Macro-Economic Time-Series Forecasting Using Linear Genetic Programming

Roberto Sánchez[1] Javier Martínez[1] Benjamín Barán[1,2]
[1] Universidad Católica “Ntra. Sra. de la Asunción”
[2] Universidad Nacional de Asunción – Paraguay {bbaran@pol.una.py}

Abstract

Recent studies in financial economics suggest that good technical analysis may have a merit in data series prediction. Linear Genetic Programming (LGP) is a genetic programming variant that evolves sequences of instructions from an imperative programming language. This paper presents a LGP approach to search times series forecasting rules. Results for three Paraguayan macro-economic time series (Consumer Price Index, Gross Internal Product & Paraguayan import from Argentina) and one artificial time series indicate that these prediction rules may be more accurate to forecast future values than some standard statistical models in use.

Keywords: Genetic Programming, Forecasting, Macro-Economic, Time Series

1. Introduction

In the last years the interest in applying artificial intelligence techniques for macro-economics analysis has been growing constantly. Some reasons are the high availability of computing resources and the quick access to large volumes of data.

The use of decision rules and analytical techniques in the financial market have become a routine and recent academic research also refers to its potential [1, 2].

The study of technical rules for time series forecasting is also of interest due to the fact that it may reveal details about time series properties of the data, not captured by standard statistical methods [4].

Genetic Programming (GP) has been noted to be an advantageous method for obtaining models for financial markets and it proved to be able to identify relevant indices [1]. Bhattacharyya *et al.* [3] consider the use of GP to induce trading decision models from high-frequency markets data. They suggest the incorporation of domain related knowledge and semantic restrictions in the genetic search process. Brock, Lakonishok and LeBaron, examined strategies for predicting financial index like Dow Jones stock Index, finding that filtering and moving average rules can generate positive returns, but these may be overshadowed by transaction costs [4].

In today economy, many sectors as government and industry use and depend on their forecasting ability for their planning process; therefore, new artificial intelligence techniques as neural networks and Genetic Programming have been studied for the last years.

Traditional forecasting methods typically are exposed to present error produced by dynamically changing conditions or human intervention. This way, it is necessary to have methods that can adjust and adapt to the changing conditions to provide better time series forecasting models.

This paper demonstrates the use of Linear Genetic Programming (LGP) to obtain time series forecasting rules [7 for a test set that includes several Paraguayan macro-economics figures as the Consumer Price Index (CPI), and the Gross Internal Product (GIP), among others. In the present work we use discrete measurements although the time is a continuous variable. Note that, two major objectives could be identified in time series analysis. The first tries to explain the variations in the past to determine a behavior pattern. The future time series behavior will be predicted using this pattern, which is the second objective of the analysis [8].

2. Genetic Programming

Evolutionary algorithm (EA) is a stochastic search process working on a population of individuals that evolves over time through the application of genetic operators and evaluated by a fitness function. The standard genetic operators used in this work are selection, crossover and mutation. Each individual represents a problem solution and fitness adaptability is a parameter that determines the quality of a solution [9].

Genetic Programming was conceived as a generalization of EA which was inspired by natural biological selection and survival of the fittest. After an initial population of potential solutions (or *programs*) is created, solutions are ranked based on their fitness. New populations are produced by selecting with larger probability good ranking solutions and performing genetic operations to produce offspring solutions that let the population evolve. This process is repeated over many generations until some stop condition is reached.

Most of the times, GP uses tree structures to represent programs (or individuals of the population). These programs are constructed from a predefined set of functions and terminals. The functions are located in the inner nodes of the tree and leaves are terminals [9]. This type of representation is known as genetic programming based on trees (*tree-based genetic programming*, TGP). A Genetic Program evolves programs for the purpose of inductive learning. Usually, it is also useful the incorporation of syntactic and semantic restrictions in the genetic process [7].

3. Linear Genetic Programming

GP definition frees the kind of representation of a program to be developed [10]. Linear Genetic Programming is a variant of the GP technique that evolves sequences of instructions from an imperative programming language, like C, C++, Java, etc. In this work, programs are represented as sequences of instructions that accept a minimum number of variables, called registers. The instruction form may be the following:

r[0] = r[1] + r[5]

where $\mathbf{r_i}$ is a register.

