and 1.

η_{LP}	η_{GP}
0.905	0.887

Table 4.7: Predictability of deterministic plus noise time series using Kaboudan's metric

τ	η_{LP}	η_{GP}
0	0.483	0.445
5	0.368	0.682
10	0.332	0.133
15	0.380	-0.122
20	0.404	0.297
25	0.472	0.716
30	0.355	0.645
35	0.228	0.421
40	0.220	0.399
45	0.459	0.418
Average η	0.370	0.403

Table 4.8: Predictability of deterministic plus noise time series using the new metric

4.5 Stock Price Series

Next the new metric is applied to calculate the predictability of two stock price time series: Compaq Computer (CPQ) and General Electricity (GE) for the year 1999, with $Q = 20$ and $\tau = 5$. The results are shown in Table 4.9 - Table 4.12.

η_{LP}	η_{GP}
0.927	0.952

Table 4.9: Predictability of CPQ stock time series using Kaboudan's metric

τ	η_{LP}	η_{GP}
0	0.613	0.630
5	0.756	0.855
10	0.671	0.909
15	0.845	0.921
20	0.667	0.944
25	0.825	0.968
30	0.788	0.962
35	0.711	0.876
40	0.573	0.572
45	0.464	0.875
Average η	0.691	0.851

Table 4.10: Predictability of CPQ stock time series using new metric

η_{LP}	η_{GP}
0.826	0.867

Table 4.11: Predictability of GE stock time series using Kaboudan's metric

τ	η_{LP}	η_{GP}
0	0.437	0.704
5	0.136	0.120
10	0.222	0.189
15	0.140	0.041
20	0.511	0.681
25	0.741	0.868
30	0.633	0.738
35	0.563	0.729
40	0.557	0.566
45	0.451	0.681
Average η	0.439	0.532

Table 4.12: Predictability of GE stock time series using new metric

Using GP, the new metric gives average $\eta = 0.851$ for CPQ and $\eta = 0.532$ for GE. These η values are different from the ones obtained from the totally deterministic time series and the random walk time series. This result suggests that these stock price series is more predictable than random walk series, but less predictable than a deterministic time series. The new metric does disclose this difference and quantifies it.

It is noticed that over all these sample time series, the linear predictor gives similar predictability values to the values given by GP, though not identical. GP gives slight better prediction error than LP on average. Both methods give exactly the same order for these sample time series (ranked by their predictability values). This implies that the predictability metrics evaluated by these two methods are to some extent related to each other. To truly compare these results would need to do more experiments and a higher order statistical analysis such as a t-test.

Chapter 5 Applications

This chapter presents results found by applying the predictability metric presented in chapter 3 to an investment strategy in the stock market. Three different modeling methods – genetic programming (GP) [15], artificial neural network (ANN) [37], and time series data mining (TSDM) [5] are used to test the effectiveness of the new metric.

5.1 Financial Applications Using Predictability Metric and GP

The first section presents the configurations of the experiments. The second section gives a simple trading strategy to show the behavior of the new metric. The third section discusses the results and proposes an improved trading strategy, which uses the predictability metric more effectively. The fourth and fifth sections present the results applying the new strategy on the 30 Dow Jones Industrial stocks.

5.1.1 Experiment Configurations

The 30 Dow Jones Industry stocks data from 1999 is used in the experiments. Kaboudan [4] concluded that stock prices p_t are mostly explained by the following ten variables:

$$p_{t-1}, p_{t-2}, p_{t-3}, hp_{t-1}, hp_{t-2}, lp_{t-1}, lp_{t-2}, vol_{t-1}, dji_{t-1}, dji_{t-2},$$

where *p* is the daily close price, *hp* and *lp* are the daily highest and lowest prices, respectively, *vol* is the daily traded volume of that stock, and *dji* is the daily Dow Jones Industrial Average. Using Kaboudan's results, these ten variables are used for GP to evolve the forecasting model:

$$p_t = f(p_{t-1}, p_{t-2}, p_{t-3}, hp_{t-1}, hp_{t-2}, lp_{t-1}, lp_{t-2}, vol_{t-1}, dji_{t-1}, dji_{t-2}).$$

The training period for the GP is the past 50 days, i.e., the GP searches for a model that minimize the sum of squared prediction error over the past 50 days, and uses this model to predict the next day's price.

The predictability metric η_n, defined below, for a particular day *n* is estimated using the method described in section 3.1. An η-series is first calculated, with window size $Q = 20$ and shift step $\tau = 1$. The η-series is represented as:

$$\{\eta_{20}^{20}, \eta_{21}^{20}, \eta_{22}^{20}, \cdots, \eta_{n}^{20}, \cdots\}.$$

The metric is defined as

$$\eta_n = (\eta_n^{20} + \eta_{n-10}^{20} + \eta_{n-20}^{20} + \eta_{n-30}^{20})/4.$$

Since η_s^{20} is calculated from the sample $\{y_t, t = s-19, s-18, \cdots, s-1, s\}$, the data set used to calculate η_n would be $\{y_t, t = n-49, n-48, \cdots, n-1, n\}$, which is exactly the same set used to calculate the prediction. Thus, it is reasonable to use the predictability metric as an indicator of whether a prediction is reliable or not. Figure 5.1 shows this method graphically.

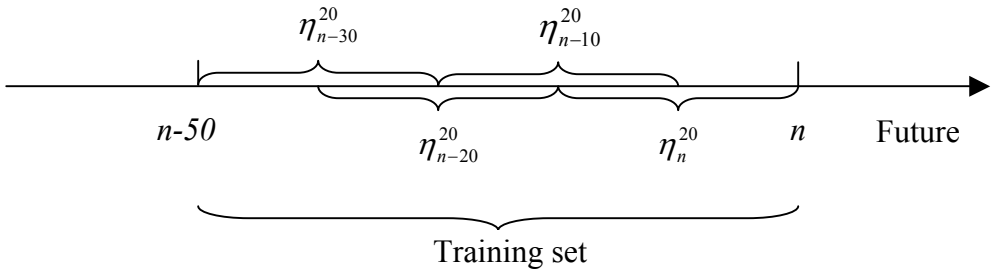

Figure 5.1: Calculating η

5.1.2 Behavior of the η-metric in Predicting Stock Market Returns

The behavior of the η-metric is obtained by comparing the performances of three different trading strategies. The three trading strategies are, buy and hold, trading based on the prediction of GP only, and trading based on both the GP prediction and the predictability metric.

