Genetic Programming and Boosting Technique to Improve Time Series Forecasting

Luzia Vidal de Souza, Aurora T. R. Pozo,
Anselmo C. Neto and Joel M. C. da Rosa
Federal University of Parana
Brazil

1. Introduction

An essential element for many management decisions is an accurate forecasting. There are several methods and techniques to forecast time series that include traditional forecasting techniques with theoretical foundations in statistics. These methods present some obstacles and complexities to overcome; one of the most important ones is the difficulty to select the model that can provide the best adjustment for a specific dataset, many attempts have to be usually done until the best model can be obtained. Considering this scenario, different machine learning techniques have been recently used in this problem, such as Artificial Neural Network (ANN), Evolutionary Computation (EC), in particular, Genetic Programming (GP), which is considered a promising approach to forecast noisy complex series (Kaboudan, 2000), there are many other works founded in the literature that use (GP) to Time Series Prediction. On the other hand, recently advances in the machine learning field show that the application of the Boosting algorithm is a powerful approach to increase the accuracy of forecasting methods. Boosting algorithm was proposed and developed by Freund and Schapire (1996). According to Allwein et al. (2000), Boosting is a method of finding a highly accurate hypothesis by combining many "weak" hypotheses, each of which is only moderately accurate. Paris et al. (2004) proposed GPBoost that uses the Boosting algorithm with the GP as base learner. We have proposed a new formula for the updating of the weights and for obtain the final hypothesis of the predictor. This algorithm was called of Boosting Correlation Coefficients (BCC) and it is based on the correlation coefficient instead of the loss function used by traditional Boosting algorithms. To evaluate this approach we conducted three experiments. In the first one, the BCC was used to forecast real time series, in this experiment the mean squared error (MSE) has been used to compare the accuracy of the proposed method against the results obtained by GP, GPBoost and the traditional statistical methodology (ARMA). In the second, to prove the efficiency of the proposed methodology a widespread Monte Carlo simulation was done covering the entire ARMA spectrum, in which artificial series were generated from the parametric space of the principal ARMA models, they are AR(1), AR(2), MA(1), MA(2) e ARMA(1,1). The database generated was composed by 214.000 time series with 150 observations each one. The training set was composed by 90% of date and the others 10% composes the test set. The results were compared out of sample and the BCC showed better performance than ARMA

Source: Evolutionary Computation, Book edited by: Wellington Pinheiro dos Santos,
ISBN 978-953-307-008-7, pp. 572, October 2009, I-Tech, Vienna, Austria

methodology, Genetic Programming and GPBoost. Finally, the BCC algorithm was also applied to multiple regressions problem and the results obtained from this method were compared with the results from Artificial Neural Network, Model Tree and Boosting. This comparison showed that the BCC supplied better results than other ones. In way compare the performance of the BCC methodology with other methods, many statistical tests were performed such as Median Square Error (MSE), Root Median Square Error (RMSE) and a non parametric test Friedman. The results were compared out of sample and the BCC methodology had been presented accurate forecasts. Future research Considering that GP is able to provide solutions of high quality, and after the success of our own experiments (Souza et al., 2007a), we are encouraged to further explore GP towards finding solutions to the problem of modeling and pattern discovery of complex time series and in additional we will investigate the procedure BCC using GP as a base learner to analyze probabilistic and stochastic processes. We will investigate new tools that can work GP to more effectively solve this problem. One of the most important applications for the time series analysis is in stock markets. The goal of this task is to choose the best stocks when making an investment, and to decide which is the best strategy at the moment. Therefore, we will investigate the appropriate means for using GP in this task, as well as other general problems in financial time series. An another application that we must investigate is in Biological Networks, for example, gene regulatory network.

2. Genetic programming

Genetic Programming (GP) is an Evolutionary Computation Technique in which the individuals are computational programs. This theory was developed by John Koza (1992) and it is based on Genetic Algorithm (GA) presented by John Holland (1975). In accordance to Banzhaf (1998) and Kaboudan (2000) GP is known as an effective research paradigm in Artificial Intelligence and Machine Learning, and have been studied in the most diverse areas of knowledge, such as: digital cirucuits, data mining, molecular biology, optimization taks and another ones. In nature, those individuals that better adapt to the environment that surrounds them, have greater chance to survive. They pass their genetic characteristics to their descendents, who will suffer modifications to better adapt to the environment. After many generations, this population reaches a natural evolution. In Genetic Programming (GP), the evolutionary algorithm operates over a population of programs that have different forms and sizes. The initial population must have enough diversity, that is, the individuals must have most of the characteristics that are necessary to solve the problem, because characteristics that do not exist in the initial population will probably not appear during the evolutionary process. The evolutionary process is guided by a fitness function that measures the individual's ability to solve the problem. Those individuals that better solve the problem will receive a better fitness value and consequently, will have a better chance to be selected for the next generation. The choice of this function depends on the domain of the problem. A good choice is essential to provide good results. Once the individuals are selected, it is time to apply the genetic operators. These are: Reproduction – an individual is replicated to the next generation, with no modification in its structure; Crossover – two programs are recombined to generate two offspring and Mutation – a new sub-tree replaces a randomly selected part of a program. This process is repeated until a satisfactory solution or a stop criterion is reached. Instead of a population of beings, GP works with a population of computer programs. The goal of the GP algorithm is to select, through recombination of "genes", the program that better solves a given problem. The main elements of GP are:

- Program Structure: a tree is the most used structure to represent programs in GP. Each node can be a function or a terminal. A function has to be evaluated considering its parameters while a terminal has its own value. The terminal (T) and function (F) datasets must be provided by the user in accordance to the current problem. For example, if the datasets are: F = {+, −, *, /} and T = {x, 2} are one simple variable arithmetic expression can be generate, such as x * x + 2 or (x^2+2). Figure 1 shows the abstract syntax tree for that expression.

