Genetic Programming for Financial Time Series Prediction

Massimo Santini and Andrea Tettamanzi

Polo Didattico e di Ricerca di Crema
Via Bramante, 65 – 26013 Crema (CR)
{santini,tettaman}@dsi.unimi.it

Abstract This paper describes an application of genetic programming to forecasting financial markets that allowed the authors to rank first in a competition organized within the CEC2000 on "Dow Jones Prediction". The approach is substantially driven by the rules of that competition, and is characterized by individuals being made up of multiple GP expressions and specific genetic operators.

1 Introduction

Predicting time-dependent phenomena is of great importance in various fields of real world problems [5]. One of these fields is trading in financial markets.

Evolutionary algorithms in general, and GP specifically, have been applied to financial time-series prediction by various authors since their beginning. Too many works have been produced recently on this task to be all cited here; applications range from trading model [13] or technical trading rule induction [1], to option pricing [2] and modeling of the dynamics underlying financial markets [4].

Approaches to time series prediction based on GP can be roughly classified into three strands:

- approaches which use GP or another evolutionary algorithm to optimize a neural network model of the time series [15,3,16];
- GP evolving some *ad hoc* structure representing in an indirect way knowledge or informations about the time series, such as decision trees [13];
- GP evolving an expression or simple program which computes future values of the time series based on a number of past values [14,12,11,10,7].

The approach we follow in this paper falls in this last strand.

As pointed out in [7], besides conducting an efficient exploration of the search space, with a population of models that adapt to market conditions, GP discovers automatically dependencies among the factors affecting the market and thus selects the relevant variables to enter the model. This may be an advantage with respect to more traditional, and popular, autoregressive statistical approaches such as ARCH, GARCH and the like [6].

This work originated from a (successful) attempt to predict the Dow Jones stock index, one of the challenges of a contest organized in the framework of the “Congress on Evolutionary Computation” (CEC2000, La Jolla Marriott Hotel, La Jolla, California, USA, 16–19 July 2000). Some apparent peculiarities of our approach discussed later, are therefore mainly due to the rules of such competition.

The main original aspects of our work are the following:

– we evolve individuals made of several distinct expression, one for each time in the future we have to predict;
– we desing two specific genetic operators of crossover and mutation adapted to individuals of this form;
– since our objective was to predict a given realization of a time series (namely the daily closing values of the Dow Jones Industrial Average in a given period of time), we use the same data set both in the GP algorithm for training (i.e. computing the fitness) and for obtaining, from the best individual thus evolved, the prediction itself.

The work is organized as follows. In Section 2 we give the details of the challenge which influence some choices in our approach illustrated in Section 3. Finally, Section 4 reports the results of an empirical study of the behaviour of the algoritm both on a toy problem and on the target problem.

2 The problem

While most time series prediction work found in the literature is on artificial functions, the “Dow Jones Prediction Competition” mentioned in the introduction was on real-world data: the Dow Jones Index.

The call for participation asked to submit by June 17, 2000 a Dow Jones prediction for the period from June 19 to June 30. That is, each contestant was required to send a file consisting of 10 real numbers each representing the forecast of the closing value of the Dow Jones index for such period.

Submissions were scored as follows: for each day, the difference between the prediced and real index at closing time was determined. The absolute values were discounted and summed up. For the first day, the discount factor was 1.0, for the second 0.9, for the third 0.8, and so forth. More precisely, the score was computed as

$$\sum_{t=1}^{10} \frac{11-t}{10} |x_t - \hat{x}_t| \tag{1}$$

where the x_t's denote the closing values of the index and the $\hat{x}_t$'s are the predictions. The rationale for this discounting is that forecasts for the near future are commonly considered easier that for the far future.