The buy and hold strategy is straight forward, simply buy the stock on the first day of the trading period and sell it on the last day. It is equivalent to going long on all the trading days. Going long is a trading strategy in which people buy shares today, hoping to sell them tomorrow at a higher price and thus make a profit.

The second strategy (labeled "GP only") uses the GP's prediction to decide whether to go long or to go short. It goes long if GP predicts a price up, and goes short if GP predicts a price down. Going short is a strategy that involves selling shares you don't yet own in the expectation that the price will fall and you can buy them back at a lower price later (thus making a profit).

The third strategy (labeled "GP/η") is similar to the second one. The difference is that it only trades on those days in which the stock has a high predictability ($\eta > 0.6$), and does not trade on the other days. The reason for this is that a high predictability means a high confidence in the accuracy of the prediction; therefore only trading on these days can potentially reduce the risk and improve the return.

Results of the trading experiment are shown in Table 5.1. The 2nd, 3rd and 4th columns give the total return of 190 trading days using the three strategies: Buy and Hold, GP only and GP/η, respectively. The 5th column is the number of days in which a trade

is performed, using the third strategy. The 6th column gives the average predictability over all the 190 trading days.

The "GP only" strategy failed to defeat the simple "Buy and Hold" strategy in this experiment, but this does not necessarily mean that GP is useless. Since the "GP only" strategy tries to predict both price up and price down, and decides to go long or go short based on its prediction, it would work consistently independent on whether the market goes up or goes down. The "Buy and Hold" can only make profit in an ascending market, such as year 1999, and will lose money in a descending market.

Stock Name	Buy and Hold	GP only	GP/η	No. of trades	Average η
Aa	100.45%	30.60%	1.14%	5	0.366
Axp	41.03%	41.53%	0.00%	0	0.439
Ba	21.73%	11.39%	2.69%	9	0.315
C	29.97%	-24.64%	-0.12%	14	0.461
Cat	2.90%	-7.31%	-7.39%	21	0.328
Dd	15.22%	16.36%	-2.30%	35	0.445
Dis	-7.04%	20.36%	-2.46%	11	0.395
Ek	6.11%	28.33%	19.62%	61	0.488
Ge	41.53%	51.36%	1.34%	8	0.381
Gm	3.00%	-6.69%	6.79%	12	0.407
Hd	62.38%	-4.47%	7.19%	13	0.421
Hon	16.33%	-9.47%	-11.21%	50	0.482
Hwp	67.56%	-9.70%	10.08%	60	0.490
Ibm	22.99%	59.99%	16.39%	151	0.674
Intc	41.30%	7.47%	9.15%	35	0.466
Ip	35.65%	-39.27%	-3.91%	20	0.364
Jnj	-0.28%	-30.02%	14.00%	85	0.588
Jpm	-3.21%	-39.90%	2.23%	27	0.487
Ko	-3.23%	-15.25%	2.33%	50	0.483
Mcd	-11.31%	6.45%	-0.77%	1	0.344
Mmm	40.27%	-16.37%	2.83%	45	0.505
Mo	-30.90%	62.18%	15.09%	35	0.477
Mrk	-13.79%	-10.31%	-2.25%	116	0.612
Msft	31.59%	48.48%	10.89%	100	0.601
Pg	11.67%	31.96%	19.79%	83	0.580

Stock Name	Buy and Hold	GP only	GP/η	No. of trades	Average η
Sbc	2.10%	-25.02%	3.71%	4	0.413
t	-4.85%	4.00%	14.84%	18	0.444
utx	-5.24%	-0.06%	10.98%	53	0.522
wmt	46.61%	13.83%	2.54%	3	0.430
xom	16.33%	-7.72%	3.53%	14	0.397
average	19.23%	6.27%	4.89%	38	0.460
std	0.280902	0.282304	0.07807		0.087

Table 5.1: Total returns of the three different trading strategies

Also, it can be seen from the result that the second strategy (GP only) gives a higher average return than the third one. But if we look at the number of trades of both strategies, the third strategy (GP/η) has fewer trades than the second one. The number of trades here means the number of days in which a trading condition is hold, either going long or going short. Thus, the number of trades for the first two strategies will be 190, which is the total number of trading days. See Table 5.2 for the average return per trade of the three trading strategies. The average return per trade is obtained by calculating the geometric mean of the total return. The total return of the third strategy is less because it has a smaller number of trades compared with the other two strategies. Its average return per trade is actually higher. This implies that the third strategy has a potential to yield higher total return by performing more trades. A new trading strategy will be proposed in the next section based on this idea. Also, the third strategy gives a much lower variance, which means the risk is lower.

Stock Name	Buy and Hold	GP only	GP/η
aa	0.37%	0.14%	0.23%
axp	0.18%	0.18%	0.00%
ba	0.10%	0.06%	0.30%
c	0.14%	-0.15%	-0.01%
cat	0.02%	-0.04%	-0.36%

Stock Name	Buy and Hold	GP only	GP/η
dd	0.07%	0.08%	-0.07%
dis	-0.04%	0.10%	-0.23%
ek	0.03%	0.13%	0.29%
ge	0.18%	0.22%	0.17%
gm	0.02%	-0.04%	0.55%
hd	0.26%	-0.02%	0.54%
hon	0.08%	-0.05%	-0.24%
hwp	0.27%	-0.05%	0.16%
ibm	0.11%	0.25%	0.10%
intc	0.18%	0.04%	0.25%
ip	0.16%	-0.26%	-0.20%
jnj	0.00%	-0.19%	0.15%
jpm	-0.02%	-0.27%	0.08%
ko	-0.02%	-0.09%	0.05%
mcd	-0.06%	0.03%	-0.77%
mmm	0.18%	-0.09%	0.06%
mo	-0.19%	0.25%	0.40%
mrk	-0.08%	-0.06%	-0.02%
msft	0.14%	0.21%	0.10%
pg	0.06%	0.15%	0.22%
sbc	0.01%	-0.15%	0.92%
t	-0.03%	0.02%	0.77%
utx	-0.03%	0.00%	0.20%
wmt	0.20%	0.07%	0.84%
xom	0.08%	-0.04%	0.25%
average	0.08%	0.01%	0.16%

Table 5.2: Average return per trade of the three different trading strategies

Another noticeable result in Table 5.1 is that the total return of the third strategy has a high relationship to the average η over the whole trading period. Figure 5.2 shows a plot of the total return vs. the average η. Each point in the plot represents a particular stock. It can be seen clearly from the plot that the two variables are positively related, i.e., higher predictability correspond to higher return. The slope of the linear fitting line is

positive. This result can also be shown by calculating the correlation coefficient between these two variables, which is 0.5232.

