Estimation Models Generation using Linear Genetic Programming

Javier Martínez Canillas

Catholic University "Ntra. Sra. de la Asunción"
martinez.javier@gmail.com

Roberto Sánchez

Catholic University "Ntra. Sra. de la Asunción"
totosanchez@gmail.com

Benjamín Barán

National University of Asunción - Paraguay
bbaran@una.edu.py

Abstract

The use of decision rules and estimation techniques is increasingly common for decision making. In recent years studies were conducted which applies Genetic Programming (GP) to obtain rules to make predictions. A new branch in the area of Evolutionary Algorithms (EA) is Linear Genetic Programming (LGP). LGP evolves instructions sequences of an imperative programming language. This paper proposes estimation models generation for time series forecasting using LGP. The forecasting result for the Consumer Price Index (CPI) and the price of soybeans per ton shows the potential of this new proposal.

Keywords: Genetic Programming, Economic Indicators, Time Series, Forecasting

1. INTRODUCTION

In recent years there has been an increased interest in applying artificial intelligence techniques to analyze financial markets. This is due to the increased availability of computational resources as well as quick access to a huge volume of data. In this context, the application of evolutionary computation would provide decision rules for the prediction of economic indicators. The use of decision rules and techniques for financial market analysis has become a routine and recent academic studies bear this out [1, 2].

Genetic Programming is a novel technique to obtain estimation models in financial markets [1]. For example work has been done where genetic programming is applied to the money exchange market [3]. In another study it is used to predict the Dow Jones index [4]. Predicting volatility in financial markets is another area of great interest where GP has been used with encouraging results [5].

The economy, government and financial companies use and rely on forecasts for the development of their activities. Current methods for time series prediction are error prone due the variables changing behavior and human intervention for data collection. Thus, it is necessary to have methods that can adapt to changing conditions allowing more accurate time series forecasting.

This study examines, among other series, the Paraguayan Consumer Price Index (CPI) and the soybeans price in tons. CPI, also used as an index of inflation, represents the prices of a basket of goods and its variation measures the changes in the cost variation of these products. This index tells us the purchasing power of the people [6].

This paper proposes to apply a new technique in the field of genetic programming, Linear Genetic Programming. LGP evolves imperative programming language instruction sequences instead of the traditional tree based Genetic Programming [7]. Although time is a continuous variable, this study will use discrete measurements. Those measurements correspond to approximately equidistant periods such as month or year [6]. It is important to note that two objectives could be identified in time series analysis. The first would be to explain the variations of the time series in the past, trying to determine whether they fit in a pattern. The second objective would be to predict the future behavior of the series using the pattern that models the best the time series behavior [8].

2. GENETIC PROGRAMMING

Evolutionary algorithms (EA) have in common the fact that simulates the evolution of a population of encoded solutions (individuals) manipulated by a set of operators and evaluated by a fitness function [9]. This fitness function is an adaptability parameter that determines the quality of a solution.

EA differ in how they represent solutions and which evolutionary operators they use. A relatively new area within evolutionary algorithms is Genetic Programming where software evolves trying to solve predefined problems. Thus, Genetic Programming solutions represents computer programs, unlike other EA were chromosomes represent parameters to be optimized. Genetic Programming is defined as a direct evolution of programs with the aim of inductive learning. This definition is independent of the programs representation [7].

Genetic operators must ensure the formation of syntactically correct programs. Also they should try that programs are semantically correct.

Koza used functional programming language syntactic trees to represent individuals [9]. In his representation, the functions are located in the interior nodes of the tree and the leaves represent constants or input variables. This type of representation is known as tree-based Genetic Programming (TGP) [7].

3. LINEAR GENETIC PROGRAMMING

Genetic Programming definition is independent of programs representation type. For that reason it has been developed a number of GP variants [10]. A promising novel approach is Linear Genetic Programming where programs are represented as instructions sequences of an imperative programming language. In this variant, instead of using the traditional tree-based structures, instruction sequences of an imperative programming language are evolve. Examples of this imperative programming language are C, C++ and Java. An example instruction could be:

$$r_0 = r_1 + r_5$$

where r_i represents a register. This type of representation is known as three-directions code since three directions are used to represent the statement. One direction it is used for the destination register (r_0 in the example). The other two directions are used for the operands and are known as operand registers. As discussed in [7], using register-based instructions provides the following advantages:

- Better execution time.
- Better storage space utilization (registers reutilization).
- Easier way to perform genetic operations (crossover, mutation, etc.).
- Ability to get multiple results (outputs).
- Better execution time.

