Time Series Forecasting using Boosting Techniques With Correlation Coefficient

Luzia Vidal de Souza
Aurora T. Pozo
Anselmo Chaves Neto
Joel M. C. da Rosa

Abstract

Time series forecasting has been widely used to support decisions. In this context, a highly accurate prediction is essential to ensure the quality of the decisions. Ensembles of machines currently receive a lot of attention; they combine predictions from different forecasting techniques as a procedure to improve the accuracy. This paper explores Genetic Programming (GP) and Boosting technique to obtain an ensemble of regressors and proposes a new formula for the final hypothesis. This new formula is based on the correlation coefficient instead of the geometric median used traditionally by the boosting algorithms. To validate this method, we carried experiments using real, financial and artificial series generated by Monte Carlo Simulation. The mean squared error (MSE) has been used to compare the accuracy of the proposed method against another ones, the t-test and ANOVA test were used too. The results obtained by using this new methodology was compared with the results obtained from GP, GPBoost and the traditional statistical methodology (ARMA). The results show advantages in the use of the proposed technique;

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 usually many attempts have to be 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 that is considered a promising approach to forecast noisy complex series [1]. This paper extends previous works found in the literature and presents results of experiments that explore GP associated with Boosting algorithm. The Boosting algorithm was proposed and developed by Freund and Schapire [2]. According to Allwein, Schapire and Singer [3], 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. [4] proposed to use the Boosting algorithm with the GP as a weak learning. This paper presents this algorithm, called Boosting using Correlation Coefficient (BCC) and describes

results of different experiments. To evaluate the BCC algorithm, we conducted three groups of experiments. In the first group, we explore the BCC for some real time series forecasting, using Genetic Programming (GP) as base learner, 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). One reason to explore GP is because it has been recently explored to forecast noisy complex series [1] with promising results. The second group of time series explored in this work is the financial series, beside the forecast a trend analysis is done and finally the third group of time series used is a widespread Monte Carlo simulation covering the entire ARMA parametric space.

2 Genetic Programming

Genetic Programming is an Evolutionary Computation technique in which the individuals are computational programs. This theory was developed by John Koza[5], based on the idea of Genetic Algorithms presented by John Holland[6]. Nowadays, GP is acknowledged as an effective research paradigm in Artificial Intelligence and Machine Learning [1,6,7], and has been found in the most diverse areas of knowledge, such as: digital circuits, data mining, molecular biology, optimization tasks and many others. In nature, those individuals that better adapt to the environment that surrounds them have a 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 offsprings and Mutation - a new sub-tree replaces a randomly selected part of a program[5]. This process is repeated until a satisfactory solution or a stop criterion is reached. The GP Algorithm's pseudo-code is given bellow:

1. Randomly create an initial population
2. Repeat until a good solution or a stop criterion is reached
   (a) Evaluate of each program by means of the fitness function
   (b) Select a subgroup of individuals onto apply the genetic operators
   (c) Apply the genetic operators
   (d) Replace the current population by this new population
3. End

3 Boosting

Boosting is a way of combining many weak classifiers to produce a powerful "committee". Boosting works by sequentially applying a classification algorithm to reweighted versions of training data, and taking a weighted majority vote of the ensemble of classifiers thus produced. Each time, the weights are computed according to the error (or loss) on each example in the learning algorithm. Initially, all the weights are equals, but on each round, those weights of the misclassified examples are increased so that the weak learner is forced to focus hard on these examples in the training set. In this way, the learning algorithm is manipulated to look closer at examples with bad predictions. For many classification algorithms, this simple strategy results in dramatic improvements in their performance. Paris [4] used the GPBoost that was based in Iba's proposal [9] and realized through some experiments that it was clear that the combination of these two algorithms produced good results. The GPBoost algorithm is showed in Figure 1. 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 f_t in the run is chosen and the weight distribution of D_t is computed according to the loss function for each example. To calculate the loss function some alternatives can be used such as the exponential showed in Equation 2. This loss function is also used to calculate the confidence of f_t (Equation 3). In each iteration of the boosting algorithm, the GP is execute with a fitness function that considers the weights of each example. The fitness function has been defined as the absolute errors weighted sum (See Equation 1). However, the fitness function can be defined according to the current problem. When the Boosting algorithm is finished, Step 3, the output must be computed as a combination of the different generated hypotheses. This can be done in different forms. To evaluate the final hypothesis F, T functions f_t will be combined. The expression for the output used by Paris [4] is the geometric median weighted by confidence coefficient. These values are sorted and the geometric median (See Equation 4) is taken to be $F(x)$, the final expression.

