# The role of predictability of financial series in emerging market applications

**Gabriela PRELIPCEAN**
**Faculty of Economic and Public Administration**
**Stefan cel Mare University of Suceava**
**Suceava, Romania, e-mail: gprelipcean@yahoo.com**

**Mircea BOSCOIANU**
**Military Technical Academy of Bucharest**
**Romania, e-mail: mircea_boscoianu@yahoo.co.uk**

**Nicolae Popoviciu**
**Faculty of Mathematics-Informatics**
**Hyperion University of Bucharest**
**Bucharest, Romania, e-mail nicolae.popoviciu@yahoo.com**

**Abstract**: A new metric that quantifies the predictability of financial time series is proposed. Time series predictability provides a measure of how well a time series can be modeled by a particular method, or how well a prediction can be made. This new time series predictability metric is developed based on the Kaboudan η –metric. The new metrics, based on Genetic Programming (GP) and Artificial Neural Networks (ANN) overcomes the stationarity problem presented in the pure η -metric and provides a new feature, which shows how the predictability changes over different subsequences in a time series. Timing detection and portfolio balancing should be based on trading strategies that evolved to optimize buy/sell decisions. The interest is to explore new trading rules based on an automated security trading decision support system triggered by both quantitative and qualitative factors. The focus 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.

**Keywords**: quantitative metrics, predictability, timing detection, portfolio selection, Genetic Programming (GP), Artificial Neural Networks (ANN).

## 1. Introduction in the predictability of financial time series

*Time series predictability* is a measure of how well future values $y_t$ can be forecasted and indicates to what extent the past can be used to determine the future. A time series generated by a *deterministic linear process* has high predictability, and its future values can be forecasted very well; if it is generated by *an uncorrelated process* it has a low predictability.

In real world, time series are represented by *a mix between deterministic and stochastic* components. *Predictability* can be viewed as the *signal strength of the deterministic component* and can be estimated by using modeling methods.

*Measuring the predictability* tell whether a time series can be predicted under a particular model. Prediction of a time series with low predictability (a random walk time series) can be avoided. In this case, past observations are of little use in predicting future values, and the future values are determined randomly/ unknown factors.

*Time series analysis* builds models that describe the underlying system that generates a time series: *ARIMA*, *Box-Jenkins* time series analysis, *artificial neural networks* (ANN), *genetic programming* (GP). The focus 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 and ANN approaches.*

Emerging stock exchange applications (portfolio/ winners selection, investment timing) provides good examples for the use of this time series predictability metric. The objective is to *identify stocks that are more predictable for a given modeling method* 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.

A new time series predictability metric for use with nonlinear time series modeling techniques is

presented. The use of this new 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.

## 2. Basic unified view of soft computing concepts applications in financial series

*Kosaka (1991)* demonstrated the effectiveness of applying FL/ NNs to *buy/sell timing detection* and *portfolio selection*. *Wilson (1994)* proposed a fully automatic stock trading system based on a five step procedure. *Frick (1996)* investigated price-based heuristic trading rules by using a heuristic charting method with buy/ sell signals based on price changes and reversals. Based on a binary representation of those charts, they used GAs to generate trade strategies from the classification of different price formations. *Kassicieh (1997)* examined the performance of GAs in formulating market-timing trading rules. The goal was to develop a strategy for deciding whether to be fully invested in a stock portfolio, or a riskless investment. Inspired from *Bauer (1994)*, their inputs were differenced time series of 10 economic indicators and the GA used the best three of these series to make the timing/ switching decision. *Allen, Karjalainen (1999)* used a GA *to learn technical trading rules* for an index. The rules were able to identify periods to be in the index when daily returns were positive and volatility was low and out of the index when the reverse was true, but these latter results could largely be explained by low-order serial correlation in stock index returns. *Fernandez, Rodriguez (1999)* investigated the *profitability of a simple technical trading rule based on NNs*. In the absence of trading costs, the technical trading rule is always superior to a buy-and-hold strategy for both "bear" and "stable" markets but that the reverse holds during a "bull" market. *Baba (2000)* integrated NNs and GAs in an *intelligent decision support system (IDSS)* capable to optimize decisions and based on the average projected value and the then-current value.