This type of representation includes an operator on operand registers. The result is assigned to a destination register. In the above example, **r[0]** is the destination register, **r[1]** and **r[5]** are operands while the operation is the addition (+). In general this type of coding is known as 3-register instruction. Other alternatives as 2-register instructions are also possible:

r[2]=sin[7]

Two major differences can distinguish LGP from a TGP [5]. Linear Genetic programs feature a graph-based flow that results from a multiple usage of register contents. This allows a more compact size of solutions.

Noneffective code coexists with effective code in LGP. It is also referred to as *introns*. The noneffective code manipulates registers that do not influence the output; therefore, introns may not be executed for fitness calculation. The effective length of a program is measured in the number of effective instructions it holds.

A typical representation of a program with LGP would be:

r[3]=r[8]-r[1]
r[6]=r[6]-r[17]
r[1]=r[6]+r[1]
r[2]=sin[7]
r[1]=r[2]+r[2]
r[0]=r[1]*r[1]

In this study all registers hold numerical values, i.e., real numbers. We define a set of registers to store constant values (as π) that are write-protected and can not become destination registers. These constant registers are initialized at the beginning of the program execution. LGP individuals consist of a sequence of instructions as shown above. In this work implementation, each instruction of a program includes four values (an operation, a destination and two operands). For instance, a 4-tuple as < +, 0, 2 ,9 > represents the instruction **r[0]=r[2]+r[9]**.

The maximum number of registers is defined at the beginning of the program execution. For most problems, the number of registers is restricted to 256. In general, the output register is **r[0]**. The imperative program structure also facilitates the use of multiple outputs [7]. One or more input register may be defined, depending on the specific problem formulation. In this work we considered the use of five input registers (for past data) and one output (forecast value).

In general, the operation set contains arithmetic, exponential and trigonometric functions, as well as other possible functions as hyperbolic and logic ones, but the latest will not be used for this work. Table 1 lists the general notation of all instructions and the operations used in this study.

Operation	Notation	Input Range
Arithmetic	$r_i = r_j + r_k$ $r_i = r_j - r_k$ $r_i = r_j * r_k$ $r_i = r_j / r_k$	$r_i, r_j, r_k \in \Re$
Exponential	$r_i = r_j \wedge r_k$ $r_i = \log(r_j)$	$r_i, r_j, r_k \in \Re$
Trigonometric	$r_i = \sin(r_j)$ $r_i = \cos(r_j)$	$r_i, r_j \in \Re$

Table 1: Set of Operation used in this study

4. Problem Formulation

The objective in this paper is the finding of possible non-linear economic time series models from historical data using Linear Genetic Programming. The rules or pattern may be regarded as a prediction model that approximate an objective function *f*

$$f : I^n \to O^m \tag{1}$$

where I^n denotes the input data space of dimension n and O^m is the m-dimensional output data space. In this work we assume that $n \in [1, 6]$. Since the output for this study is the next value in a time series, $m = 1$. The solutions are supposed to generalize from the training data to unknown data. For forecasting models, LGP finds a function *f* (a program in an imperative language) that using an input with the latest *n* known observations of time series, is able to forecast a future value with a minimum error, typically, in a training period using well known sliding window techniques [12]. Of course, minimizing an error in a training period does not guarantee a good forecast in a future period; therefore, a validation period is normally used to make sure that the obtained function *f* is really a good model that generalizes properties and characteristics of the studied time series.

The fitness value measures the prediction quality of an individual or program. Since our goal is to obtain predictors given input values, to evaluate the performance of such predictors we can use different notions of distance between the predicted value and the real value. There are several techniques for fitness calculations [11]. Usually fitness is a mapping error between the predicted value p_j, and the real value x_j. A popular error function for approximation problems is the mean absolute error [11] given in (2). In our context, the fitness selected is inversely proportional to the mean absolute error (MAE), measured with N consecutive samples of time series.

$$e = \frac{\sum_{j=1}^{N} |x_j - p_j|}{N} \tag{2}$$

5. Classical Methods

In order to test the experimental behavior of this proposal, we compared the performance of the implemented LGP rules to that of a 3 conventional methods [13].