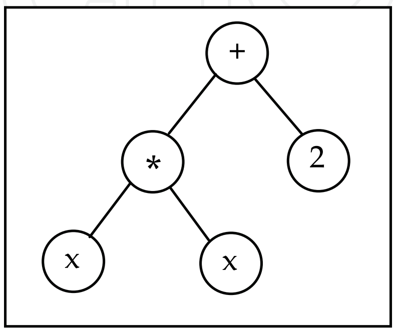

Fig. 1. Sintax tree for (x*x+2)

- Fitness Function and Selection: in nature, individuals are selected based on how well they fit to the environment. The individuals that are able to better solve the problem have better chance to be selected.
- Parameters: there are some parameters that will guide the evolutionary process, these parameters will limit and control the search performance. Some of them are: genetic operators rates (crossover rate, mutation rate), population size, selection rate (tournament size), maximum depth of the individual, etc.

In GP the population is composed by individuals that are computational programs (Koza, 1992). The first step of the algorithm is to create randomly the initial population that is the Generation 0. After that, there are two majors tasks processed in a loop with two main steps:

1. The evaluation of each program by using a fitness function: the GP algorithm receives the set that includes the values that represent the solution for the problem. For example, in a Symbolic Regression problem, it is necessary to provide the set of values of x and f (x) to the GP algorithm. These values are applied to the programs generated with the defined sets of operators and terminals. At the end, the fitness value is obtained.
2. The new population is created by selecting individuals that have better fitness value and by applying the genetic operators.

```
1.  Randomly create an initial population
2.  Repeat until a good solution or a stop criterion is reached.
    2.1  Evaluate of each program by means of the fitness function
    2.2  Select a subgroup of individuals onto which applies the genetic operators
    2.3  Apply the genetic operators
    2.4  Replace the current population by this new population
3. End
```

Fig. 2. Pseudo code of Genetic Programming

Each run of this loop represents a new generation of individuals, that are the new population that will substitute the previous one. This process is repeated until a solution is found or until a maximum number of generations is reached. At the end, the GP algorithm presents the best tree that is able to solve the given problem in the best way. The pseudo code of the GP algorithm is showed in the Figure 2.

3. Boosting algorithms

The Boosting algorithm was proposed and developed by Freund and Schapire (1996) for binary problems. According to Schapire and Singer (1997) Boosting is a method to find a highly accurate hypothesis by combining many weaker hypotheses, each of which is only moderately accurate. It manipulates the training examples to generate multiple hypotheses. In each iteration the learning algorithm uses different weights on the training examples and returns a hypothesis h_t. The weighted error of h_t is computed and applied to update the weights on the training examples. The result of the change in weights is to place higher weights in training examples that were misclassified by h_t, and lower weights on examples that were correctly classified. The final classifier is composed by the weighted vote of the individual classifiers. Freund and Schapire (1996) introduced the AdaBoost algorithm commonly referred to as the Discrete Boosting algorithm, Ridgeway (1999) developed for binary classification problems. Freund and Schapire (1997) extended AdaBoost to a multi-class case, which they called AdaBoost.M1 and AdaBoost.M2. In order to solve regression problems, Schapire (2002) extended the AdaBoost.M2 and called it AdaBoost.R. It solves regression problems by reducing them to classification ones. Drucker (1997) developed AdaBoost.R2 algorithm, which is an ad hoc modification of AdaBoost.R. He carried out some experiments with AdaBoost.R2 for regression problems and obtained some promising results (Solomatine and Shrestha, 2004). All these approaches follow the view of boosting as a gradient descent procedure (Breiman, 1997), or as residual-fitting, (Buhlmann and Yu, 2003), (Mason et al. 1999). Instead of being trained on a different sample of the same training set, as in previous boosting algorithms, a regressor is trained on a new training set that has different target values (e.g., the residual error of the sum of the previous regressor) (Assad and Bone, 2003). Bone et al. (2003) adapted a Boosting algorithm to time series forecasting using neural networks as base learners. Their experiments showed that Boosting actually provides improved results in regression problems. Iba (1999) proposed a version of AdaBoost for GP and regression. In his work the distribution was implemented picking up examples to generate a new learning set for each Boosting round. The probability of an example being picked up is proportional to its weight, and any example can be picked from 0,1 up to several times. According to Paris (2002), this approach makes the implementation easier, but the precision of weights is somewhat lost in the process.

3.1 GPBoost

Paris proposed a Boosting method that retains the precision of weights and operates on the whole set of examples for every round of Boosting. Their algorithm, named GPBoost, uses a weighted based fitness function in GP and the generic AdaBoost.R2 algorithm to update the weights its pseudo code is showed in the figure 3. The GP technique allows us to obtain accurate models from different datasets with diverse characteristics, and from the obtained model to estimate or predict the behavior of the system along time. On the other side, using Boosting, the results obtained with the merging hypothesis are always better than the results obtained with GP technique.

```
Algorithm 2 – GPBoost
1.  Given a training set S = {$(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$}; $x_i \in X$, $y_i \in R$, GP is the base algorithm
2.  $D_1(i)$ is the weight of the example $(x_i, y_i)$, $D_1(i) := 1/m \; \forall (x_i, y_i) \in S$
    Repeat: $t = 1$ to $T$
        Run GP over Dt using the function fitness:
            $fit = \sum_{i=1}^{m} \left( |f(x_i) - y_i| * D_t(i) \right) * m$,
        where $f$ is a function in GP population and $f_t$ is the best individual in the run t.
        Calculate: the loss function $L_i = \frac{|f_t(x_i) - y_i|}{\max_{i=1\ldots m} |f_t(x_i) - y_i|}$
        Calculate the average loss function: $\bar{L} = \sum_{i=1}^{m} L_i D_i$
        Let $\beta_t = \frac{\bar{L}}{1 - \bar{L}}$ is the confidence given to the function $f_t$
        Update for the next round:
         distribution $D_{t+1}(i) := \frac{D_t(i)^{1-L_i}}{Z_t}$, $Z_t$ is a normalization factor.
3.  Output: Final Hypothesis $F(x) = \min\{y \in R : \sum_{t: f_t(x) \le y} \log\left(\frac{1}{\beta_t}\right) \ge \frac{1}{2} \sum_{t=1}^{m} \log\left(\frac{1}{\beta_t}\right)\}$
```