*Lowe (1994)* demonstrated the efficiency of the use of NNs in *effective portfolio optimization* and *short-term prediction of multiple equities*. *Wendt (1995)* builded a *portfolio efficient frontier by using GA technique*. *Guo, Huang (1996)* proposed a method for optimizing asset allocation by using *Zimmermann's fuzzy programming method*. This algorithm permitted maximal flexibility for decision makers to effectively balance the portfolio's return and risk. *Jackson (1997)* applied a GA to the problem of *asset allocation*, first using the traditional mean variance approach and then using a direct utility maximization method for a step utility function. He compared the performance of GAs with the classical method of optimization and demonstrated the robustness to discontinuities in the search space, and sensitive to the starting values.

*Fogel* [3] added noise to data generated by the *Lorenz system* and the *logistic system*. Using GP and Akaike's information criterion (AIC) [9], it was demonstrated 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 errors (SSE). His predictability metric was defined based on comparing the SSE between the original time series and its reshuffled version.

## 3. Time series analysis, data mining and time series predictability

Time series modeling selection may be application dependent. According to the No Free Lunch (NFL) theorems [6], there is *no search algorithm that can outperform all other search algorithms over all possible search problems*. *Kaboudan* reported that genetic programming (GP) showed an equivalent or better performance in predicting stock price time series than other methods. Artificial neural networks (ANN) are also recognized to be effective in the problem of financial market forecasting [7].

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. Kaboudan's $\eta$-metric measures the level of GP-predictability of a time series.

The goal is to investigate *new time series predictability metrics* with better behavior than Kaboudan's, which represents an original contribution in the field of time series analysis and data mining. This provides an explicit measure of time series predictability.

### 3.1. Autoregressive Integrated Moving Average (ARIMA)

ARIMA model of order *(p, P, q, Q),* expressed by $\phi_p(B)\phi_P(B^L)z_t = \delta + \theta_Q(B^L)a_t$, is limited by

the requirement of *stationarity*; additionally, the residuals, the differences between the time series and the ARIMA model, are independent and normally distributed.

Here

$\phi_p(B) = \left(1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p\right)$

represents the *nonseasonal autoregressive operator of order p*;

$\phi_P\left(B^L\right) = \left(1 - \phi_{1,L} B^L - \phi_{2,L} B^{2L} - \cdots - \phi_{P,L} B^{PL}\right)$

is the *seasonal autoregressive operator of order P*;

$\theta_q(B) = \left(1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q\right)$represents the *nonseasonal moving average operator of order q*;

$\theta_Q\left(B^L\right) = \left(1 - \theta_{1,L} B^L - \theta_{2,L} B^{2L} - \ldots \theta_{Q,L} B^{Q,L}\right)$ the *seasonal moving average operator of order Q*;

$\delta = \mu\phi_p(B)\phi_p\left(B^L\right)$ is a *constant term*, where $\mu$ is the true mean of the stationary time series being modeled;

$\phi_1, \phi_2, \ldots, \phi_p; \phi_{1L}, \phi_{2L}, \ldots, \phi_{P,L}$;

$\theta_1, \theta_2, \ldots, \theta_p; \theta_{1,L}, \theta_{2,L}, \ldots, \theta_{Q,L}$

and $\delta$ are *unknown parameters* that must be estimated from sample data;

$a_t, a_{t-1}, \ldots$ 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 *backshift operator B* shifts the subscript of a time series observation backward in time. That is, $By_t = y_{t-1}$ and $B^k y_t = y_{t-k}$.

To identify the particular form of the ARIMA model, we follow the *Bowerman* algorithm:

1. Whether the constant term $\delta$ should be included in the model.
2. Which of the operators $\varphi_p$ *(B)*, $\varphi_P$ *(*$B^L$*)*, $\theta_q$ *(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 $\delta$ should be included if:

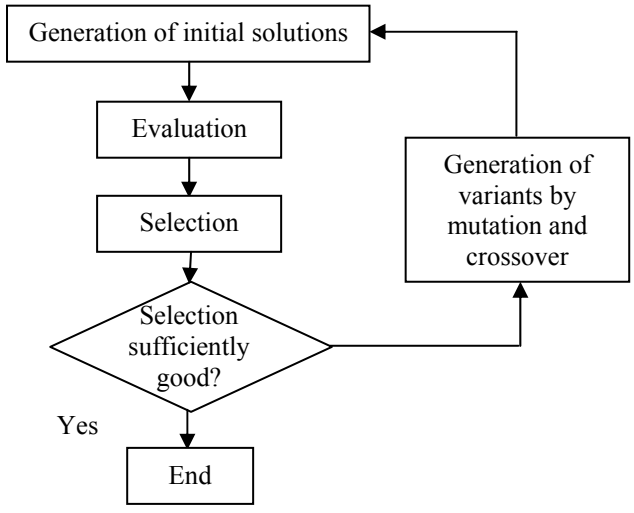
$$\left|\frac{\mu_z}{\sigma_Z / \sqrt{N_Z}}\right| > 2,$$

where $\mu_Z$ is the mean of the time series, $\sigma_Z$ 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.