4 Boosting using Correlation Coefficients

After having been carried through the study of the Boosting Algorithms, it is possible to remark that these algorithms have been sufficiently explored in classification problems. The traditional form of obtaining the Output Function of a Boosting algorithm is to use some kind of weighted combination of the outputs of the different boosting iterations. The weighted combination are always based on the loss function (or the confidence) of the different functions f_t such as standard median, geometric median, arithmetic median and arithmetic RMS-based median. Paris[4] reported no significant differences between the different forms of outputs. However, the loss function is one of the possible information that can be used to obtain these weights. The proposal is to use the correlation coefficient for updating the weights, it has been observed that it has direct influence on the minimization of the loss function. The same coefficient can also be used in the final combination of the predictors. The correlation coefficient is a metric function that measures the association degree between two variables. The method BCC is based on this metric and the algorithm is showed at Figure 2.

Given: $(x_1, y_1), \ldots (x_m, y_m)$,$Y \in R$
Initialize $D_1(i) = 1/m$ For $t = 1 \ldots T$:

- Train base learner using distribution D_t
- Get base hypothesis $h_t : X \rightarrow \mathsf{R}$
- Evaluate Loss Function

$$L_i(t) = 1 - exp\left(\frac{|h_t(x_i) - y_i|}{max_{i=1,\ldots,m}|h_t(x_i) - y_i|}\right) \tag{1}$$

- Compute $\beta_t = \dfrac{\bar{L}_t}{1 - \bar{L}_t}$
- Update:

$$D_{t+1}(i) = \frac{D_t(i)^{1-L_i}}{Z_t} \tag{2}$$

  Z_t is a normalization factor

Output the final classifier (geometric median of $h_t()$)

$$F(x) = min\{y \in R : \sum_{h_t(x) \leq y} log\left(\frac{1}{\beta_t}\right) \geq \frac{1}{2}\sum_{t=1}^{T} log\left(\frac{1}{\beta_t}\right)\} \tag{3}$$

Figure 1. Boosting in Regression Problems

Given: $(x_1, y_1), \ldots (x_m, y_m)$,$Y \in R$ Initialize $D_1(i) = 1/m$ For $t = 1 \ldots T$:

- Train base learner using distribution D_t
- Get base hypothesis $h_t : X \rightarrow \mathsf{R}$
- Evaluate Loss Function

$$L_i(t) = 1 - exp\left(\frac{|h_t(x_i) - y_i|}{max_{i=1,\ldots,m}|h_t(x_i) - y_i|}\right) \tag{4}$$

- Update:

$$D_{t+1}(i) = \frac{\rho(h_t(x), y)D_t(i)^{1-L_i}}{Z_t} \tag{5}$$

  $\rho_t = \rho(h_t(x), y)$ is the Pearson Correlation Coefficient between $h_t(x)$ and y evaluated in the training sample. Z_t is a normalization factor

Output the final classifier

$$F(x) = \sum_{t=1}^{T} \frac{\rho_t h_t(x)}{\sum_{t=1}^{T} \rho_t} \tag{6}$$

Figure 2. Boosting with Correlation Coefficient

5 Time Series Forecasting

In this section, we compare the GPs performance with GP using traditional Boosting and the new Boosting algorithm (BCC) using as base learner a GP algorithm. The results are also compared with the Box and Jenkins[10] traditional statistical methodology. We also present the main steps followed to configure the GP algorithm, GPBoost, BCC and Box and Jenkins methodology. Three experiments have been carried out: one exploring benchmark time series, another one is about financial time series including a trade analysis and the last one is a widespread Monte Carlo simulation covering the entire ARMA parametric space.

5.1 Forecasting using Box & Jenkins Methodology

In order to allow a comparison with traditional methods, we used the ARMA(p,q) (AutoRegressive Moving Average) models, where p and q are the autoregressive and moving average order parameters. The best model for each data set was selected using the AIC criterion, with p and q varying between 0 and 4. These models were adjusted using the free statistical software R [14], and the predictions values were evaluated for each test dataset. The ARMA model can be represented by the equation:

$$Z_t = \delta + \phi_1 Z_{t-1} + \ldots \phi_p Z_{t-p} + \theta_1 a_{t-1} + \ldots + \theta_q a_{t-q} + a_t \tag{7}$$

where δ is a constant term, ϕ_i are the autoregressive parameters and θ_i are the moving average parameters, a_t is white noise and Z_t is the time series value at time t.