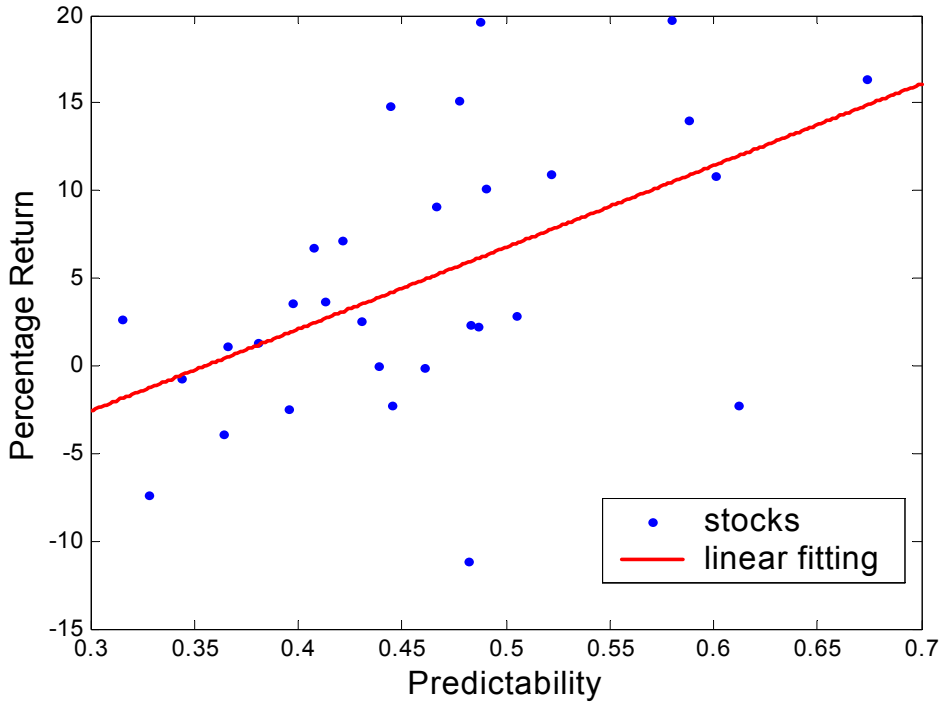

Figure 5.2: Plot of total return vs. predictability

5.1.3 Trading Strategy

In the previous section, a simple trading strategy using both GP prediction and the predictability metric is proposed. This trading strategy gives a total return lower than using GP only, but an important feature of this strategy is observed. That is, the lower total return is due to fewer number of trades; the average return per trade of this strategy is actually higher. This happens because for a particular stock, the predictability changes from day to day, sometimes it is high, and sometimes it is low. In the previous strategy, no trade is performed in those days with low predictability to improve the return. The idea of the new strategy is to look at many stocks for each trading day. For example, if

we look at 30 stocks in one particular day, 20 of them may have low predictabilities, but there is a good chance that several stocks with fairly high predictabilities can be found. Investors can put their money in those stocks with the highest predictabilities. Thus, the number of trading times increases while the advantage of the high average return per trade shown in the previous strategy still being hold. Based on this consideration, a new improved trading strategy is implemented as follows.

1. *Choose a set of stocks to be traded on.*
2. *For each stock, calculate its η-metric.*
3. *Select N stocks that have the highest η to trade.*
4. *Invest equally on the N stocks. Use GP's prediction to decide whether to go long or to go short for each stock with high predictability for the current trading day.*

New experiments are done on the 30 Dow Jones Industry stocks using the same configurations again. The number of stocks to be traded, *N*, is set from 1 to 30, and the total returns are calculated for each *N*. Results are shown in Figure 5.3**.**

Ideally, if the metric is a perfect predictability measure, this figure should show a set of monotonically decreasing bars, and only to trade on the most predictable stock (*N*=1) should give the highest return. But in practice this is not realistic, since stock time series are so complex that GP could not capture all the information underlying these time series. Statistical errors also distort the structure of the figure. From the plot, it can be seen that on the whole, selecting high predictable stocks based on our predictability metric to trade gives higher return than trading on all stocks. For example, trading on the top 10 high predictable stocks gives more than twice the return than trading on all the 30 stocks.

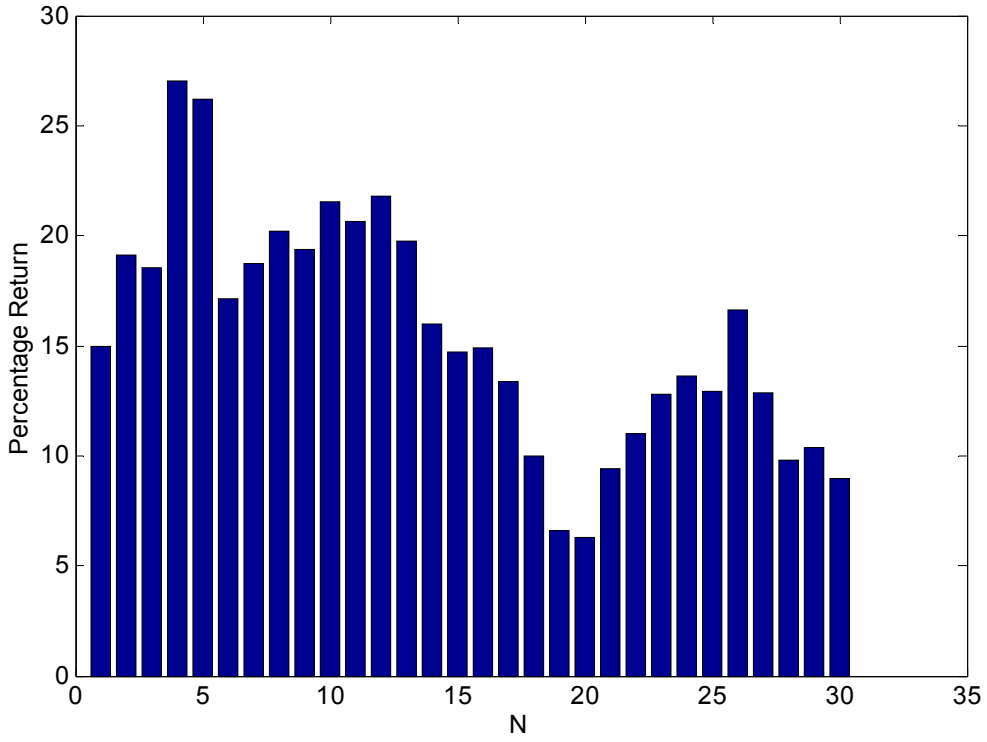

Figure 5.3: Results of the improved trading strategy using GP

5.2 Financial Applications Using Predictability Metric and ANN

A similar set of experiments is conducted using an ANN instead of a GP in this section, and similar results are observed.