In Linear Genetic Programming we can identify two spaces. The phenotype space is the set of mathematical functions and the genotype space is the set of programs in a certain representation. These programs are composed from elements of the programming language used. LGP programs are represented as instruction sequences that when executed give us the solution to our problem. For this work, the instructions can have two or three registers. Each instruction is composed of:

- A destinations register (output) that stores the result.
- One or two registers that store the values used in the operation.
- An operation from the operation set defined.

A typical representation of a LGP program using 2 and 3 registers instructions would be:

$$\begin{aligned} r[3] &= r[8] - r[1] \\ r[6] &= r[6] - r[17] \\ r[1] &= r[6] + r[1] \\ r[2] &= \sin[7] \\ r[1] &= r[2] + r[2] \\ r[0] &= r[1] * r[1] \end{aligned}$$

All registers store values, but a number of registers are reserved for numeric constants and are write-protected. These constants are initialized at program startup. Each individual consists of a sequence of instructions where each instruction is represented as a 4-tuple, $\langle op, i, j, k \rangle$. Where op represents the operation to be carried out, the letter i is the the destination register index and j, k are the operands registers indexes. For a 4-tuple defined as $\langle +, 0, 2, 9 \rangle$ the corresponding instruction would be: $r[0] = r[2] + r[9]$.

All registers including the constant ones are usually defined at the beginning of the execution. For most problems it is not necessary a very large register set. Usually the total number of registers is less than 256. By convention, if the output register is only one, it is stored at register $r[0]$. The input values would be stored starting at register $r[1]$, leaving the final registers for intermediate variables [7].

Linear Genetic Programming can generate two kinds of code. The first type is known as effective code and the program output depends on it. The code that does not affect the program output is known as non effective code or introns. These introns are usually useful in the evolution for two reasons. First it reduces the effect a variation can have in the effective code. Second the non effective code allows variations that remain neutral in terms of fitness [7]. Therefore, even if the program can no longer grow in absolute size by a priori restrictions, its effective code can still grow.

4. PROBLEM'S APPROACH

In this study, rules or patterns are obtained for time series estimation. Based on historical values we applied Linear Genetic Programming to find these estimation models. If we represent our estimation model as an objective function

$$f : I^n \rightarrow O^m \begin{cases} n \text{ denotes the input vector cardinality} \\ m \text{ denotes the output vector cardinality} \end{cases} \quad (1)$$

then individuals may be treated as predictive models that approximate the objective function f . In (1), I^n represents the input data in a n -space dimension while O^m represents the output data. In this work, n take a value in the range $[1, 6]$ and m is equal to 1. We only use one output because we are only interested in calculating a single output: the forecast for the next value in the time series studied.

The historical training data helps the evolutionary process in searching a program (individual) that represents the best estimation model. In general, given a sufficient training data it is possible to generate predictions for future unknown data at the estimation time.

For this work we use a set of 256 records, described below:

- **Calculator registers:** have data that will be used in the calculation.
- **Destination register:** stores the operation results using calculator registers.

In this paper we define a fixed number of constant registers, which can be only calculator registers. In the other hand, the registers not defined as constant registers, could be either calculator or destination register. There are also special registers known as input registers and output registers:

- **Input registers:** where input data is stored before evaluating a program ($r[10]$ to $r[15]$ in this paper).
- **Output registers:** where the results are stored after the execution of a program ($r[0]$ for this work).

LGP has two types of operations, the well known mathematical operations (arithmetic, exponential, trigonometric, boolean) and conditional ones. In the implementation done for this study we use arithmetic, exponential and trigonometric operations. Neither boolean nor conditional operations were implemented since statistical estimation models does not have those neither. Table 1 summarizes the operations used for this work.

Table 1: Linear Genetic Programming Operations.

Operation	General Notation	Input domain
Arithmetic	$r[i] = r[j] + r[k]$ $r[i] = r[j] - r[k]$ $r[i] = r[j] / r[k]$ $r[i] = r[j] * r[k]$	$r[i], r[j], r[k] \in \mathbb{R}$
Exponential	$r[i] = r[j]^{r[k]}$ $r[i] = \log_{10}(r[j])$ $r[i] = \sqrt[r[k]]{r[j]}$	$r[i], r[j], r[k] \in \mathbb{R}$
Trigonometric	$r[i] = \sin(r[j])$ $r[i] = \cos(r[j])$	$r[i], r[j], r[k] \in \mathbb{R}$

A key element of the evolutionary process is the comparison between individuals using an adaptability measure or fitness. There are many techniques for fitness calculation [11]. For this work, an individual fitness is determined by an error function that measures the mean absolute error between the estimation and the training data (2). The fitness is inversely proportional to the approximation error measured in relation to a training time series with N samples.

$$e = \frac{\sum_{t=1}^n |x_t - p_t|}{n} \quad (2)$$

where x_t is the real value and p_t the forecast for the moment t.