3 The Algorithm

To fix the notation, suppose we are given a time-series $\boldsymbol{x} = (x_1, x_2, \ldots, x_T, x_{T+1}, \ldots, x_{T+h})$ for some *observation time* $T > 0$ and *horizon length* $h > 0$, together with some auxiliary time-series $\boldsymbol{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_T^{(i)})$ for $1 \leq i \leq a$ (for some $a > 0$). Our *data set* is defined as $\boldsymbol{D} = \{\boldsymbol{x}^{(i)}, 0 \leq i \leq a\}$, where $\boldsymbol{x}^{(0)}$ is defined as $(x_1, x_2, \ldots, x_T)$ and $a = 0$ means that no auxiliary time-series are considered.

Our goal is to obtain a vector $\boldsymbol{e} = (e_1, e_2, \ldots, e_h)$ of expressions $e_j : \boldsymbol{D} \times \{1, \ldots, T\} \to \mathbf{R}$, such that the value $e_j(\boldsymbol{D}, T)$ is a *prediction* of x_{T+j}, for $1 \leq j \leq h$.

3.1 The Initial Structures

The expressions e_j are built from the *terminal set* of constants in $[-c, c] \subseteq \mathbf{R}$ and the *function set* $F = \{+, -, \times, /, \mathrm{POW}, \mathrm{SQRT}, \mathrm{LOG}, \mathrm{EXP}\} \cup \bigcup_{0 \leq i \leq a} \{\mathrm{DATA}[i], \mathrm{DIFF}[i], \mathrm{AVE}[i]\}$. First of all, notice that all the functions are "protected" in the sense that they return 0 if evaluated outside their domain of definition; the access to the data set is also "protected" in the sense that $x_k^{(i)}$ takes value 0 if $k < 1$ or $k > T$, for every $0 \leq i \leq a$. The functions $\{+, -, \times, /, \mathrm{POW}, \mathrm{SQRT}, \mathrm{LOG}, \mathrm{EXP}\}$ have their usual meaning, while the value of functions in $\bigcup_{0 \leq i \leq a} \{\mathrm{DATA}[i], \mathrm{DIFF}[i], \mathrm{AVE}[i]\}$, on input $\alpha \in \mathbf{R}$ and at time $0 \leq t \leq T$, is defined as follows:

$$
\begin{aligned}
\mathrm{DATA}[i](\alpha, t) &= x_{t - \lfloor |\alpha| \rfloor}^{(i)} \\
\mathrm{DIFF}[i](\alpha, t) &= x_t^{(i)} - x_{t - \lfloor |\alpha| \rfloor}^{(i)} \\
\mathrm{AVE}[i](\alpha, t) &= \frac{1}{1 + \lfloor |\alpha| \rfloor} \sum_{k = t - \lfloor |\alpha| \rfloor}^{t} x_k^{(i)}
\end{aligned}
$$

Observe that according to the above definition, the value of each expression at any time $1 \leq t \leq T$ depends only on those value of past values in $\boldsymbol{D}$: more formally, if $\boldsymbol{D}_t = \{x_k^{(i)} : x_k^{(i)} \in \boldsymbol{D}, k \leq t\}$ we have that, for every $1 \leq t \leq T$, $e(\boldsymbol{D}, t) = e(\boldsymbol{D}_t, t)$.

For a fixed horizon length $h > 0$, the *population* consists of N *individuals* each one being a vector $\boldsymbol{e} = (e_1, e_2, \ldots, e_h)$ of expressions. The expressions are represented as strings of symbols in reverse Polish notation. The initial population is built by a procedure that generates independently each expression and individual [8]. To each element of the function set and to the constants of the terminal set is assigned a probability, each expression is then built recursively by selecting each function, or constant, according to such probabilities; if the depth (nesting level) of the expression exceeds a specified value m, then only constants are selected to ensure that every expression has at most a fixed depth.

3.2 Fitness

Since our goal is to obtain predictors for the values $(x_{T+1}, x_{T+2}, \ldots, x_{T+h})$, to evaluate the performance of such predictors we can use different notions of "distance" between the predicted and actual value.