1- Moving Average: is an average of the last k values used to estimate a future value p_{j+1} of a random variable x for the next time slot (t+1).

$$p_{t+1} = \frac{\sum_{i=t-k}^{t} x_i}{k} \tag{3}$$

2- Exponential Smoothing: estimates future values as a weighted sum giving larger weights to the latest observations, using a constant smoothing α that takes values in the interval [0, 1].

$$p_{t+1} = \alpha x_t + (1-\alpha) p_t \tag{4}$$

3- Exponential Smoothing with Tendency: estimates a value as a smoothing exponential method that includes a tendency factor as in:

$$p_{t+1} = \alpha x_t + (1-\alpha) p_t + T_t \tag{5}$$

where the tendency is calculate as:

$$T_t = \beta L_t + (1-\beta) T_{t-1} \tag{6}$$

where $\beta \in$ [0.1], and

$$L_{t+1} = \alpha(x_t - x_{t-1}) + (1-\alpha)(p_t - p_{t-1}) \tag{7}$$

6. Experimental Results

Four time series were chosen for these experiments, three are real Paraguayan Macro Economics data [14]:

1. Consumer Price Index (CPI);
2. Gross Internal Product;
3. Paraguay imports from Argentina;

and one is a simulated time-series to check the ability to capture periodical seasonal characteristics. The simulated time series is similar to a proposal given in Wagner *et. al* [12]. The real data time series chosen for experimentation were obtained from historical data, according to "*Banco Central del Paraguay*" [14].

Data were divided into training and validation periods. The considered time series have 80 historical values. Thus, the 60 first values correspond to a training period and the final 20 values correspond to the validation period, when the real comparison is done. For the reported experimental results, a standard laptop was used to prove that no special super-facilities are needed to implement LGP.

Parameter	**Value**
Max.instructions for program	200
Number of registers	20
Population size	1.000
Mutation rate	95%
Number of training data	60
Number of test data	20
Fitness measures	MAE
Number of tournament	4

Table 2: LGP parameters used in this work

As in other evolutionary algorithms, LGP needs a user-defined number of general system parameter shown in Table 2. Other important considerations of the implemented LGP algorithm are:

- elitism is used to avoid loosing the best found solution;
- only new individual have to be evaluated in the population;
- steady-state algorithm is implemented; therefore, four tournaments are performing for each generation;
- the tournament winners are modified using genetic operators as crossover and mutation with given probabilities;
- implemented LGP stops when a maximum number of generations is reached.

Table 3 presents experimental results for the LGP model compared to the three traditional models given above. As already mentioned in a previous section, the fitness is measured by calculating the MAE of all forecasts in the training period, not considering the validation period. This work used 1.000, 5.000 and 10.000 generations, respectively.

	CPI	GIP	Imports	Artificial Series	Average	*Ranking*
LGP 10.000	0,23	7.068,36	40,44	60,83	15,38	1
LGP 5.000	0,24	8.906,38	39,48	60,89	15,44	2
LGP 1.000	0,26	9.012,69	53,51	61,63	15,63	3
Smoothing	0,45	33.183,19	50,75	65,16	17,12	4
Smoothing with Tendency	0,32	75.602,72	110,15	69,1.43	19,17	5
Moving Average	89,97	369.633,53	54,33	1.95,22	58,05	6

Table 3: Forecasting Results 1

Table 3 reveals some interesting results. First, in all our experiments LGP models outperform traditional statistical models. It should be noted that considering Imports from Argentina, error using 5.000 generations is lower that using 10.000 generations, what may be due to overtraining. The overspecializations of solutions to the training data reduce the performance to other periods as the validation one. Results show LGP potential to produce nonlinear model for real world applications.

To illustrate the prediction capabilities of LGP, Figures 1, 2 and 3 plot CPI growth, GIP and Paraguay imports from Argentina, respectively, comparing the real values to the predicted ones.