### 3.2 Genetic Programming

*Genetic algorithms (GA)* adapt some concepts (reproduction, recombination, mutation, survival of the fittest, and populations) from evolutionary biology to fields of engineering, optimization, and machine learning. Such algorithms evolve populations of candidate solutions to a problem with the goal of finding near optimal candidates. Koza [8] extended this genetic approach and introduced *the concept of genetic programming* (*GP*). Each candidate solution in the search space is represented by a genetic program. *GP* maintains a population of solutions and evolves it by using transformation operators (crossover, mutation) 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* is now widely recognized as an effective search paradigm in artificial intelligence, databases, classification, robotics and many other areas. The major difference between *GP* and *GA* is that genetic program 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. 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.

The flowchart of a basic evolutionary algorithm is presented in Fig.1.




1. Generate initial population.
2. Evaluate fitness for each individual in the population.
3. Selection
4. If solution is sufficient, end the process; present the best individual in the population as the output from the algorithm.
5. Do variations by mutation, crossover/ 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 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.

*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

*Crossover:* combines the genetic material of two parents by swapping a part of one parent with a part of the other.

*Mutation:* select a point in the tree randomly and replaces the existing sub-tree 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.

*Fitness:* is the measure used by *GP* during simulated evolution of how well an individual program has learned to predict the output from the input.

### 3.3. Fast Evolutionary Programming (FEP). Reduced Parameter Bilinear (RPBL)

*Rao, Chellapilla* [10], [12] proposed an alternative modeling approach called *fast evolutionary programming* (FEP) to optimize the parameters of a *reduced parameter bilinear model* (RPBL). The *RPBL approach* [13] is capable of effectively modeling nonlinear time series with fewer parameters than a conventional bilinear model. *FEP* evolves *RPBL* models with lower normalized mean squared error (NMSE) and also lower model order than evolved with conventional evolutionary programming.

#### 3.3.1. Fast Evolutionary Programming

*FEP* is implemented by using an $(\mu+\lambda)$ evolution strategy:

1. Generate the initial population of $\mu$ randomly selected individuals, and set the generation number, *k* to one. Each individual is taken as a pair of real-valued vectors $(x_i,\eta_i), i=1,\ldots,\mu$, where $x_i$ includes the values of the solution vector elements and $\eta_i$ includes the mutation parameter values. Typically the elements f xi is selected randomly following a uniform distribution over the search space.
2. Evaluate the error score for each individual, in terms of the objective function, $f(x_i)$.
3. Mutate each parent $(x_i,\eta_i)$ to create a single offspring $(x_i',\eta_i')$ by

$$x_i'(j)=x_i(j)+\eta_i(j)C(0,1)$$
$$\eta_i'(j)=\eta_i(j)\exp\left[\tau' N(0,1)+\tau N_j(0,1)\right]$$

for $j=1,\ldots,n$, where $x_i(j)$, $x'_i(j)$, $\eta_i(j)$ and $\eta'_i(j)$ denote the $j$-th component of the vectors $x_i$, $x'_i$, $\eta_i$ and $\eta'_i$, 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 $f(x)=\dfrac{1}{s\pi\left(1+\left((x-t)/s\right)^2\right)}$ 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 $\tau$ and $\tau'$ are commonly set to be $\left(\sqrt{2\sqrt{n}}\right)^{-1}$ and $\left(\sqrt{2n}\right)^{-1}$.

4. Calculate the fitness of each offspring.
5. Conduct pair wise 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 $\mu$ 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.

#### 3.3.2. Reduced Parameter Bilinear Model

*RPBL* [13] is defined as:

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

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,

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

The 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, ARMA model is a special case of the bilinear model where $\xi_i$ 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 $x_i = [p,q,m,k, \{\varphi_j\}, \{\theta_j\}, \{\xi_j\}, \{\zeta_j\}]$. In the initial population, *p, q, m,* and *k* parameters are randomly selected and the model coefficients were selected uniformly from [-1,1].