5.2.1 Experiment Configurations

The same inputs and output are used for an ANN as for a GP, i.e.,

$$p_t = NN(p_{t-1}, p_{t-2}, p_{t-3}, hp_{t-1}, hp_{t-2}, lp_{t-1}, lp_{t-2}, vol_{t-1}, dji_{t-1}, dji_{t-2}).$$

Recall that when applying GP and Kaboudan's original η-metric to a long time series, the metric are usually badly scaled because of the nonstationery of the time series discussed in section 1.1. A new metric was designed to fix this problems by dividing the

training set into smaller sections and calculating the average predictability over these smaller set.

For ANN the problem is simpler because when the training set size is less than 50, the experiment results showed that the η-metric is not sensitive to whether the time series is stationary, i.e., the value of the η-metric would not depends on the length of the time series. Thus, there is no need to divide the 50 past data into smaller set. Kaboudan's original η-metric is good enough in this particular problem.

The training period is set to the past 50 days, i.e., the ANN is trained to find the best fit model that minimize the sum of squared prediction error over the past 50 days, and uses this model to predict the next day's price.

A feed-forward backpropagation neural network containing two hidden layers, each consisting of 3 neurons, and a output layer that has a single output neuron is trained. The network is created using the following MATLAB code:

```
net = newff(PR,[3 3 1],{'logsig' 'logsig' 'purelin'});
```

where PR is a matrix specifying the boundaries of the inputs, and {'logsig' 'logsig' 'purelin'} specifies the transfer functions of the two hidden layers and the output layer respectively.

5.2.2 Experiment results

The three similar trading strategies, i.e., buy and hold, using ANN only, and using ANN and η, are used to conduct the experiments. Results of the trading experiment are shown in Table 5.3. The 2nd, 3rd and 4th column give the total return of 190 trading days using those three trading strategies, respectively. The 5th column is the number of days in

which a trade is performed, using the third strategy. The 6th column gives the average predictability over all the 190 trading days.

Stock Name	Buy and Hold	ANN only	ANN/ η	No. of trades	Average η
Aa	100.45%	-19.56%	16.26%	82	0.5542
Axp	41.03%	-23.34%	-8.59%	12	0.3467
Ba	21.73%	5.73%	-2.38%	38	0.4074
c	29.97%	82.53%	6.58%	58	0.4868
cat	2.90%	7.78%	-0.38%	32	0.3747
dd	15.22%	-30.33%	0.87%	20	0.3757
dis	-7.04%	-16.83%	10.11%	80	0.561
ek	6.11%	67.82%	41.66%	60	0.4799
ge	41.53%	-20.82%	2.04%	40	0.4326
gm	3.00%	37.98%	0.89%	6	0.338
hd	62.38%	-51.18%	-4.22%	4	0.3382
hon	16.33%	47.77%	6.65%	13	0.3609
hwp	67.56%	44.28%	5.62%	19	0.367
ibm	22.99%	21.97%	1.26%	15	0.3536
intc	41.30%	12.25%	17.42%	6	0.3424
ip	35.65%	59.80%	14.04%	23	0.361
jnj	-0.28%	69.37%	-0.01%	2	0.3042
jpm	-3.21%	13.34%	0.00%	0	0.2507
ko	-3.23%	6.66%	0.17%	24	0.3977
mcd	-11.31%	-11.16%	-3.86%	26	0.435
mmm	40.27%	25.18%	19.31%	26	0.3318
mo	-30.90%	82.15%	59.52%	56	0.4993
mrk	-13.79%	-30.09%	-13.98%	21	0.3422
msft	31.59%	-17.17%	-2.70%	20	0.3829
pg	11.67%	19.28%	-3.94%	1	0.2605
sbc	2.10%	52.74%	29.61%	38	0.4357
T	-4.85%	94.46%	12.88%	27	0.3944
Utx	-5.24%	113.37%	12.75%	25	0.3865
Wmt	46.61%	21.58%	13.74%	22	0.3851
Xom	16.33%	3.61%	-6.27%	11	0.3576
Average	19.23%	22.31%	7.50%	26.9	0.388
Std	0.280902	0.4186	0.1522		0.087

Table 5.3: Total returns of the three different trading strategies

It can be seen from the result that the "ANN only" strategy gives a higher average return than the "ANN/η" strategy. But if we look at the number of trades of both strategies, the "ANN/η" strategy has fewer number of trades (26.9) than the "ANN Only" strategy (190). The number of trades here means the number of days in which a trading condition is hold, either going long or going short. Thus, the number of trades for the first two strategies will be 190, which is the total number of trading days. See Table 5.4 for the average return per trade of the three trading.

Stock Name	Buy and Hold	ANN Only	ANN/η
aa	0.37%	-0.11%	0.18%
axp	0.18%	-0.14%	-0.75%
ba	0.10%	0.03%	-0.06%
c	0.14%	0.32%	0.11%
cat	0.02%	0.04%	-0.01%
dd	0.07%	-0.19%	0.04%
dis	-0.04%	-0.10%	0.12%
ek	0.03%	0.27%	0.58%
ge	0.18%	-0.12%	0.05%
gm	0.02%	0.17%	0.15%
hd	0.26%	-0.38%	-1.07%
hon	0.08%	0.21%	0.50%
hwp	0.27%	0.19%	0.29%
ibm	0.11%	0.10%	0.08%
intc	0.18%	0.06%	2.71%
ip	0.16%	0.25%	0.57%
jnj	0.00%	0.28%	-0.01%
jpm	-0.02%	0.07%	0.00%
ko	-0.02%	0.03%	0.01%
mcd	-0.06%	-0.06%	-0.15%
mmm	0.18%	0.12%	0.68%
mo	-0.19%	0.32%	0.84%
mrk	-0.08%	-0.19%	-0.71%
msft	0.14%	-0.10%	-0.14%
pg	0.06%	0.09%	-0.22%
sbc	0.01%	0.22%	0.68%
t	-0.03%	0.35%	0.45%

Stock Name	Buy and Hold	ANN Only	ANN/η
utx	-0.03%	0.40%	0.48%
wmt	0.20%	0.10%	0.59%
xom	0.08%	0.02%	-0.59%
average	0.08%	0.07%	0.18%

Table 5.4: Average return per trade of the three different trading strategies

New experiments are conducted on the 30 Dow Jones Industrial stocks using ANN and the following trading strategy (same as described in section 5.1.3),

1. *Choose a set of stocks to be traded on.*
2. *For each stock, calculate its η-metric.*
3. *Select N stocks that have the highest η to trade.*
4. *Invest equally on the N stocks. Use GP's prediction to decide whether to go long or to go short for each stock with high predictability for the current trading day.*

The results are shown in Figure 5.4, where *N* is the number of stock to be traded. The total returns are calculated for each *N* from 1 to 30.