Let $\delta : \mathbf{R} \times \mathbf{R} \rightarrow \mathbf{R}^+$ be a function measuring such distance, for instance $\delta(x,y) = (x-y)^2$ or $\delta(x,y) = |x-y|/|x|$. For a fixed δ, we define the mean error for the individual $\boldsymbol{e}$ at time $1 \leq t \leq T$ as

$$\varepsilon(\boldsymbol{e}, t) = \frac{1}{h} \sum_{j=1}^{h} \delta(x_{t+j}, e_j(\boldsymbol{D}, t)),$$

hence, we define the mean error of individual $\boldsymbol{e}$ on the whole observation time as

$$f_s(\boldsymbol{e}) = \frac{1}{T-h} \sum_{t=1}^{T-h} \varepsilon(\boldsymbol{e}, t) = \frac{1}{h(T-h)} \sum_{t=1}^{T-h} \sum_{j=1}^{h} \delta(x_{t+j}, e_j(\boldsymbol{D}, t)).$$

Given that we choose positive valued δ, we have $f_s(\boldsymbol{e}) \geq 0$ for every individual $\boldsymbol{e}$, so that we can use $f_s(\boldsymbol{e})$ as the *standardized fitness*, $f_a(\boldsymbol{e}) = (1 + f_s(\boldsymbol{e}))^{-1}$ as *adjusted fitness* and $f(\boldsymbol{e}) = f_a(\boldsymbol{e}) / \sum_{\hat{\boldsymbol{e}}} f_a(\hat{\boldsymbol{e}})$ as *normalized fitness*.

3.3 Operations for Modifying Structures

Reproduction For the reproduction we adopted two different approaches: fitness proportional, and truncation selection [9] whereby, for some ratio $0 < \rho_s < 1$, the new population is obtained replicating a suitable number of times the best $\lfloor N\rho_s \rfloor$ individuals of the previous generation. In the following, by $\rho_s = 0$ we mean that we choose the fitness proportional reproduction scheme.

Crossover Following [8], we defined the crossover between two expression e, e' as the operation that selects at random two subexpressions, one in e and the other in e' and exchanges them. Then, the crossover between two individuals $\boldsymbol{e} = (e_1, e_2, \ldots, e_h)$ and $\boldsymbol{e}' = (e'_1, e'_2, \ldots, e'_h)$ is defined as the operation performing the crossover between every pair of expressions e_j, e'_j, for $1 \leq j \leq h$.

The individuals in the population are arranged in $N/2$ pairs and to each of these pairs, with a fixed probability $0 \leq p_c \leq 1$, the above crossover is applied.

Mutation We defined the mutation of an individual as a crossover between two of its expressions chosen uniformly at random, or, with probability $1/2$, of an expression chosen uniformly at random from the individual and a new expression generated at random. Mutation is applied to every individual in the population with a fixed probability $0 \leq p_m \leq 1$.

4 Experiments

In order to assess the validity of our approach we selected a very simple test problem where the data are not actually stochastic but are generated by a completely specified, deterministic, law.

In that way, once we would be able to have the algorithm learn that law, we could rule out gross conceptual and implementation errors in our approach and then attack a more complex task such as predicting a financial time series.

4.1 Test problem: parabola

As a suitable law for our test problem we chose a parabola. More precisely, we let $x_k = k^2$ for $1 \le k \le T + h$; the parameters of the algorithm are set according to Table 1.

Parameter	*Explanation*	*Value*
T	observation time	30
h	horizon length	5
N	population size	500
G	number of generations	200
m	expression max depth	5
δ	distance measure	$\lvert x-y\rvert/\lvert x\rvert$
ρ_s	selection ratio	0.1
p_c	crossover probability	0.2
p_m	mutation probability	0.01

Table 1. Parameter setting for the parabola prediction.

We ran several experiments, collecting the maxmium and average adjusted fitness for every generation. Figure 1 shows the typical behaviour of these experiments where the above line represents the maximum adjusted fitness and the other represents the average.