Fig. 3. Pseudo code of GPBoost Algorithm

3.2 Boosting Correlation Coefficients (BCC)

After many studies through the Boosting algorithms, it is possible to point out that these algorithms have been sufficiently explored over classification problems. Less emphasis, however, has been given to regression problems. The Boosting algorithm for regression problems is an adaptation of the concepts used in classification. The traditional form of obtaining the weights of each example is based on error or loss function. However, the loss function is one of the possible information that can be used to obtain these weights. Recently, (Souza et al., 2007; 2009) used the BCC algorithm that is Boosting using Correlation Coefficients to time series forecasting and regressions problem. The method is a new approach of the Boosting algorithm for regression and is empirically based. BCC uses the coefficients of correlation for the updating of the weights and it has been observed that this directly influences the minimization of the loss function. The same coefficients can also be used in the final combination of predictors. The correlation coefficient is a statistical measure of the interdependence of two or more random variables. Fundamentally, the value indicates how much of a change in one variable is explained by a change in another. The BCC method is based on this measure and will be described within this section, but firstly, a brief review on the definition of Correlation Coefficients is given.

Definition: The correlation between two samples X and Y, with m observations, is calculated by using the Equation 1.

$$\rho(x,y)=\frac{\sum_{i=1}^{m}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{m}(x_i-\bar{x})^2\sum_{i=1}^{m}(y_i-\bar{y})^2}} \tag{1}$$

where $\bar{x}$ and $\bar{y}$ are the mean values of the samples X and Y respectively. The coefficient ranges from −1 to 1. If the coefficient has the value 1 it means that a linear equation describes the relationship perfectly and positively, with all data points lying on the same line and with Y increasing with X. A score of −1 shows that all data points lie on a single line but that Y increases as X decreases.
The Boosting Correlation Coefficient is a metric function that measures the relation degree between two variables, in this case between the real and the predicted values. The structure of the BCC algorithm is similar to the GPBoost algorithm. First of all, the weight distribution D_t is initialized in Step 1 and the boosting iterations start (Step 2) by calling each time the GP algorithm. After the GP's complete execution, the best individual ft in the run is chosen. After, the loss function is computed. To calculate the loss function different forms can be used, such as the exponential showed in Equation 2. Then, the correlation coefficients are calculated (Equation 3) and after the next weights are updated using the Equation 4. Finally, Step 3, the output must be computed as a combination of the different generated hypotheses. To calculate the final hypothesis F, T functions ft will be combined using again the correlation coefficients, see Equation 5.

$$L_t(x_i)=1-\exp\left(-\frac{|f_t(x_i)-y(x_i)|}{\max_{i=1\ldots m}|f_t(x_i)-y(x_i)|}\right) \tag{2}$$

$$\rho_t(f_t(x_i),y(x_i))=\frac{\sum_{i=1}^{m}\left(f_t(x_i)-f_t(\bar{x})\right)\left(y(x_i)-y(\bar{x})\right)}{\sqrt{\sum_{i=1}^{m}\left(f_t(x_i)-f_t(\bar{x})\right)^2\sum_{i=1}^{m}\left(y(x_i)-y(\bar{x})\right)^2}} \tag{3}$$

$$P_{t+1}(x_i)=\rho_t\left(f_t(x_i),y(x_i)\right)*P_t*L_t(x_i) \tag{4}$$

$$F(x_{i-1})=\frac{\sum_{t=1}^{T}\rho_t\left(f_t(x_i),y(x_i)\right)*f_t(x_i)}{\sum_{t=1}^{T}\rho_t\left(f_t(x_i),y(x_i)\right)},\ i=1,\ldots,m\text{-}1 \tag{5}$$

$$F(x_m)=\frac{\sum_{t=1}^{T}f_t(x_i)}{T},\ i=m$$

The intention in use the correlation coefficients is to promote a smoothness in the update of the weights, because it was observed that there is an inherent roughness on it, on the other hand the correlation coefficients were used to combine the obtained hypothesis in only one hypothesis that can be better than each one because of its goodness of estimation.

4. Time Series Forecasting

Time Series Forecasting is an important area of the knowledge and there are many applications in the real world. Accurate forecasting is an essential element for many management decisions. There are several methods and techniques to find a good model that can be used to produce accurate forecasting the traditional techniques have your foundations in statistics. The most important model statistical methodology is the Autoregressive and Moving Average (ARMA) models. These methods presents some obstacles and complexities to overcome. The major difficulty is to select the good model that can best adjustment for a specific dataset, usually many attempts must be performed until the best model must be found. Because of these difficulties, many researchers have been done several efforts to overcome these problems, such as Artificial Neural Network (ANN), Evolutionary Computation (EC) and in special Genetic Programming (GP) that have been provided good results in time series forecasting.

5. Experiments using real time series

This section will describe the experiments that have been accomplished using the BCC algorithm with GP as a base learner and the results are compared with other techniques such as Box & Jenkins; traditional GP and GPBoost. In the first experiment the algorithms were used in a group of academic series, in the second one (Section 6) we used a widespread Monte Carlo simulation covering the entire ARMA spectrum. First of all, we will describe the data sets that were used in the experiments, second the configuration of the methodology that were used and after the results are presented and the evaluate of the algorithms is done.

5.1 Academic and Benchmark Time Series

The academic series used in this experiment can be found at the website: (www.ime.usp.br/~pam/ST.html) and the Benchmark series at (www.economatica.com). A brief explanation about the series is presented at Table 1. Each data set was divided into two other data sets: training set that contains the 90% first values from the data set that were used to train the methods and the second one that contains the remaining values.