### 3.3.3. The Identification for FEP

The identification 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: $(N-\gamma)log(\sigma_e^2)+\left(\frac{1}{2}\right)$ (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

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

*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, Chellapilla's method. 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.

## 3.4. Time Series Data Mining (TSDM)

This framework adapts *data mining concepts* to time series applications. It creates a set of methods that reveal hidden temporal patterns that are characteristic and predictive of time series events. *TSDM* is focused on *characterizing and predicting events*, and therefore overcome the limitations of requiring stationarity of the time series and normality and independence of the residuals.

### 3.4.1. Introduction in TDSM

An *event* is defined as an important occurrence in time. The associated event characterization function *g(t),* represents the value of future occurrences 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* in the series is a hidden structure that is characteristic and predictive of events.
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.

### 3.4.2 Time Series Data Mining Method

The first step in applying the TSDM method is to *find hidden temporal patterns*, characteristic of time series events. The steps in TSDM are:

*I. 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.

*II. 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.

### 3.4.3. The Optimization. An adapted GA

Adaptations include an initial random search and hashing of fitness values.

*I. Create an elite population*

1) Randomly generate a large population *(n* times normal population size)
2) Calculate fitness
3) Select the top *1/n* of t he population to continue

*II. While all fitness have not converged*

1) Selection
2) Crossover
3) Mutation

4) Reinsertion

Initializing the GA with the results of a Monte Carlo search contribute to 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%.

### 3.5. Existing Time Series Predictability metrics ( $\eta$-metric) in the literature

*Kaboudan* $\eta$ -metric 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.

## 4. A new η-metric based on GP and FEP

There are two main problems with Kaboudan's $\eta$-metric: the value of the metric largely depends on the length of the time series; for a long-term stock series, the value of the $\eta$ -metric will be distributed in a very narrow range; hence, the resolution of the metric is limited. The interest is to propose a new time series predictability metric and its method for estimating it.

### 4.1. An innovative η-metric

For a long-term time series:

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

the $\eta$ -metric is calculated on the first $Q$ points, that is, a sample series:

$\{y_t, t = 1,2,\ldots,Q\}$, ($Q<N$)

Then, the sample series is shifted by $\tau$, and the $\eta$-metric is calculated again on the new sample

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

Continuing this process, a series of $\eta$ 's is generated, which are the local predictability estimations of the subsequences of the time series. Generally, $\eta_s^Q$ is defined as the *H*-metric over the sample:

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

Thus, the $\eta$ -series is represented by

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

Since all the $\eta's$ 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 ($\eta$ depends on the length of time series) and partially solves the second problem (badly scaled and low resolution). Additionally, by examining the resulting $\eta$-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 $H$ over all windows.

To completely address the second problem, *Kaboudan*'s definition of $\eta$ is analyzed:

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

Because it uses squared error, it makes the ratio of the prediction error between the original time series $Y$ and the reshuffled version $S$ fall into a narrow range. Since the original metric compared squared error, apply the square root operator to the error is a better approach.

### 4.2. An innovative mixture between η-metric and GP/ FEP

*GP* and *FEP* approaches are considered for use as modeling methods. *GP* has *a better search ability* than *FEP*, especially when dealing with more predictable time series but *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.

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 only 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 stock prices, respectively, $vol$ is the daily traded volume of that stock, $dji$ is the value of the index.

Following this suggestion, these ten variables are used for *GP* to evolve the forecasting model using a simple 1-step predicting 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 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.

In most cases, the resulting GP equations are very complex and almost impossible to translate into humanly understandable relations between variables.

### 4.3. The use of Artificial Neural Network (ANN) in η-metric

The same inputs and output are used in the neural network model as in the *GP* model:

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

The function *NN* represents the neural network system. It takes ten past variables as the inputs and gives one single output as the prediction.

The feed-forward BP NN is created by using the MATLAB function "newff". For example, the following MABLAB code returns a *two-hidden-layer* feed-forward backpropagation NN (BkPNN). The first parameter PR is a $R \times 2$ matrix of min and max values for $R$ input elements. 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* (or output functions or activation functions) for each layer, respectively.

So, the architecture of BkPNN can be illustrated in the following schema

Sx, Sh1, Sh2, So where

Sx is the input layer,

Sh1, Sh2 are the hidden layer 1 and 2 respectively

So is the output layer.