Again, conclusion can be drawn from the plot that selecting high predictable stocks based on the new predictability metric to trade gives higher return than trading on all stocks. ANN gives higher return than GP on average and the shape of the ANN plot is also closer to the ideal case (in which the bars should be monotonically decreasing). These results may imply that ANN has better search ability than GP in the application of stock market predicting. More experiments and statistical analysis need to be done to verify this conclusion.

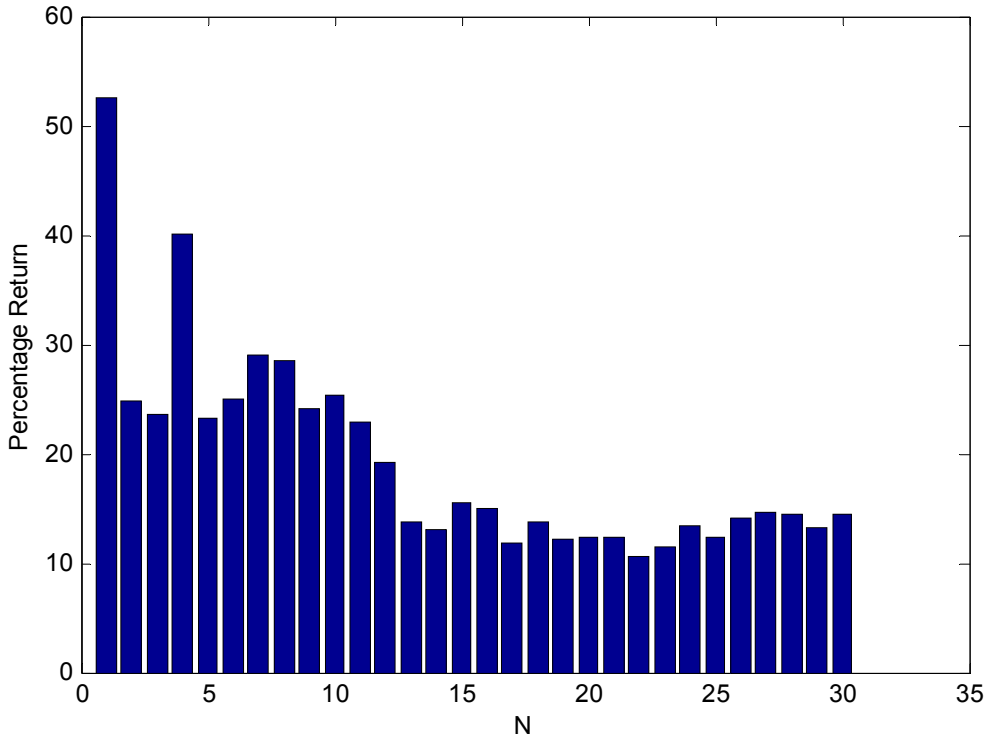

Figure 5.4: Trading results using ANN

Chapter 6 Attempts Using Other Modeling Approaches

This chapter presents some other nonlinear time series modeling approaches that have been tested in this research. As mentioned in Chapter 3, FEP may be an alternative approach which can be used in calculating the predictability metric. The TSDM method, however, shows no improvement when combine with time series predictability approaches.

6.1 Fast Evolutionary Programming (FEP)

Rao and Chellapilla [38] proposed an alternative modeling approach called fast evolutionary programming (FEP) to optimize the parameters of a reduced parameter bilinear model (RPBL). The RPBL model [21] is capable of effectively representing nonlinear models with the additional advantage of using fewer parameters than a conventional bilinear model. FEP, which can be used to determine RPBL model structure, is shown here in this section to have reasonable optimization performance. In comparison with conventional evolutionary programming, FEP evolves RPBL models with lower normalized mean squared error (NMSE) and also lower model order. This approach will be shown to have less computational cost and less model complexity when compared with GP. However, FEP prediction accuracy is lower than GP.

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.

6.1.1 Experiments and results

6.1.1.1 The Mackey-Glass Time Series

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 point 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 6.1. Results from GP and FEP are shown in Table 6.1.

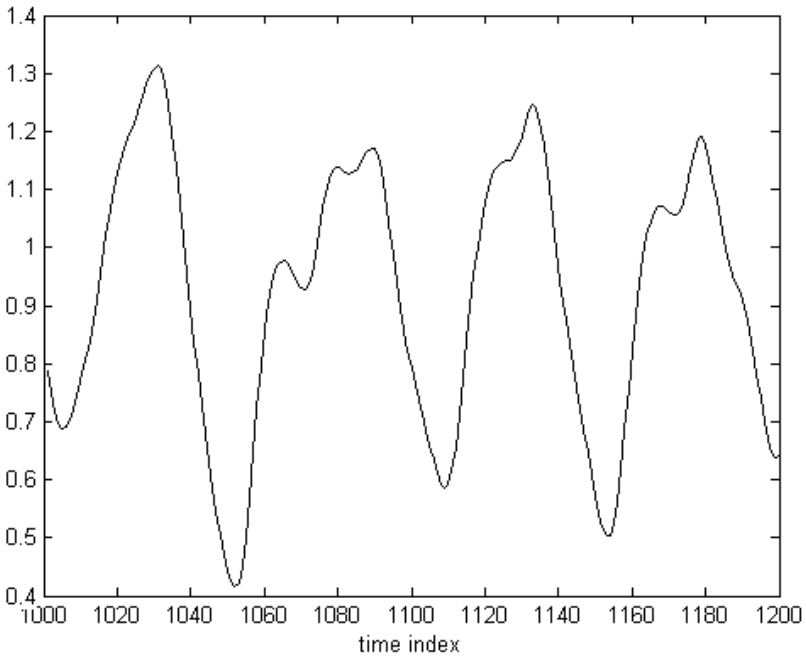

Figure 6.1: Mackey-Glass map

	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 6.1: Results for the Mackey-Glass time series

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. The large difference between training MSE and Test MSE of FEP shows that FEP is badly over-trained in this case. 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.