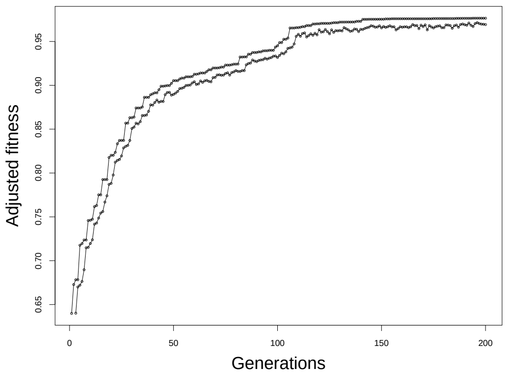

Figure 1. Graph of average and best adjusted fitness found by the algorithm after different numbers of generations, averaged over 40 runs (Parabola case).

In Figure 2 we plot the data versus the prediction: the line represent the data having coordinates (k, k^2) for $1 \le k \le T$, and the points represent the predictions, having coordinates $(k+j, e_j(\boldsymbol{D}, k))$ for $1 \le k \le T$ and $1 \le j \le h$.

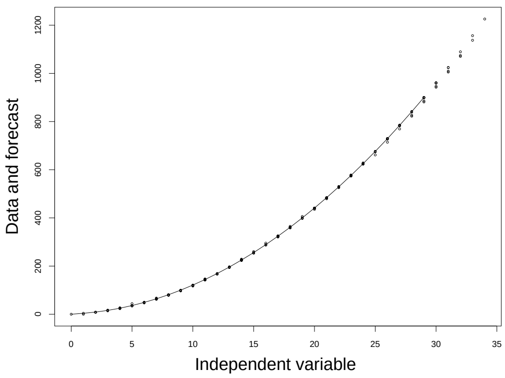

Figure 2. Graph of data versus the predictions found by the algorithm (Parabola case).

4.2 Predicting the Dow Jones

In adapting our algorithm to our main objective, we had to make several choices, which are discussed below.

Fitness Since our aim was to win the "Dow Jones Prediction Competition", our first attempt in choosing δ was to mimic the score in Equation (1). However it was clear from the very first experiments that such a choice resulted in poor predictions apart from the first or second day in the future. This can be explained by the fact that the evolutionary algorithm found it more profitable to make a slight improvement in the first days than to make more important improvements in the farther future.

Indeed, experiments convinced us to use the relative error *without discount*, $\delta = |x - y|/|x|$. While the profound reason why this choice should give better results *even for the competition*, which uses a discounted score, is not clear and calls for a deeper investigation, nonetheless empirical evidence for this fact was overwhelming.

Observation Time The challenge rules didn't place any constraint as to the observation time T so this was one of the parameters to fix. In principle, one would expect a longer T to give more accurate predictions, of course at the cost of a higher computational effort. However, a financial index like the Dow Jones shows a very marked nonstationary behaviour: exhogenous factors can cause trend inversions, rallies, dips, etc.; too long an observation time T can cause the algorithm to be misled by the long-term behaviour, while we were interested in a short term prediction. For this reason, after some trial, we chose to fix $T = 116$ (which accounts for the period from January 1, 2000 up to July 17, 2000, the deadline for the challenge).

Algorithm Parameters Some of the parameters showed no particular impact on the performance of the algorithm so, after some experiments, and following the relevant literature [8], they were set as follows: population size $N = 500$, expression (initial) max depth $m = 5$ and selection ratio $\rho_s = 1/10$.

Next we turned to study an appropriate value for the number of generations G. We ran several experiments whose results are summarized in Figure 3; it is clear from the figure that after 10 generations on average the fitness tends to stabilize so we set $G = 20$ to have a reasonable confidence to stop each run after the fitness has stabilized and no more improvements are expected.