The transfer functions are denoted by

$f1, f2, f3$ for Sh1;

$g1, g2, g3$ for Sh2;

h for So.

The transfer functions are at user's disposal. All these functions could have different forms or could be the same functions. The transfer function must be a differentiable function. Generally BkPNN uses two types of transfer functions: identical function or a sigmoid function, respectively

$$f(s) = s; \quad f(s) = \frac{1}{1+e^{-as}}, \quad a > 0, s \text{ real}$$

variable

The sigmoid function has the main property that its derivative can be expressed by using $f(s)$, i.e.

$$f(s) = af(s)[1 - f(s)].$$

This result is very important in BkP algorithms based on sigmoid functions.

Usually, in applications all the transfer functions have the same form.

Conversely: The sigmoid function $f(s)$ could be obtained by solving the Cauchy problem

$$f'(s) = a[f(s) - f^2(s)], \; f(0) = \frac{1}{2}.$$

Let us suppose that by the BkP algorithm we have to obtain the mapping $R^n \to R^M$.

A BkP algorithm uses three types of errors:

a. individual error of neuron (processing element PE) $k$, denoted $e_k(t)$, $k = 1, M; t = 1, N$;

b. So layer error at time $t$

$$E(t) = \frac{1}{2}\sum_{k=1}^{M}[e_k(t)]^2, \; t = 1, N$$

c. Total error of NN

$$E = \sum_{t=1}^{N} E(t) = \frac{1}{2}\sum_{t=1}^{N}\sum_{k=1}^{M}[e_k(t)]^2.$$

The main problem of BkP algorithm is to minimize the total error $E$. The algorithm is based on the steepest descent gradient. The gradient is denoted by [10]

$$grad_{w_j}[E(t)] = \left(\frac{\partial[E(t)]}{\partial w_{ij}(t)}\right).$$

### 5. Applications in Investment Timing in Emerging Markets

Three different modeling methods, *GP*, *ANN*, and *TSDM* are used to test the effectiveness of the new metric.

### 5.1. Financial Applications Using Predictability Metric and GP

*Kaboudan* [4] proposed the following ten variables to explain stock prices $p_t$ dynamics:

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

which can be 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* to predict the next day's price is the past 50 days. The predictability metric $\eta_n$, defined below, for a particular day $n$ is estimated using a window size $Q$ ($Q$=20) and shift step $T = 1$:

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

The new metric is defined as:

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

Thus, it is reasonable to use the predictability metric as an indicator of whether a prediction is reliable or not. The *behavior of the η-metric* is obtained by comparing the performances of *different trading strategies*: *buy and hold, trading based on the prediction of GP only, and trading based on both the GP prediction and the predictability metric (GP/η).* The third strategy (GP/η) trades on those days in which the stock has a high predictability ($\eta$>0.6), and does not trade on the other days. This idea is link to the fact 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. (*GP/η*) strategy has fewer trades than the second one and gives a much lower variance (and risk).

The idea of *the new timing strategy* is also to look at the *market portfolio*. Many stocks 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 returns per trade shown in the previous strategy still being hold. Based on this consideration, *a new improved trading strategy* is proposed.

*Choose a set of stocks to be traded on.*

1. *For each stock, calculate its H -metric.*
2. *Select N trading stocks that have the highest H.*
3. *Invest equally on the N stocks. Use GP's prediction to decide whether to go long/ short for each stock with high predictability for the current trading day.*

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. 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.

### 5.2. ANN and the Predictability Metric

For ANN the problem is simpler because even a short training set is effectively because $\eta$ -metric is not sensitive to stationary string of data and there is no need to divide past data into smaller set. Kaboudan's original $H$ -metric is good enough in this particular problem. A feed-forward BP- NN containing two hidden layers, each consisting of 3 neurons, and a output layer that has a single output neuron is trained. The trading strategies buy and hold, ANN only, and ANN/ $\eta$, are used to conduct the experiments in the following algorithm:

1. *Choose a set of stocks to be traded on.*
2. *For each stock, calculate its H -metric.*
3. *Select N stocks that have the highest H 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.*

Conclusions 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. ANN has better search ability than GP in the application of stock market predicting.

## 6. Other nonlinear time series modeling approaches

### 6.1. Fast Evolutionary Programming

*FEP* was proposed by Rao, Chellapilla 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 $\Sigma$ is the standard deviation.