6.1.1.2 The Sunspot Time Series

The second experiment was conducted on the yearly sunspot series for the years 1800-1999 [39], see Figure 6.2. Once again, the first 100 data points are used as training set and the next 100 points are used for testing. The results are given in Table 6.2.

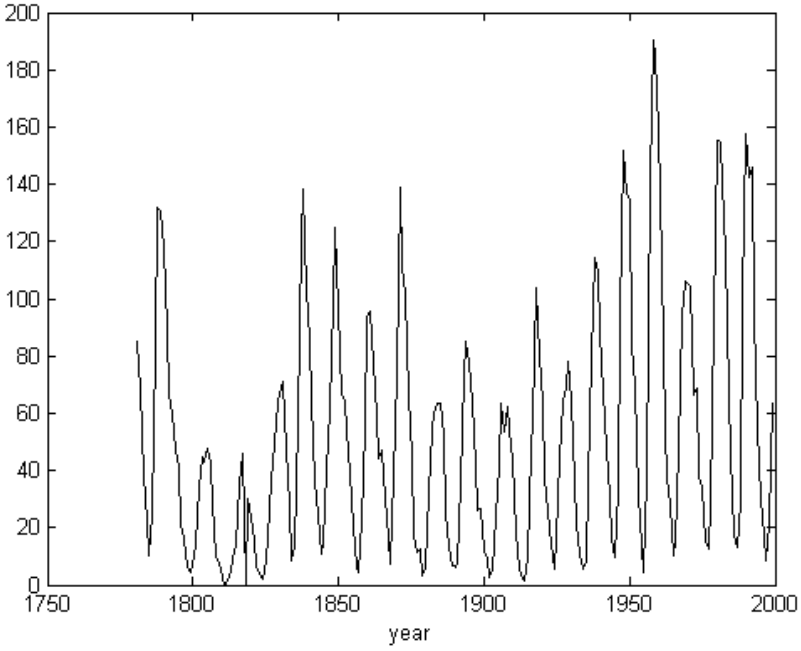

Figure 6.2: Sunspot time series

	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 6.2: Results for the sunspot time series

In modeling the sunspot time series, the accuracy performance between the two methods is similar. The GP gives slightly better accuracy, but again, it takes three times as long to compute as the FEP method.

6.1.1.3 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 6.3 and Figure 6.4. Table 6.3 and Table 6.4 present the modeling results.

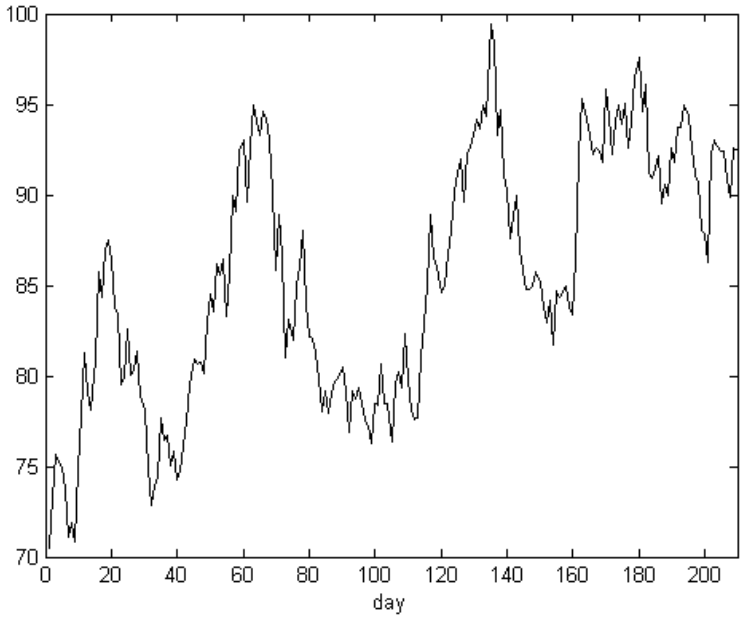

Figure 6.3: MSFT price time series

	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 6.3: Results for the MSFT time series

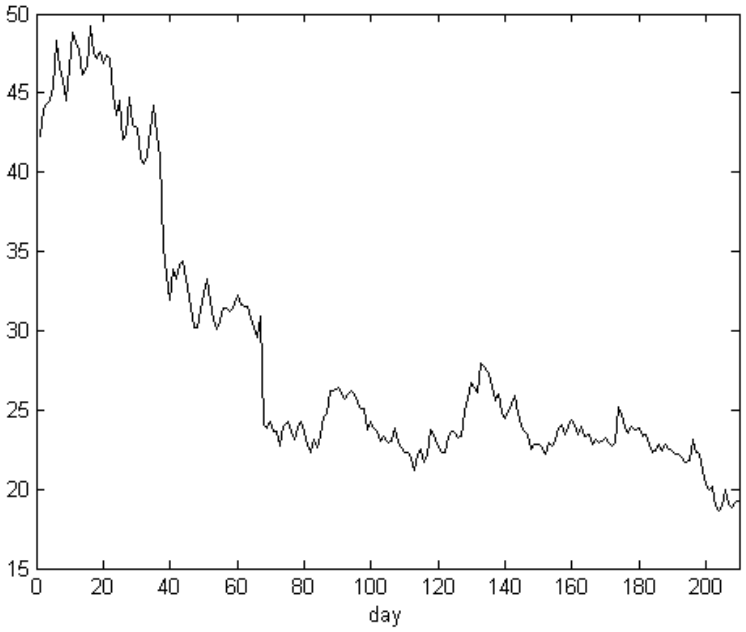

Figure 6.4: CPQ price time series

	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 6.4: Results for the CPQ 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 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 generations increases, the models evolved by FEP tend to become simpler while those evolved by GP always become more complex (measured by the total number of nodes in the GP tree). This explains why GP is not as consistent as FEP. As the GP runs the learned model becomes more complex. This means that more of the solution space is being explored. Note the space of functions explored by the GP is much larger than the function space searched by the FEP. Thus as the GP runs it will encounter more local minimum in each generation. In the experiments, the best solution is always found by GP. This also suggests that GPs have relatively stronger search ability.