5.2 Box and Jenkins methodology- ARMA models

The Box & Jenkins methodology is one of the most important and recognized work in Time Series area. The research was made by George Box and Gwiliyn Jenkins (1970) and it is based on the Wold (1954) studies, who proved that any time series can be represented by a structure of infinity moving average . The methodology adjust Autoregressive and Moving Average Models ARMA(p, q) to the data set, where p is the parameter of the Autoregressive and q is the parameter of the Moving Average models and they represent the order of the model to be used. The Box and Jenkins methodology is composed by four steps.

Step 1. Identification of the model to be used on the dataset, the best model is selected by using Akaike (1974) Information Criterion (AIC), see Equation 6, where k is the number of parameters in the statistical model, and L is the maximized value of the likelihood function for the estimated model.

$$AIC = 2k - 2\ln(L) \tag{6}$$

Real Time Series	
Atmosphere:	daily measurements of the temperature in degrees centigrade, at twelve o'clock from 1/1 to 12/31/1997, São Paulo, Brazil, 365 examples.
Beverages:	monthly figures of drinks industrial production, from 1/1985 to 7/2000, Brazil, 187 examples.
Consumption:	monthly figures of product sales, from 1/1984 to 10/1984, São Paulo, Brazil, 154 examples.
Fortaleza:	annual measurements of atmospheric precipitation, from 1849 to 1997, Fortaleza, Brazil, 149 examples.
ICV:	monthly figures, pricing index, from 1/1970 to 6/1980, São Paulo, Brazil, 126 examples.
IPI:	monthly observations of Food products (Industrial Production index), from 1/1985 to 7/2000,187 examples.
Lavras:	monthly measurements of atmospheric precipitation, from 1/1966 to 12/1997, Lavras, Brazil, 384 examples.
Sunspots:	annual observations of the Wolfer Sunspots, from 1749 to 1924, 176 examples.
Djiad, Nasdaq Ibovd	daily figures of the Stock returns from their respective financial markets, from 13/08/2001 to17/08/2005, 1100 examples each dataset.

Table 1. Real Time Series

Serie	**Dataset (100%)**	**Training set (90%)**	**Test set (10%)**
Atmosphere	365	329	36
Beverage	187	169	18
Consumption	154	139	15
Fortaleza	149	135	14
ICV	126	114	12
IPI	187	169	18
Lavras	384	346	38
Sunspots	176	159	17
Djiad	1100	990	110
Ibovespa	1100	990	110
Nasdaq	1100	990	110

Table 2. Number of examples of each data set

Step 2. Estimation of the parameters to adjust the order of the model to be used.
Step 3. Adaptability verification of the model. Trough the residual analysis is verified if the model has a good fit to the dataset.
Step 4. Forecasting is made using the chosen model.

5.3 Configuration of GP, GPBoost and BCC

To apply these algorithms, we chose the tool Lil-GP 1.0 (Zongker, 1995) which is a free and easily configurable software, implemented according to Koza's GP (1992). To solve the problems it is necessary to provide the configuration files, standard GP parameters, functions and variables used to generate the models, input and output files (training and test set), as well as to specify the fitness function. The parameters used in this work to configure GP tool are showed at Table 3. These parameters have been gotten empirically after many tests have been accomplished. The terminal set used was T = { Z_{t-1}, Z_{t-2}, Z_{t-3}, Z_{t-4}} that is, the last four observations are used to make a prediction at the time t. Beside these a constant α is also used and the functions set is F = {+, −, *, /, log, cos, sin, exp, root}. This Function set allows us to obtain non linear models that have better adjust to the complex series than linear models. The fitness function used was defined as the Weighted Root Mean Square Error (WRMSE) that is, in general, used to measure the accuracy of a forecasting method. The WRMSE is showed in Equation 7, where x_i is the real value, $\hat{x}_i$ is the estimated value, D_i is the weight of each example and n is the size of the dataset. In this experiment, individuals with WRMSE equal to 0 or near to 0 are the best.

$$Fitness = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \hat{x}_i)^2 D_i m}{n}} \tag{7}$$

The Boosting algorithm with the different output hypotheses has been implemented in C computer language. For each data set, ten models of each algorithm (GP, GP-Boost and BCC) were obtained using a different random initial seed and for the GPBoost and BCC algorithm it was used ten Boosting iterations. After that, each generated model was used to forecast the values in the test set.

Parameters	**Value**
Population Size	4,000
Population initialization method	Full
Number of generations	250
Selection method	Best
Initial depth	2 - 10
Maximum depth limit	10
Maximum number of nodes	50
Crossover rate	0.7
Reproduction rate	0.2
Mutation rate	0.1

Table 3. Parameters of GP, GPBoost and BCC

5.4 Results

In order to evaluate the performance of these methods, we used the average of the Mean Square Error (MSE) obtained by using 10 initial seeds over the test set. This measure was used as a parameter of comparison because it is accepted by the statistical community. For the ARMA model, there is only one prediction and then the value of n is one. The results of this experiment are summarized in Table 4. The BCC algorithm presented the best performance, in almost all the data set the MSE average was the lowest.

Serie	ARMA	GP	GPBoost	BCC
Atmosphere	38.98	38.08	36.72	**7.59**
Beverage	308.25	245.65	231.76	**62.59**
Consumption	137.76	236.15	140.52	**60.86**
ICV	445,690.37	423,083.18	400,212.40	**209,855.37**
IPI	624.96	130.99	124.45	**17.05**
Lavras	5,335.99	13,788.93	8,249.27	**4,623.50**
Sunspots	902.70	320.85	**318.76**	329.10
Djiad	0.05	0.00	0.00	0.00
Ibovd	0.26	0.00	0.00	0.00
Nasdaq	0.06	0.00	0.00	0.00

Table 4. Average values of MSE from 10 run of the algorithms.