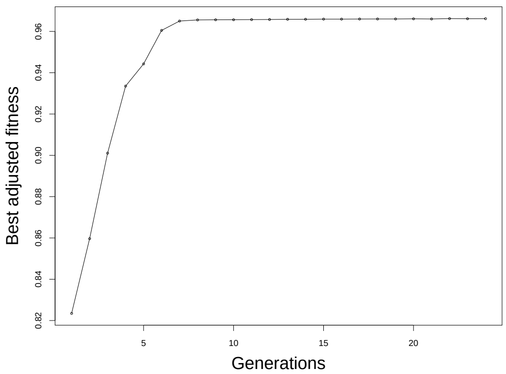

Figure 3. Graph of best adjusted fitness found by the algorithm after different numbers of generations, averaged over 40 runs (Dow Jones case).

The crossover probability p_c and the mutation probability p_m deserved a similar analysis; their effects are strongly correlated so we designed experiments to find the best combination of values for them. After plotting (see Figure 4) the best adjusted fitness after 20 generations, averaged over 20 runs, for (p_c, p_m) in the range of values usually found in the literature (that is for $p_c \in [0.4, 0.8]$ and $p_m \in [0, 0.4]$) we observe that no particular combination stood out as better than the others. So we simply set $p_c = 0.6$ and $p_m = 0.2$; we observe that this very high mutation probability is due to the fact that our mutation operator is indeed a crossover between distinct chromosomes of the same individual instead of a completely random perturbation.

The best values of the parameters determined through all the experiment discussed above are summarized in Table 2:

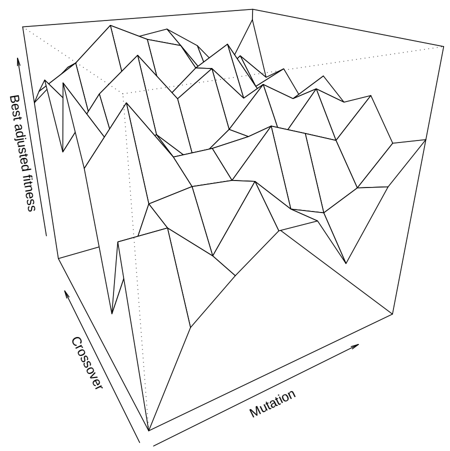

Figure 4. Graph of best adjusted fitness found by the algorithm for $p_c \in [0.4, 0.8]$ and $p_m \in [0, 0.4]$, averaged over 20 runs. Even though from the plot there appears to be a wide variation along the vertical axis, adjiusted fitness is comprised within the interval $[0.8126, 0.9671]$ (Dow Jones case).

Results Using the algorithm with parameters set as in Table 2, we took the best individual over 40 runs; Table 3 compares the actual closing values of the Dow Jones in the specified period against the predictions thus produced; in the last column is cumulated the score obtained by expresson in Equation (1).

5 Conclusion

There are already more than a dozen papers devoted to financial time series prediction using genetic programming in the literature; however, none of them approaches a problem formulated as in the Dow Jones Prediction competition. To be provocative, we could say that our work was on how to win a prediction competition rather than on how to actually predict the Dow Jones for investment or speculation purposes.

Nonetheless, we learned some lessons that might have some general value: at first, we thought that applying sophisticated statistical techniques such as ARCH, GARCH models, Markov processes and probabilistic learning, and using evolutionary algorithms to fine tune their parameters would help us obtain high-quality results. In fact, it was not so, and eventually we resolved on using a pretty simple "pure" GP approach, where all non-standard enhancements are almost dictated by the terms of the problem.

The experiments demonstrated the adopted algorithm to be quite robust with respect to parameter settings. The variability between distinct runs with the same parameter setting was some order of magnitude larger than the variability between average performances of different parameter settings. We believe this to

Parameter	*Explanation*	*Value*
T	observation time	116
h	horizon length	10
N	population size	500
G	number of generations	20
m	expression max depth	5
δ	distance measure	$\lvert x-y\rvert/\lvert x\rvert$
ρ_s	selection ratio	0.1
p_c	crossover probability	0.6
p_m	mutation probability	0.2

Table 2. Parameter setting for the Dow Jones predicion.