**a) 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.

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.

**b) The Sunspot Time Series**

The second experiment was conducted on the yearly sunspot series. 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.

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.

Table 6.1 – *Results for the sunspot time series*

| | GP | FEP |
|---|---|---|
| Training MSE | $2.409\times10^{-2}$ | $4.019\times10^{-2}$ |
| $\sigma_{Train}$ | $6.11\times10^{-3}$ | $1.34\times10^{-3}$ |
| Test MSE | $4.582\times10^{-2}$ | $5.765\times10^{-2}$ |
| $\sigma_{Test}$ | $1.582\times10^{-2}$ | $4.23\times10^{-3}$ |
| Time (sec) | 205.1 | 70.1 |
| $\sigma_{Time}$ | 28.1 | 4.4 |

**c) Stock Prices Time Series**

The results from two arbitrarily selected 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.

In conclusion, 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, 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)

TSDM adapts and innovates *data mining* concepts *to time series* analysis. The focus is on characterizing and predicting events. It does not require the time series to be stationary and it overcomes the limitations of traditional methods of requiring normality and independence of the residuals in the time series.

In literature is shown that *TSDM is effective in recognizing patterns contained in stock price time series*; the patterns found in the training time series also exist in the test time series, but TSDM could also find *patterns in a pure noise time series in the training stage*. 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. The *η –metric is not applicable to TSDM* because it can hardly tell the differences between the original time series and the reshuffled one. Another 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]. The TSDM method takes advantage of some kind of predictability information of a time series, and thus the attempt of trying to build predictability metric on top of it could not do any better.

## 7. Conclusions

This paper 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 Kaboudan *η* – metric but overcomes the 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 GP and ANN in the application of stock time series prediction.

It was 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 feature is used 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. FEP has worse accuracy performance than GP, is not 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 $H$-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] G.E. Box, Jenkins G., Time series analysis: Forecasting and Control. Cambridge University Press, 1976

[2] W. Brock, Lakonishok J., LeBaron B. (1992) Simple Technical Trading Rules and the Stochastic Properties of Stock Returns, *Journal of Finance* XLVII, no 5, pp. 1731-1764.

[3] D. Fogel, "Preliminary experiments on discriminating between chaotic signals and noise using evolutionary programming." proceedings of Genetic Programming 1996, pp. 512-520

[4] M. Kaboudan, "Genetic Programming Prediction of Stock Prices," *Computational Economics, 2007*

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

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

[7] B. Freisleben, "Stock Market Prediction with Back propagation Networks," *Lecture on computer science,* vol. 604, 1992

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

[9] Akaike, "A new look at the statistical model identification," *IEEE Trans. Auto Control,* vol. 19, 1994.

[10]. N. Popoviciu, *Neural Networks*. *Mathematical Foundation, Algorithms and Applications*, Hyperion University Press, Bucharest (in press, 2008)

[11] G. Prelipcean, M. Boscoianu, "Some aspects regarding the dynamic correlation between different types of strategic investments in Romania after integration", Proceedings of the 9th WSEAS International Conference on Mathematics and Computers in Business and Economics (MCBE '08), Bucharest, Romania, Published by WSEAS Press, p.162-166

[12] G. Prelipcean, M. Boscoianu, "Computational Framework for Assessing Decisions in Energy Investments Based on a Mix between Real Option Analysis and Artificial Neural Networks", Proceedings of the 9th WSEAS International Conference on Mathematics and Computers in Business and Economics (MCBE '08), Bucharest, Romania, Published by WSEAS Press, p.179-184

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

[14] A.N. Refenes, Burgess A.N., Bentz Y. (1997) Neural Networks in Financial Engineering: A Study in Methodology. *IEEE Transactions on Neural Networks* 8 (6), pp. 1222-1267.

[15] M. Stoica, "Fuzzy sets and their apllications", Proceedings of the 9th WSEAS International Conference on Mathematics and Computers in Business and Economics (MCBE '08), Bucharest, Romania, Published by WSEAS Press, p.162-166

[16] X. Yao, "Fast evolutionary programming," Proceedings of 5th Annual Conference on Evolutionary Programming, Cambridge, MA, 1996, pp. 451-456

[17] Y. Zhang, "A Reduced Parameter Bilinear Time Series Model." *IEEE Trans Signal Processing,* vol. 42, 1994