Beside the MSE, it was applied a non parametric test, Friedman Test (Siegal, 1988) and (Demsar, 2006), to estimate the BCC performance against the other methods. The null hypothesis that all the algorithms are equivalent was tested and rejected, after that the post-hoc multiple comparisons were performed. The results are summarized in Table 5. The algorithms ARMA, GP, GP-Boost and BCC and the result of the test shows that there is a significant difference between ARMA and BCC and between GPBoost and BCC all the other algorithms have no significant difference.

Algorithms	Statistical Difference
ARMA – GP	FALSE
ARMA – GPBoost	FALSE
ARMA – BCC	TRUE
GP – GPBoost	FALSE
GP – BCC	FALSE
GPBoost – BCC	FALSE

Table 5. Multiple comparisons among evaluated methods after Friedman test

6. Experiment using Monte Carlo simulation

A widespread Monte Carlo simulation has been done to exhaustively evaluate the performance of the BCC algorithm with GP as a base learner, in which we simulated artificial time series that belong to the entire spectrum of the structures AR(1), MA(1), AR(2), MA(2) and ARMA(1,1). To generate these series we used the free statistical software R). The parameters were put through variations within their respective parametric spaces and a noise component was added. The noise has normal distribution with mean 0 and standard deviation 1.

6.1 Parameters setting

In order to setting the parameters that were used in this simulation, it was considered the stationary region in the parametric space of the principal structure of the ARMA models: AR(1), AR(2), MA(1), MA(2), ARMA(1, 1). The stationary region of the AR(1) and MA(1) structure are represented by the Equation 8. Where ϕ_1 is the autoregressive parameter and θ_1 is the moving average parameter, both of them are defined from -1 to 1. This space was divided using step 0.1

$$\begin{gathered} -1 < \phi_1 < 1 \\ -1 < \theta_1 < 1 \end{gathered} \tag{8}$$

The parametric space of the stationary region of the AR(2) structure is defined by the Equation 9. Where ϕ_1 and ϕ_2 are the autoregressive parameters, for the MA(2) structure there is no restriction of the stationary (Morettin and Toloi, 2004), but its invertibility region is the same of the stationary region of the AR(2) structure. This region was divided using step 0.2 on the x and y axes.

$$\begin{gathered} \phi_1 + \phi_2 < 1 \\ \phi_2 - \phi_1 < 1 \\ -1 < \phi_2 < 1 \end{gathered} \tag{9}$$

Finally, the parametric space of the stationary region of the ARMA(1, 1) structure is defined by the Equation 10. Where ϕ_1 is the autoregressive parameter and θ_1 is the moving average parameter.

$$\begin{gathered} \theta_1 + \phi_1 < 1 \\ -1 < \theta_1 < 1 \\ -1 < \phi_1 < 1 \end{gathered} \tag{10}$$

Using this criterion, the data set included 214.000 series distributed for each structure as showed in the Table 6. For each set of parameter, 500 series were created, for example, the AR(1) structure has 19 parameters, then the number of series is 500 × 19 = 9,500. Each data set was divided in training and test set in the same way that was made for the real time series. The number of the examples of each data set was 150, then the training set contain 135 examples and the test set 15.

Structures	**Parameters**	**Series**
AR(1)	19	9,500
AR(2)	90	45,000
MA(1)	19	9,500
MA(2)	200	100,000
ARMA(1, 1)	100	50,000

Table 6. Monte Carlo Simulation Series

6.2 Evaluation metrics

Despite using MSE as a comparison measure, this is not enough to guarantee that the algorithm is better than another one. To analyze more precisely the performance of the proposed algorithm against other, the Friedman test was performed. The null hypothesis that states that all the algorithms are equivalent was rejected and then, the post-hoc multiple comparisons were performed. The analysis was made for each structure considering the MSE results of each set of parameters. For example, the MA(2) structure has two hundred different sets of parameters, for each of them, one MSE result is obtained for each algorithm. In this simulation, only one seed was used to GP due to the largest of the data set. The training set contains the first 90% of the values from the data set and was used to train the methods. The testing set contains the remaining values and was used to evaluate the methods.

6.3 Computational implementation

For each series from the data set (214,000) it was run 10 Boosting algorithm (GPBoost and BCC). The data sets have been run in a computational platform that includes 42 dual processor computers. The computer language used to implement the algorithms GPBoost, BCC and GP was C++ that used the Lil-GP 1.0 to run the GP as a sub routine. The data set was divided into 428 groups each one containing 500 series. Each group ran 10 Boosting algorithms including GPBoost and BCC. The computational time to run each group was 32 hours.

6.4 Results

The results were analyzed through the MSE in the test set, the MSE was calculated in accordance with Equation 11. Where serie(x_i) are the real values, ARMA(x_i) are the predicted values from ARMA models, GPBoost(x_i) are the predicted values from GPBoost and BCC(x_i) are the predicted values from BCC algorithm.

$$\mathrm{error}(h_i) = \begin{cases} \frac{1}{500}\sum_{i=1}^{500}\left(\mathrm{serie}(x_i) - \mathrm{ARMA}(x_i)\right)^2 \\ \frac{1}{500}\sum_{i=1}^{500}\left(\mathrm{serie}(x_i) - \mathrm{GP}(x_i)\right)^2 \\ \frac{1}{500}\sum_{i=1}^{500}\left(\mathrm{serie}(x_i) - \mathrm{GPBoost}(x_i)\right)^2 \\ \frac{1}{500}\sum_{i=1}^{500}\left(\mathrm{serie}(x_i) - \mathrm{BCC}(x_i)\right)^2 \end{cases} \tag{11}$$