5. TRADITIONAL METHODS

To validate this proposal, we compared the results obtained using LGP with the predictions made using 3 widely known linear methods [11].

5.1 Moving Average

It performs an average of the last k values to estimate the value F of a random variable x for the next instant $(t + 1)$.

$$F_{t+1} = \frac{\sum_{i=t-k+1}^t x_i}{k} \quad (3)$$

5.1.1 Exponential Smoothing

It estimates the next value as a weighted sum giving more weight to recent observations using a smoothing constant α taking values in the interval $[0,1]$.

$$F_{t+1} = \alpha x_t + (1 - \alpha)F_t \quad (4)$$

where $\alpha(0 < \alpha < 1)$ is the smoothing constant. Therefore, the forecast is a weighted sum of the last point x_t and the previous forecast F_t . Expanding this recursion relation between F_{t+1} and F_t , we can express F_{t+1} as:

$$F_{t+1} = \alpha x_t + \alpha(1 - \alpha)x_{t-1} + \alpha(1 - \alpha)^2 x_{t-2} + \dots \quad (5)$$

5.1.2 Exponential Smoothing with Trend

It uses the exponential smoothing method and tailors it to include a trend factor.

$$F_{t+1} = \alpha x_t + (1 - \alpha)F_t + T_{t+1} \quad (6)$$

where the trend factor T is obtained from the equation:

$$T_{t+1} = \beta L_{t+1} + (1 - \beta)T_t \quad (7)$$

where $\beta(0 < \beta < 1)$ is the trend smoothing constant and L_{t+1} is the latest trend at time $t + 1$. The trend is obtained using the last two values and the last two forecasts. Therefore, the formula used to derive L_{t+1} , is:

$$L_{t+1} = \alpha(x_t - x_{t+1}) + (1 - \alpha)(F_t - F_{t-1}) \quad (8)$$

6. LGP COMPARISON WITH TRADITIONAL METHODS

This paper reports experimental results with a test set of four time series. Two of these are time series obtained from the Central Bank of Paraguay historical data: the price of soybeans in tons and the Consumer Price Index (CPI) [12]. Also for mathematical validation purposes, we used two artificial time series, the first is the sine function $\sin(x)$ while the second is an artificial time series proposed in [13] as follows:

$$f(x) = \begin{cases} \sin(x) + \sqrt{x} & 0 < x < 20, \\ e^x + 2 & 20 < x < 40, \\ \sin(x) + \sqrt{x} + 22 & 40 < x < 60, \end{cases} \quad (9)$$

The series produced by (9) tries to emulate a quasi-cyclic behavior [13].

7. EXPERIMENTAL RESULTS

Experimental tests were performed with various time series to validate our proposal to use LGP to approximate time series. To obtain these experimental results, 80 historical values were used. From those values 60 were used to train individuals (programs) and the remaining 20 were used to validate the models found.

LGP effectiveness, like other evolutionary algorithms is sensitive to several parameters, such as those presented in Table 2, which summarizes the values used in the this work. Every evolutionary algorithm needs a stop condition. In our implementation the stop condition is a maximum number of generations. Also, because the algorithm implemented is elitist, the best solution found is never lost. This best solution is selected as the predictive model for the series in question.

Tables 3 and 4 shows the errors (defined in equation 2) obtained by approximating the time series studied with different prediction models, using 1.000, 5.000 and 10.000 generations, respectively. As shown, the errors using LGP models are consistently lower. Interestingly, the error for the price of soybeans per tone is lower for the model obtained with 5.000 generations than the one with 10.000 generations. This probably is due an overtraining in the training period, which decreases its ability to generalize for other periods like the validation period.

Table 2: Parameters used for the experiments.

Parameter	Value
Maximum program length for initial population individuals	20
Minimum program length for initial population individuals	1
Maximum program length	200
Number of registers	20
Number of variable registers	10
Number of operators	9
Number of individuals in the population	1.000
Number of individuals per tournament	4
Number of tournaments per generation	10
Number of validation data	20
Number of training data	60
Probability of reproduction among tournament winners	100%
Probability of inserting an instruction among tournament winners	50%
Probability of mutating (effective) between tournament winners	95%

Table 3: Experimental errors when using LGP with 1.000, 5.000 and 10.000 generations.