5.2 Configuration of the GP, GPBoost and BCC

To apply these algorithms, we used the Lil-GP 1.0 [11] free software implemented according to Koza's GP[5]. For each problem to be solved by the tool, it is necessary to provide configuration files, standard GP parameters, functions and variables to be used for discovering the models input and output files (training set) and to specify the fitness function evaluation. The parameters used to configure the GP tool are presented in Table 1. The fitness function was defined to be the RMSE (Root Mean Square Error) (see equation 6). The RMSE is very used to measure the accuracy of forecasting methods. The terminal set used is composed by $Z_{t-1}, Z_{t-2}, Z_{t-3}, Z_{t-4}$, the four lagged variables are used to estimate Z_t. Additionaly, the value for a constant term is randomly selected.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{m}(Z_i - \hat{Z}_i)^2}{m}} * D_i(i) * m \tag{8}$$

5.3 Academic and Benchmark Time Series

The data sets used in this section are from Morettin [12], found at (http://www.ime.usp.br/pam/ST.html) and three financial time series found in (http://www.economatica.com). Each data set was divided into two other data sets: the training and testing ones. The training set contains 90% of the data series and the remaining 10% is used as test set, Table describes the data sets. Furthermore, for each dataset, ten models of each algorithm (GP, GPBoost, BCC) were obtained using a different random initial seed for each training set. After that, each generated model is used to forecast the values in the test set.

Table 1. Academic Benchmark Time Series

Real Time Series Series	Number of Examples
Atmosphere	365
Beverages	187
Consumption	154
Fortaleza	149
ICV	126
IPI	187
Lavras	384
Sunspots	176
Djiad	1100
Ibovespa	1100
Nasdaq	1100

5.4 Results

To evaluate the performance of the different methods, we used of the mean square error (MSE) average, defined in Equation 9 obtained by using the 10 initial seeds over the test set. For ARMA process, we have only one prediction and then the value of m is one. These results are summarized in Table 3.

$$MSE = \frac{\sum_{i=1}^{m}(Z_i - \hat{Z}_i)^2}{m} \tag{9}$$

5.5 Monte Carlo Simulation

In order to exhaustively evaluate this new method, a Monte Carlo Simulation have been accomplished, 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 create this series we have used the free statistical software R, that can be founded at (http://www.r-project.org/). The parameters have been varied in its respective parametric spaces and a noise component has been added. The noise has normal distribution with mean zero and standard deviation one. The dataset included 214.000 series distributed for each structure as showed in the Table 7.

Table 2. Monte Carlo Simulation

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

The same methods used in academic series were applied to these artificial time series.

5.6 Evaluation Metrics

In our approach we have considered the MSE as a comparison measure because it is an accepted metric used by the statistical community. However, it is not always an easy task to know when an algorithm presents better results from another only based on this metric. In order to analyze the relative performance of the algorithms more precisely, one statistics technique, ANOVA [13], is used to test if there is significant difference between the algorithms. Once the ANOVA test shows that there is a significant difference between two methods then Tukey-Kramer test [13] is applied to verify which the algorithms is significant better than the others.

5.7 Results

Table 8 presents the MSE in the foreseen values for all the algorithms, Table 9, shows the Anova test for AR(1) and AR(2) structures, Table 10 shows the results of the Anova test for MA(1) and MA(2) structures and finally the Table 11 shows the results of the ANOVA test for ARMA(1,1). In the Tables the symbol X is used to denote no statistical difference between the methods. For the structures AR(1) and ARMA(1,1) in 74% of the cases the method BCC is significant better than the other at 99% confidence level; on the other hand, for the structures AR(2) in 94% of the cases the method BCC is significant better than the other at 99% confidence level; following for the MA(1) structures in 87% of the cases the method BCC is significant better than the other at 99% confidence level; finally, for the structures MA(2) in 54% of the cases the method BCC is significant better than the other at 99% confidence level. Concluding, in almost all the cases the method BCC is the best, when the method is not the best, there no statistical difference between the methods.

Table 3. Genetic Programming Parameters

Parameters	Value
Population size	4000
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

We observe that in all the cases the BCC methodology was better then the another analyzed methods, because using this BCC the returns are positive bringing positive returns to the investor.