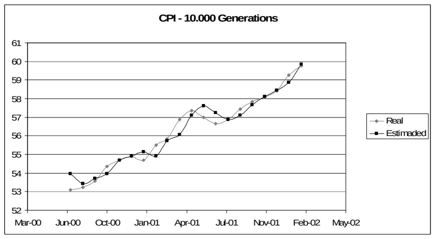

Fig. 1: LGP vs. CPI

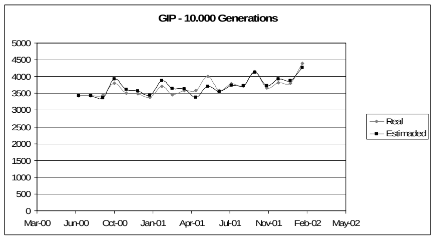

Fig. 2: LGP vs. GIP

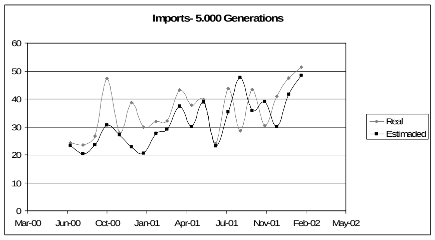

Fig. 3: LGP vs. Imports

7. Conclusion and Future Work

In this study, an LGP is developed to obtain models to estimate time series. The LGP model ability for forecasting is tested with macro-economics and artificial time series. Experimental results show that an LGP model presents better forecasting than classical statistical models, for all studied time series.

Table 3 summarizes the overall performance considering four time series, showing that LGP outperforms traditional methods. Results highlight LGP potential as a nonlinear model for real world applications. Models obtained with LGP indicate that the choice of selected parameters plays an important role in the performance of an LGP program, i.e. an overspecialization may damage solution quality.

As future work, the authors planned to continue testing, analyzing other time series. Also, they intend to compare LPG to other non-linear techniques such as Neural Networks [11]. Finally, the use of LGP in a truly multi-objective approach is also investigated.

8. References

[1] F. Allen and F. Karajalainen, "Using genetic algorithms to find technical trading rules," *J. Finan. Econom. 51*, pp. 245-271, 1999.

[2] M. Santini and A. Tettamanzi, "Genetic Programming for Financial Time Series Prediction," *In P.L. Lanzi et al. (eds.), Proceeding of EuroGP 2001. LNCS vol. 2038*, pp. 361-370. Springer, Berlin, 2001.

[3] S. Bhattacharyya, O. Pictet and G. Zumbach, "Knowledge-Intensive Genetic Discovery in Foreign Exchange Markets," *In IEEE Transactions, vol. 6, no. 2*, pp. 169-181, 2002.

[4] W. Brock, J. Lakonishok and B. LeBaron, "Simple technical trading rules and the stochastic properties of stock returns," *J. Finance*. *47*, pp. 1731-1764, 1992.

[5] G. Zumbach, O. Pictet and O. Masutti, "Genetic Programming with Syntactic Restrictions applied to Financial Volatility Forecasting," *Olsen & Associates Research Institute for Applied Economics*, 2001.

[6] J. Wooldridge, "Introductory Econometrics: A Modern Approach," *South-Western Thomson Learning*, Sydney, Australia, 1999.

[7] M. Brameier and W. Banzhaf, "Linear Genetic Programming (Genetic and Evolutionary Computation)", Springer, EEUU, 2007.

[8] F. Martinez, "Análisis de las series temporales de los precios del mercado eléctrico mediante técnicas de clustering". Universidad de Sevilla – España, http://www.lsi.us.es/docs/doctorado/memorias/Martinez,%20Francisco.pdf

[9] J. Koza, "Genetic Programming: On the Programming of Computers by Means of Natural Selection", Cambridge, MA: Mit Press, 1992.

[10] W. Banzhaf, P. Nordin, R. Keller, and F. Francone, "Genetic Programming – An Introduction. On the Automatic Evolution of Computer Programs and its Application," Morgan Kaufmann, Heidelberg/San Francisco, 1998.

[11] M. Brameier and W. Banzhaf, "A comparision of linear genetic programming and neural networks in medical data mining," *In IEEE Transactions, vol. 5*, pp.17-26, 2001.

[12] N. Wagner, Z. Michalewicz, M.Khouja and R. Mcgregor, "Time Series Forecasting for Dynamic Environments: The DyFor Genetic Program Model Evolutionary Computation," *IEEE Transactions on Evolutionary Computation, Vol. 11, No. 4.*, pp. 433-452, 2007.

[13] L. Hillier, "Investigación de Operaciones," *Séptima edición*, McGraw Hill, México, 2002.

[14] http://www.bcp.gov.py/gee, Informe de Inflación, Gerencia de Estudios Económicos del Banco Central del Paraguay.