	1.000 generations	5.000 generations	10.000 generations
Sine	0,01	0,0074	0,000097
CPI	0,32	0,29	0,28
Soybean	7,97	7,68	7,84
Artificial Series	346,14	338,32	311,09

Table 4: Experimental errors using statistical models.

	Exp. Smoothing w/ Trend	Exp. Smoothing	Moving Average
Sin	0,75	0,69	0,81
CPI	0,39	0,48	12,79
Soybean	8,48	8,72	56,32
Artificial Series	489,17	448,86	1201,56

The experiments were performed on a x86 personal computer with 2.4 GHz processor and 512 megabytes of RAM. The algorithms were implemented in C#. Table 5 shows the execution times.

Table 5: Time required to find predictive models with different numbers of PGL for generations.

	1.000 Generations	5.000 Generations	10.000 Generations
LGP	12 seg.	1 min 05 seg.	3 min. 21 seg.

Figure 1: Soybean prices and its estimation using LGP

Figure 2: Consumer Price Index (CPI) and its approximation using LGP

Figures 1 and 2 shows both test series and the series generated by the predictive models. The data is from the validation period, calculated with 10.000 generations of evolution using the training period. For space reasons there are no results for the other time series. To obtain these results we used six input registers: the last 4 values of the time series studied, the latest forecast and an index representing the prediction number.

8. CONCLUSIONS AND FUTURE WORK

In this paper LGP was used to obtain estimation models for time series prediction. We used as test cases economic indicators and artificial time series. The results obtained using LGP models shows their effectiveness to makes better predictions than traditional statistical methods.

These encouraging experimental results using Linear Genetic Programming may be due to LGP's ability to obtain both linear and nonlinear predictive models. This results in more efficient predictive models. Making a good choice of parameters for a run, one can find solutions that represent both linear functions and other models that show a nonlinear relationship between the variables of interest, achieving overall better predictions.

As future work we plan to continue the experiments analyzing other time series sets, and compared these results with other techniques like neural networks. Another promising area of research is the implementation of LGP with a multi-objective approach that considers various scenarios for training and validation.

References

- [1] F. Allen and F. Karjalainen. Using genetic algorithms to find technical trading rules. In *J. Finan. Econom.* 51, pages 245–271, 1999.
- [2] Santini M. and A. Tettamanzi. Genetic programming for financial time series prediction. In *In P.L. Lanzi et al. (eds.), Proceeding of EuroGP 2001. LNCS vol. 2038*, pages 361–370. Springer, Berlin, 2001.
- [3] S. Bhattacharyya, O. Pictet, and Zumbach G. Knowledge-intensive genetic discovery in foreign exchange markets. In *In IEEE Transactions*, vol. 6, no. 2, pages 169–181, 2002.
- [4] W. Brock, J. Lakonishok, and B. LeBaron. Simple technical trading rules and the stochastic properties of stock returns. In *J. Finance.*, pages 1731–1764, 1992.
- [5] G. Zumbach, O. Pictet, and Masutti O. *Genetic Programming with Syntactic Restrictions applied to Financial Volatility Forecasting*. Olsen and Associates Research Institute for Applied Economics, 2001.
- [6] J. Wooldridge. *Introductory Econometrics: A Modern Approach*. South-Western Thomson Learning, Sydney, Australia, 1999.
- [7] M. F. Brameier and W. Banzhaf. *Lineal Genetic Programming (Genetic and Evolutionary Computation)*. Springer, 2007.
- [8] F. Martinez. Análisis de las series temporales de los precios del mercado eléctrico mediante técnicas de clustering. <http://www.lsi.us.es/docs/doctorado/memorias/Martinez,%20Francisco.pdf>.
- [9] J. R. Koza. *Genetic Programming*. MIT Press, 1992.
- [10] N. Nedjah, A. Abraham, and L. de Macedo Mourelle. *Genetic Systems Programming: Theory and Experiences. Studies in Computational Intelligence Vol. 13*. Springer-Verlag Berlin Heidelberg, 2006.
- [11] F. S. Hillier and G. J. Lieberman. *Operational Research. Seventh Edition*. McGraw Hill, 2002.
- [12] Gerencia de Estudios Econmicos del Banco Central del Paraguay. Informe de inflación. <http://www.bcp.gov.py/gee>.
- [13] N. Wagner, Z. Michalewicz, M. Khouja, and R.R. McGregor. Time series forecasting for dynamic environments: the dyfor genetic program model. *IEEE Transactions on Evolutionary Computation*, Vol. 11, 2007.