Table 4. MSE Average values test sets

Series	GP	GP-Boost	ARMA	BCC
Atmosp.	38,08	36,72	38,98	7,59
Bever.	245,65	231,76	308,25	62,59
Consum	236,15	140,52	137,76	60,86
Fortal..	423083,18	400212,40	445690,37	209855,37
ICV	554,11	573,53	661026,97	42614,53
IPI	130,99	124,45	624,96	17,05
Lavras	13788,93	8249,27	5335,99	4623,50
Suns.	320,85	318,76	902,70	329,10
Djiad	0,00	0,00	0,05	0,00
Iboves.	0,00	0,00	0,26	0,00
Nasdaq	0,00	0,00	0,06	0,00

6 Conclusion

This paper explores Boosting technique to obtain an ensemble of regressors and proposes a new formula for updating the weights and for the final hypothesis. Differently from other approaches 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 sucessfully when trying to improve the results of other methods. To evaluate the new BCC algorithm, we conducted three groups of experiments. We have also explored the BCC for time series forecasting, using Genetic Programming (GP) as base learner, in the first group of the experiments it was used academic series, to the second group of the experiments was used financial series and a trading analysis was madecomparing the methods GP, GPBoost, ARMA and the BCC method. The third group of the experiments a widespread Monte Carlo simulation was made. 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 the cases the method BCC is the best and when the method is not the best, there is no statistical difference between the methods compared. We conclude that the algorithm proposed (BCC) is very advantageous time series forecasting. These results encourage us to conduct future experiments to explore the BCC algorithm with 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 data sets.

Table 5. MSE in Foreseen values

	AR(1)	AR(2)	MA(1)	MA(2)	ARMA(1,1)
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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 6. p-values ANOVA for AR(1) and AR(2)

Forec.	p-value	Best	Forec.	p-value	Best
e136	0,0758955	BCC	e136	2,52E-12	BCC
e137	1,46E-07	X	e137	2,67E-11	BCC
e138	6,71E-10	BCC	e138	4,89E-11	BCC
e139	1,57E-08	X	e139	2,90E-09	BCC
e140	3,63E-13	X	e140	1,89E-08	BCC
e141	1,14E-14	X	e141	1,34E-09	BCC
e142	2,60E-06	X	e142	1,08E-06	BCC
e143	6,88E-15	X	e143	3,50E-05	BCC
e144	1,38E-14	X	e144	0,064436	X
e145	2,81E-08	X	e145	9,88E-15	BCC
e146	0,1778037	X	e146	1,62E-06	BCC
e147	1,25E-06	X	e147	1,04E-13	BCC
e148	0,3044825	X	e148	2,93E-05	BCC
e149	4,36E-14	X	e149	7,70E-06	BCC
e150	0,0209229	X	e150	7,08E-12	BCC

Table 7. p-values ANOVA MA(1) and MA(2)

Forec.	p-value	best	Force.	p-value	best
e137	0,098902	X	e137	6,04E-11	BCC
e138	0,0004836	X	e138	0,004464	BCC
e139	0,0002593	BCC	e139	0,070179	X
e140	0,0008891	BCC	e140	9,51E-09	BCC
e141	0,0008415	BCC	e141	2,55E-06	BCC
e142	0,0014546	BCC	e142	1,05E-06	BCC
e143	9,27E-05	BCC	e143	0,533148	X
e144	0,0090384	BCC	e144	0,669801	X
e145	0,0075967	BCC	e145	0,341698	X
e146	0,0004726	BCC	e146	0,287049	X
e147	0,0010775	BCC	e147	0,465983	X
e148	0,0012959	BCC	e148	0,022435	X
e149	0,0009356	BCC	e149	0,000156	BCC
e150	0,0016053	BCC	e150	0,037747	X

Table 8. p-values ANOVA ARMA(1,1)

Forec.	p-value	best
e136	0,0758955	X
e137	1,46E-07	BCC
e138	6,71E-10	BCC
e139	1,57E-08	BCC
e140	3,63E-13	BCC
e141	1,14E-14	BCC
e142	2,60E-06	BCC
e143	6,88E-15	BCC
e144	1,38E-14	BCC
e145	2,81E-08	BCC
e146	0,1778037	X
e147	1,25E-06	BCC
e148	0,3044825	X
e149	4,36E-14	BCC
e150	0,0209229	X

Table 9. Financial return in 110 days

	BCC	ARMA	PG	GPBoost
Djiad	2,0%	0,5%	-1,0%	-3,5%
Ibovd	16,9%	-1,7%	-8,7%	-6,9%
Nasdaq	7,3%	-5,9%	-8,8%	-3,7%

Table 10. Financial annualized returns

	BCC	ARMA	PG	GPBoost
Djiad	4,6%	1,0%	-2,3%-8,0%	
Ibovd	76,6%	-3,9%	-19,9%	-15,7%
Nasdaq	16,7%	-13,5%	-20,2%	-8,5%