In Table 7 is presented the MSE in the forecasted values for all the algorithms. Table 8 shows the results of the MSE for all the structure, due to space reasons only the sum of the MSE is presented. Table 9 shows the pos-hoc multiple comparisons results. In this Table, the symbol "FALSE" is used to denote that there is no statistical difference between the methods ARMA, GP, GP-Boost and BCC. From these Tables it is possible to conclude that:

- For AR(1) structure, the BCC is significantly better than the other algorithms; GP and GPBoost algorithms have no difference, as well as, ARMA and GP algorithms.
- For AR(2) structure, the BCC is significantly better than ARMA and GP algorithms, and equal to GPBoost; ARMA and GP have no difference.
- For MA(1) structure, the BCC is significantly better than the other algorithms; there is no difference between: ARMA and GP algorithms, ARMA and GP-Boost algorithms, and GP and GP-Boost algorithms.
- For MA(2) structure, the BCC is significantly better than the other algorithms; there is no difference between ARMA and GP algorithms; GP-Boost is better than GP and ARMA algorithms.
- For ARMA(1,1) structure, the BCC is significantly better than the other algorithms; there is no difference between ARMA and GP algorithms; GP-Boost is better than GP and ARMA algorithms.

Concluding, in almost all the cases the BCC method is the best and when the method is not the best, there is no statistical differences between the methods. GP-Boost is significantly better than GP and ARMA algorithms in many cases, but GP and ARMA algorithms have no difference.

Forecasting	MSE Average	AR(1)	AR(2)	MA(1)	MA(2)	ARMA(1, 1)
e136	**ARMA**	2.3702	4.3479	2.3150	3.0335	1.7917
	GP	1.0567	1.3176	5.6063	2.2456	1.8133
	GPBoost	1.0118	1.1906	1.1318	2.0985	1.1327
	BCC	**0.9282**	**1.0997**	**1.0781**	**1.9234**	**1.1087**
e137	**ARMA**	2.3527	4.1828	1.8223	2.7355	2.0809
	GP	1.9907	1.5090	1.6281	2.1690	1.2657
	GPBoost	1.1842	1.3537	1.1394	2.0773	1.1427
	BCC	**1.0924**	**1.2563**	**1.0260**	**1.8843**	**1.0553**
e138	**ARMA**	2.0450	4.0820	1.8098	2.6617	2.2494
	GP	1.0470	1.3674	1.1456	3.2616	1.3300
	GPBoost	1.0311	**1.1733**	1.0935	2.0698	1.1277
	BCC	**0.9583**	1.4734	**0.9838**	**1.9117**	**1.0897**
e139	**ARMA**	2.1817	4.1294	1.9132	2.6473	2.3892
	GP	1.3314	1.4605	1.2253	3.0242	1.6402
	GPBoost	1.2955	1.2963	1.1898	2.0711	1.1417
	BCC	**1.2194**	**1.5489**	**1.0972**	**1.9318**	**1.0871**
e140	**ARMA**	2.0486	4.2354	1.7691	2.6666	2.4768
	GP	1.5099	1.6394	1.2392	2.1816	1.3522
	GPBoost	1.3779	**1.4485**	1.1130	2.0828	1.1416
	BCC	**1.2547**	1.4844	**1.0077**	**1.8970**	**1.0806**
e141	**ARMA**	1.9550	4.5942	1.8326	2.6444	2.5734
	GP	1.4487	1.5345	1.1601	2.3654	1.2922
	GPBoost	**1.2907**	1.3558	1.1285	2.0613	1.1399
	BCC	1.4669	**1.3466**	**1.0123**	**1.9397**	**1.1006**
e142	**ARMA**	1.6495	4.5034	1.8526	2.6580	2.6567
	GP	2.2842	1.6545	1.2233	2.3011	1.6019
	GPBoost	1.1341	**1.1711**	1.1996	2.0800	**1.1467**
	BCC	**1.0966**	1.2458	**1.0695**	**1.9264**	1.2413
e143	**ARMA**	1.4880	4.7462	1.8323	2.6514	2.6738
	GP	2.0030	1.7681	1.2425	3.1361	1.2913
	GPBoost	0.9834	**1.3549**	1.1424	**2.0581**	1.1375
	BCC	**0.9053**	1.4738	**1.0188**	4.4017	**1.0651**
e144	**ARMA**	1.2274	3.6964	1.7596	2.6497	2.7013
	GP	0.8461	5.0540	1.3345	2.7877	1.3619
	GPBoost	0.8106	**1.2776**	1.1668	**2.2478**	1.1386
	BCC	**0.7475**	1.5608	**1.0914**	2.6743	**1.0664**
e145	**ARMA**	1.4540	3.6346	1.8127	2.6596	2.7278
	GP	1.1212	1.4859	1.4670	3.0190	1.3692
	GPBoost	**1.1029**	1.2636	1.1852	**2.0717**	**1.1365**
	BCC	1.3978	**1.2616**	**1.0703**	2.4415	1.4385
e146	**ARMA**	1.6769	3.3792	1.8306	2.6673	2.7513
	GP	1.4171	2.4862	1.1538	2.5006	10.2940
	GPBoost	1.3755	1.2689	1.1349	**2.0858**	1.1417
	BCC	**1.2677**	**1.2078**	**1.0377**	2.4169	**1.1677**
e147	**ARMA**	1.2946	3.4343	1.8601	2.6690	2.7759
	GP	1.0576	1.5164	1.2334	2.7557	1.8779
	GPBoost	1.0276	1.3071	1.1368	**2.0776**	**1.1450**
	BCC	**0.9326**	**1.2561**	**1.0224**	2.5989	1.1471
e148	**ARMA**	1.3387	3.3868	1.7664	2.6551	2.7658
	GP	1.1235	2.2805	1.2059	2.7182	6.5438
	GPBoost	1.0967	1.2308	1.0886	**2.0734**	1.2860
	BCC	**1.0049**	**1.1477**	**1.0428**	2.1184	**1.2057**
e149	**ARMA**	1.2239	3.5028	1.7232	2.6740	2.8048
	GP	1.0304	1.5118	1.1609	2.7705	1.3648
	GPBoost	1.0237	**1.2823**	1.1196	2.1262	1.1386
	BCC	**0.9286**	1.8203	**1.0069**	**1.9977**	**1.0578**
e150	**ARMA**	1.7268	3.4916	1.7436	2.6571	2.8316
	GP	1.7379	1.6753	1.1366	2.5503	3.4325
	GPBoost	**1.5064**	**1.4174**	1.0936	**2.0637**	1.1398
	BCC	1.5922	1.4630	**0.9879**	2.0916	**1.0977**