In these two stock time series, FEP shows better performance in both accuracy and computation time than GP. But as mentioned previously, the solutions found by GP have a fairly large variance compared with FEP. This is because that GP is more likely to fall into a local minimum and generate poor solutions. The results of GP could be improved further with throwing away these bad solutions. To demonstrate this, the 50% solutions that have low training MSE are kept for testing, and the remaining 50% of the high error solutions are discarded. The results after this process are shown below in Table 6.5. It can be seen that GP has better accuracy performance that FEP.

	GP	FEP
MSFT	5.368×10^{-3}	6.280×10^{-3}
CPQ	2.092×10^{-3}	2.100×10^{-3}

Table 6.5: Test MSE by averaging best 50% solutions

6.1.2 Conclusion

In this section, 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 dealing with more predictable time series. FEP performs better when applied to noisier time series. For real world time series such as sunspot time series and stock price time series, its accuracy performance is similar to GP, but with less computational effort.

However, in the financial applications showed in chapter 5, the accuracy performance is much more important than computational performance. Therefore, GP is considered to be a better modeling approach in this particular application. FEP may be more useful in some other applications where the computation time is more important.

6.2 Time Series Data Mining (TSDM)

Povinelli introduced a new framework for analyzing time series data called Time Series Data Mining (TSDM) [5, 26, 27]. This framework adapts and innovates data mining concepts to time series analysis. Unlike most of the other time series analysis methods that try to characterize and predict all time series observations, TSDM methods focus on characterizing and predicting events. Therefore, it does not require the time series to be stationary. It also overcomes the limitations of traditional methods of requiring normality and independence of the residuals in the time series.

In the previous work [5], the TSDM method has been shown to be able to effectively recognize patterns contained in stock price time series, i.e., the patterns found in the

training time series also exist in the test time series, but it was noticed that TSDM could also find patterns in a pure noise time series in the training stage. This result has been shown empirically in this thesis. This fact leads to the result that the TSDM method found events in the reshuffled time series no worse than in the original time series. For example, in the AXP stock time series from 1999, the TSDM method found 20 price-up events that have an average return of 0.63% (Table 6.6). In the reshuffled version of this time series, the TSDM method found 23 price-up events that have an average return of 17.6% (Table 6.7). Results from other stock time series are similar. This makes the η-metric not applicable to the TSDM method because it can hardly tell the differences between the original time series and the reshuffled one. Some other metric is needed for this method.

Index	Return
12	2.29%
17	5.49%
22	-3.35%
39	-1.18%
43	0.81%
63	2.05%
74	1.43%
89	-2.01%
99	-1.94%
103	3.69%
111	-1.28%
119	1.52%
121	-0.63%
122	1.72%
134	-2.45%
143	2.97%
145	-1.53%
180	2.67%
187	4.10%
190	-1.76%

Average	0.63%

Table 6.6: TSDM predictions for AXP 1999

Index	Return
8	−1. 20%
13	24. 6%
19	43. 5%
26	7. 57%
57	49. 5%
68	16. 0%
85	44. 2%
88	67. 6%
103	49. 1%
107	3. 43%
112	38. 0%
116	−3. 47%
122	45. 2%
135	−32. 3%
136	48. 2%
140	−2. 99%
144	45. 8%
151	42. 8%
153	28. 6%
157	−32. 6%
163	−42. 8%
170	0. 20%
171	−33. 7%
Average	0. 63%

Table 6.7: TSDM predictions for reshuffled AXP 1999

The second attempt is to use the probability value α as a possible predictability metric, where α is the probability to reject the test hypothesis that the set of eventnesses associated with the temporal pattern cluster is different from the set of eventnesses not associated with the temporal pattern cluster [5]. This attempt failed again since the experiment result showed that over the 30 Dow Jones Industry stocks in year 1999, the

α value and the return of the investment have a correlation coefficient of –0.028 (523 predictions, see Table 6.8), which indicates that these two variables do not have much relationship with each other. Therefore, the α value cannot be used as an indicator to predict the return.

Index	α	Return
1	0.0198	−5.41%
2	0.0131	−5.12%
3	0.0559	−4.21%
4	0.0696	−3.49%
5	0.0903	−3.09%
6	0.0130	−2.83%
7	0.0003	−1.89%
8	0.0005	−1.38%
9	0.1270	−4.70%
10	0.0007	−1.87%
…		
520	0.0000	2.31%
521	0.0588	2.73%
522	0.0000	3.28%
523	0.0777	4.03%

Table 6.8: TSDM results: α vs. return

A possible reason for the above two failures may be due to the fact that the TSDM method only tries to recognize patterns in a time series and predict the events following these patterns. It is not trying to predict the data that it cannot recognize. This is very similar to the way that the predictability metric works. In other words, the TSDM method already takes advantage of some kind of predictability information of a time series, and thus the attempt of trying to build another predictability metric on top of it could not do any better.

Chapter 7 Conclusions and Future Work

This thesis makes an original contribution to the field of time series analysis and forecasting by developing a new time series predictability metric and studying its applications in financial time series forecasting. This new time series predictability metric was developed based on the η-metric method introduced by Kaboudan [4], but overcomes the two main disadvantages of the pure η-metric method. It also provides a new feature, which shows how the predictability changes over different subsequences in a time series.

The new metric can be built on top of many time series modeling methods and improves their performance in time series forecasting. Successful attempts have been made with Genetic Programming (GP) and Artificial Neural Networks (ANN) in the application of stock time series prediction.

This thesis has demonstrated that the new metric has successfully determined the difference between different kinds of time series including deterministic time series, white noise time series, deterministic plus noise time series, random walk time series and stock price time series. This test shown in Chapter 4 validates that this metric does have the capability to discover the predictability information underlying a given time series. This feature is used in Chapter 5 to develop a new stock trading strategy, which evaluates the predictability metric for a set of stocks, and trades on those stocks with relatively high predictability. The results showed that combining the predictability metric and time series modeling technique generate better return than without using the predictability metric.

Besides GP and ANN, two other modeling techniques, Fast Evolutionary Programming (FEP) and Time Series Data Mining (TSDM), were considered as modeling methods. Chapter 6 shows that FEP has worse accuracy performance than GP, and there is no good way to combine TSDM with the predictability metric. Therefore, these two techniques were not used in the trading experiments.

Possible future work of this research includes more robust statistical analysis of the results, study of the η-metric for other time series modeling techniques, further empirical studies, and theoretical evaluation of the metric. By doing these researches, the current predictability metric may be generalized so that it does not only apply for one specific modeling method.