Prediction	*DowJones*	*Diff.*	*Disc. diff.*	*Score*
10449.300	10557.80	108.5	108.5	108.5
10449.300	10435.20	14.1	12.69	121.19
10449.300	10497.70	−48.4	38.72	159.91
10417.374	10376.10	41.274	28.8918	188.8018
10476.400	10404.80	71.6	42.96	231.7618
10535.523	10542.99	−7.467	3.7335	235.4953
10435.246	10504.46	−69.214	27.6856	263.1809
10461.648	10527.79	−66.142	19.8426	283.0235
10418.607	10398.04	20.567	4.1134	287.1369
10421.874	10447.89	−26.016	2.6016	289.7385

Table 3. Results of the best predictor evolved by the algorithm for the "Dow Jones Prediction Competition".

be a strength of the approach. Very good results were obtained by just running the algorithm as many times as it was possible after all the data needed were available and before the deadline for submission of results expired, and taking the highest fitness individual, which is a use of GP not too dissimilar in spirit from Monte Carlo methods.

References

1. F. Allen and R. Karjalainen. Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, 51(2):245–271, 1999.
2. N. K. Chidambaran, C. H. Jevons Lee, and J. R.Trigueros. An adaptive evolutionary approach to option pricing via genetic programming. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 38–41, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998. Morgan Kaufmann.

3. R. E. Dorsey and R. S. Sexton. The use of parsimonious neural networks for forecasting financial time series. *Journal of Computational Intelligence in Finance*, 6(1):24–31, 1998.
4. C. Dunis, editor. *Forecasting Financial Markets.* Wiley, 1996.
5. S. Andreas *et al.* *Time Series Prediction: Forecasting the future and understanding the past.* Addison-Wesley, 1994.
6. C. Gourieroux. *ARCH Models and Financial Applications.* Springer Verlag, 1997.
7. H. Iba and N. Nikolaev. Genetic programming polynomial models of financial data series. In *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, pages 1459–1466, La Jolla Marriott Hotel La Jolla, California, USA, 6-9 July 2000. IEEE Press.
8. J. R. Koza. *Genetic Programming.* MIT Press, Cambridge, MA, 1992.
9. H. Muehlenbein and D. Schlierkamp-Voosen. Analysis of selection, mutation and recombination in genetic algorithms. *Lecture Notes in Computer Science*, 899:142–??, 1995.
10. B. S. Mulloy, R. L. Riolo, and R. S. Savit. Dynamics of genetic programming and chaotic time series prediction. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 166–174, Stanford University, CA, USA, 28–31 July 1996. MIT Press.
11. M. Numata, K. Sugawara, S. Yamada, I. Yoshihara, and K. Abe. Time series prediction modeling by genetic programming without inheritance of model parameters. In M. Sugisaka, editor, *Proceedings 4th International Symposium on Artificial Life and Robotics*, B-Con Plaza, Beppu, Oita, Japan, 19-22 January 1999.
12. M. Numata, K. Sugawara, I. Yoshihara, and K. Abe. Time series prediction by genetic programming. In John R. Koza, editor, *Late Breaking Papers at the Genetic Programming 1998 Conference*, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998. Stanford University Bookstore.
13. M. Oussaidène, B. Chopard, O. V. Pictet, and M. Tomassini. Parallel genetic programming and its application to trading model induction. *Parallel Computing*, 23:1183–1198, 1997.
14. I. Yoshihara, T. Aoyama, and M. Yasunaga. GP-based modeling method for time series prediction with parameter optimization and node alternation. In *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, pages 1475–1481, La Jolla Marriott Hotel La Jolla, California, USA, 16–19 July 2000. IEEE Press.
15. B. Zhang, P. Ohm, and H. Mühlenbein. Evolutionary induction of sparse neural trees. *Evolutionary Computation*, 5(2):213–236, 1997.
16. B. T. Zhang. Forecasting high frequency financial time series with evolutionary neural trees: The case of hang-sheng stock market. In *Proceedings of ICAI'99*, 1999.