Table 7. MSE average in the test set

Model	ARMA	GP	GPBoost	BCC
AR(1)	29.99	25.24	20.48	15.29
AR(2)	356.08	169.57	116.35	123.88
MA(1)	35.01	29.34	21.61	19.70
MA(2)	537.74	530.49	417.93	455.41
ARMA(1,1)	255.00	252.21	114.91	113.40

Table 8. Sum of MSE average in the test set

Friedman Test Comparison						
Structure	ARMA GP	ARMA GPBoost	ARMA BCC	GP GPBoost	GP BCC	GPBoost BCC
AR(1)	FALSE	TRUE	TRUE	FALSE	TRUE	TRUE
AR(2)	FALSE	TRUE	TRUE	TRUE	TRUE	FALSE
MA(1)	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE
MA(2)	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE
ARMA(1, 1)	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE

Table 9. Multiple comparisons among evaluated methods Friedman test

7. Regression problems

In this section we describe an experiment applying the BCC algorithm for Multivariate Regressions Problems. The goal is to evaluate the proposed method in this kind of problem. In order to compare with other techniques, we used the same datasets and evaluation measures described in Souza and Vergilio (2006). In their work, they presented the results obtained with the application of Neural Networks (multi-layer perceptron, ANN), M5 Model Tree (MT), Adaboost.R2 (R2) and Adaboost.RT (RT), here replicated for comparison purposes with our approach.

7.1 Data sets

The dataset´s features are described in Table 10, these datasets are used as a benchmark for comparing results with the previous research. The dataset was divided into training (70%) and test (30%). The subsets were obtained using a random selection without replacement strategy. The procedure was repeated for ten times with different seeds in order to obtain ten statistically different subsets of data. These ten subsets were prepared to obtain consistent results. The BCC algorithm was implemented using the software R and as base learner the Cart (Classification and Regression Tree) algorithm was used. The reason to select this learning technique is that it is simple, easy and fast to train.

Data Base	**Number of Instances**	**Number of Atributtes**
CPU	209	6
Housing	506	13
Auto-Mpg	398	7
Friedman #1	1500	5

Table 10. Multiple Regression Dataset

7.2 Results

The performances of the different algorithms are presented in the Table 11. By analyzing the results we can conclude that there is no technique that will always present the best results. For some datasets the best model was obtained by using one approach, for other datasets the best model was obtained with another one. In order to analyze more precisely the algorithm's relative performance some quantitative measurement is needed rather than just subjective comparison. For this reason, following the methodology used by Souza and Vergilio (2006), we used the so-called scoring matrix. It shows the average relative performance (in %) of one technique over another technique for all the data sets considered. Element of scoring matrix SMij should be read as the ith machine's average performance (header row in Table 12) against machine j (header column in Table 12) and is calculated for a N number of datasets as is showed in the Equation 12 (for i ≠ j). From Table 12 it can be clearly observed that BCC scores highest.

$$SM_{ij} = \frac{1}{N}\sum_{k=1}^{N}\frac{RMSE_{k,j} - RMSE_{ki}}{\max\left(RMSE_{kj}, RMSE_{ki}\right)} \tag{12}$$

Bases	BCC	MT	Bagging	ANN	AdaBoost.R	AdaBoost.RT
CPU	**22.48**	34.65	32.64	**13.91**	24.45	26.52
Housing	**1.12**	3.62	3.24	3.54	3.23	3.23
Auto-Mpg	**0.85**	3.01	2.86	3.79	2.84	2.96
Friedman	**0.7**	2.19	2.06	1.51	1.82	1.72

Table 11. Performance comparison (RMSE) between the different algorithms in the test set

Machine	MT	Bagging	ANN	R2	RT	BCC	Total
MT	0.0	-6.8	-18.1	-11.5	-14.3	-61.0	-117.7
Bagging	6.8	0.0	-12.8	-9.4	-8.0	-58.2	-81.6
ANN	18.1	12.8	0.0	6.6	7.3	-40.4	4.4
R2	11.5	9.4	-6.6	0.0	1.6	-51.2	-35.3
RT	14.3	8.0	-7.3	-1.6	0.0	-52.8	-39.4
BCC	61.0	58.2	40.4	51.2	52.8	0.0	263.6

Table 12. Scoring matrix for different machines

8. Conclusion

In this paper we present the BCC algorithm that uses the correlation coefficients between the real and the forecasting value obtained using GP as a base learner. The correlation coefficient is used for update the weights and for the generation of the final formula. Differently from works found in the literature, in this paper we investigate the use of the correlation metrics as a factor, besides the error metric. This new approach, called Boosting using Correlation Coefficients (BCC), has been empirically obtained when trying to improve the results from the other methods. The correlation coefficient was considered because two algorithms could present the same mean error and different correlation coefficients for a dataset. This difference on behavior of the two algorithms can be measured by the correlation coefficient. A good correlation coefficient results in small errors for each example. The correlation coefficient is used in the proposed algorithm with two purposes:

the first use of the correlation coefficient is for the update of the weights, here the intent is to promote a smoothness of the update, because it has been observed that the original equation has an inherent roughness. The second use is to combine the final hypothesis, and in this case the intent is to allow that each hypothesis contributes to the final hypothesis according to its goodness of estimation. This idea was evaluated through two groups of experiments. In the first group of experiments, we explore the BCC for time series forecasting, using Genetic Programming (GP) as a base learner. Differently from works found in the literature a great number of series were used considering academic series and a widespread Monte Carlo simulation. In the Monte Carlo Simulation, series were generated
in the entire parametric space for the main ARMA structures: AR(1), AR(2), MA(1) MA(2) and ARMA(1,1). From all these experiments we can conclude that in almost all cases the BCC method is the best and when the method is not the best, there is no statistical difference between the compared methods. The goal of the second group of experiments was to evaluate the proposed method in multivariate regression problems. We have compared the BCC algorithm with the results reported by other authors, comparing a M5 model tree (MT), bagging, AdaBoost.R2, AdaBoost.RT and Artificial Neural Networks (ANN). For our work, a model tree was chosen as the base learner. We use a scoring matrix to analyze the algorithms' relative performance. It shows the average relative performance (in%) of one technique over another technique. Like in time series forecasting it can be clearly observed that BCC scores the highest. We can conclude that the proposed algorithm is very advantageous for regression problems. The BCC algorithm was evaluated using two different base learners and it always showed good results. These results encourage us to carry out future experiments to explore other base learners. We intend to better evaluate the proposed approach and to explore meta-learning to select the best algorithm according to the characteristics of the datasets. The meta-learning approach will help us to better understand the problems that better fit to one algorithm or another. Other future works are the investigation of other application domains like Software Reliability. Also, an interesting idea is to extend the work for classification problems, however, the correlation coefficient
can be used only for continuous variables, and then, other metrics must be considered.

9. References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Trans Autom Control,* Vol. 11, pp. 716–723.

Allwein, E. L.; Schapire, R. E. & Y. Singer. (2000). Reducing multiclass to binary: A unifying approach for margin classifiers, *Journal of Machine Learning Research*, Vol.1, pp. 113-141.

Assad, M. and Bone, R. (2003) Improving time series prediction by recurrent neural network ensembles. *Universite de Tours, Tech. Rep.*, 2003.

Banzhaf, F.; W. Nordin, Keller, P.; and Francone, F. D. (1998). *Genetic Programming: An Introduction*, Morgan Kaufmann.

Bone, R.; Assad M. and Crucianu, M. (2003). Boosting recurrent neural networks for times series prediction. *In: Proceedings of the international conference in Roanne. Springer Computer Science*, Springer, Roanne, pp 18–22

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

Breiman, L. (1997). Prediction Games and Arcing Algorithms. *Neural Computation*, Vol. 11, N· 7, pp. 1493-1518.

Buhlmann, P and Yu, B. (2003). Boosting with the loss: Regression and classification. *Journal of the American Statistical Association*, Vol. 98, no 462, pp. 324–339, June 2003.

Demsar, J. (2006). *Statistical comparisons of classifiers over multiple data sets*. J Mach Learn Res Vol. 7, pp. 1–30.

Drucker, H. (1997). Improving regression using boosting techniques. *In: Proceeding of International Conference on Machine Learning, ICML97*, Orlando, 1997.

Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. *Proceedings of the Thirteenth Conference*, pp. 148-156, Ed. Morgan Kaufmann.

Freund, Y. and Scahpire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, Vol. 55, pp. 119–139.

Kaboudan, M. A. (2000). Genetic programming prediction on stock prices, *Journal Computational Economics*, Vol. 16, pp. 207–236.

Koza, J. (1992). *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, Cambridge.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*, MIT Press.

Iba, H. (1999). Bagging, boosting, and bloating in genetic programming. *In: Proceedings of the Genetic and Evolutionary Computation conference*, pp. 1053–1060

Mason, L.; Baxter, J.; Bartlett, P. and Frean, M. (1999). *Advances in Large Margin Classifiers*. MIT Press, Functional gradient techniques for combining hypotheses, Cambridge, MA, pp. 221–247.

Moretin, P; Toloi, P. A. and Toloi, C. M. C. (2004). *Análise de séries temporais*. Ed. Edgard Blucher LTDA. São Paulo.

Paris, G.; Robiliard, D. and Fonlupt, C. (2002). Applying boosting techniques to genetic programming. *In: Selected papers from the 5th European conference on artificial evolution*. Springer, London, pp 267–280.

Ridgeway, G. (1999). The state of boosting. *Computing Science And Stastic*, vol. 31, pp. 172–181.

Schapire, R. (2002). The boosting approach to machine learning: An overview. In: *MSRI Workshop on Nonlinear Estimation and Classification*, March 2002, Berkeley, CA.

Siegal, S. and Castellan, N. (1988). *Non-parametric statistics for the behavioural sciences*. McGraw–Hill, New York

Solomatine, D. P and Shrestha, D. L. (2004). Adaboost.RT: a Boosting Algorithm for regression Problems. *IEEE*, pp. 1163-1168.

Souza, L. V.; Pozo, A. R. T; Da Rosa, J. C. M. and Chaves Neto, A. (2007). The Boosting Technique using Correlation Coefficient to Improve Time Series Forecasting Accuracy, *Proceedings of Congress on Evolutionary Computation (CEC 2007),* pp. 1288-1295. IEEE Transactions.

Souza, L. V.; Pozo, A. R. T; Da Rosa, J. C. M. and Chaves Neto, A. (2009). Applying Correlation to enhance boosting technique using genetic programming as a base learner. *Applied Intelligence*. Vol. 30, No. 7, (Feb, 2009).

Souza GA, Vergilio SR (2006). Modeling software reliability growth with artificial neural networks. *In: Proceedings of the IEEE*. Latin American test workshop, Buenos Aires, Argentina, pp 165–170

Wold, H. (1954). *A Study in the analysis of Stationary Time Series*. Almguist & Wiksell. 1st. ed., Stocolm.

Zongker, D. and Punch, B. (1995). *Lil-gp 1.0 user's manual*. Michigan State University, East Lansing.