References

[1] C. H. Chen, “Neural networks for financial market prediction,” proceedings of IEEE International Conference on Neural Networks, 1994, pp. 1199-1202.

[2] S.-H. Chen, “Genetic programming and the efficient market hypothesis.,” proceedings of Genetic Programming 1996: Proceedings of the First Annual Conference, Cambridge, MA, 1996, pp. 45-53.

[3] D. Fogel, “Preliminary experiments on discriminating between chaotic signals and noise using evolutionary programming.,” proceedings of Genetic Progamming 1996: Proceedings of the First Annual Conference., 1996, pp. 512-520.

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

[5] R. J. Povinelli, *Time Series Data Mining: Identifying Temporal Patterns for Characterization and Prediction of Time Series Events*, Ph.D. Dissertation, Marquette University, 1999.

[6] M. Kaboudan, “A Measure of Time-Series' Predictability Using Genetic Programming Applied to Stock Returns,” *Journal of Forecasting*, vol. 18, pp. 345-357, 1999.

[7] B. L. Bowerman and R. T. O'Connell, *Forecasting and time series: an applied approach*, 3rd ed. Belmont, California: Duxbury Press, 1993.

[8] H. Kantz and T. Schreiber, *Nonlinear time series analysis*. Cambridge: Cambridge University Press, 1997.

[9] C. Jacob, *Illustrating Evolutionary Computation with Mathematica*. San Diego: Morgan Kaufmann Publishers, 2001.

[10] W. G. M. David H. Wolpert, "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67-82, 1997.

[11] B. Freisleben, "Stock Market Predictioin with Backpropagation Networks," *Lecture notes in computer science*, vol. 604, pp. 451-460, 1992.

[12] G. E. P. Box and G. M. Jenkins, *Time series analysis: forecasting and control*, Rev. ed. San Francisco: Holden-Day, 1976.

[13] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, 1st MIT Press ed. Cambridge, Massachusetts: MIT Press, 1992.

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

[15] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming ~ An Introduction*

- *On the Automatic Evolution of Computer Programs and Its Applications*. San Francisco: Morgan Kaufmann Publishers, 1998.

[16] E. Howard and N. Oakley, "The Application of Genetic Programming to the Investigation of Short, Noisy, Chaotic Data Series," proceedings of AISB Workshop, Leeds, U.K., 1994, pp. 320-332.

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

[18] Akaike, "A new look at the statistical model identification," *IEEE Trans. Auto. Control*, vol. AC-19, pp. 716-723, 1974.

[19] S. S. Rao, "Evolving reduced parameter bilinear models for time series prediction using fast evolutionary programming.," proceedings of Genetic Programming 1996: Proceedings of the First Annual Conference, Cambridge, MA, 1996, pp. 528-535.

[20] X. Yao, "Fast evolutionary programming," proceedings of Evolutionary Programming V: Proceedings of 5th Annual Conference on Evolutionary Programming., Cambridge, MA, 1996, pp. 451.

[21] Y. Zhang, "A Reduced Parameter Bilinear Time Series Model.," *IEEE Trans. Signal Processing*, vol. 42, pp. 1867-1870, 1994.

[22] A. Rao, D. Miller, K. Rose, and A. Gersho, "Deterministically Annealed Mixture of Experts Models for Statistical Regression," proceedings of Proc. ICASSP, 1997, pp. 3201-3204.

[23] I. Rechenberg, *Evolution strategy: Nature's way of optimization. In Optimization: Methods and Applications, Possibilities and Limitations.* Berlin: Springer-Verlag, 1989.

[24] L. J. Fogel, *Artificial Intelligence Through Simulated Evolution*. New York: John Wiley and Sons., 1966.

[25] D. Fogel, "An introduction to sumulated evolutionary optimisation," *IEEE Trans. On Neural Networks*, vol. 5, pp. 3-14, 1994.

[26] R. J. Povinelli, "Identifying Temporal Patterns for Characterization and Prediction of Financial Time Series Events," proceedings of International

Workshop on Temporal, Spatial and Spatio-Temporal Data Mining: TSDM2000, Lyon, France, 2000, pp. 46-61.

[27] R. J. Povinelli, "Characterization and Prediction of Welding Droplet Release using Time Series Data Mining," proceedings of Artificial Neural Networks in Engineering, St. Louis, Missouri, 2000, pp. 857-862.

[28] B. Efron, *The Jackknife, the Bootstrap, and Other Resampling Plans.* Philadelphia: Society for Industrial and Applied Methematics, 1982.

[29] A. Qureshi, "GPsys," available at http://www.cs.ucl.ac.uk/staff/A.Qureshi/gpsys.html, 2000, cited 2000.

[30] T. Ankenbrand and M. Tomassini, "Predicting multivariate financial time series using neural networks: the Swiss bond case," proceedings of IEEE/IAFE Conference on Computational Intelligence for Financial Engineering, 1996, pp. 27-33.

[31] Y. Bentz, L. Boone, and J. Connor, "Modeling stock return sensitivities to economic factors with the Kalman filter and neural networks," proceedings of IEEE/IAFE Conference on Computational Intelligence for Financial Engineering, 1996, pp. 79-82.

[32] C.-F. Chang, b. J. Sheu, and J. Thomas, "Multi-Layered Back-Propagation Neural Networks for Finance Analysis," proceedings of World Congress on Neural Networks, Portland, Oregon, 1993, pp. 445-450.

[33] G. Deboeck, *Trading on the edge: neural, genetic, and fuzzy systems for chaotic financial markets*. New York: Wiley, 1994.

[34] B. Freisleben and K. Ripper, "Economic forecasting using neural networks," proceedings of IEEE International Conference on Neural Networks, 1995, pp. 833-838.

[35] R. N. Kahn and A. K. Basu, "Neural networks in finance: an information analysis," proceedings of IEEE/IAFE Conference on Computational Intelligence for Financial Engineering, 1995, pp. 183-191.

[36] M. C. Mackey and L. Glass, "Oscillations and chaos in physiological control systems," *Science*, vol. 197, pp. 287, 1977.

[37] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Networks Design*. Boston, MA: PWS Publishing Company, 1995.

[38] S. S. Rao and K. Chellapilla, "Evolving Reduced Parameter Bilinear Models for Time Series Prediction using Fast Evolutionary Programming," proceedings of Genetic Programming: proceedings of the first annual conference, Stanford, CA, 1996, pp. 528-535.

[39] "Sunspot archive," available at http://sidc.oma.be/index.php3, 2002, cited 2000.