Comprehensibility & Overfitting Avoidance in Genetic Programming for Technical Trading Rules

Lee A. Becker & Mukund Seshadri
Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA 01609 USA
{lab | mukund}@cs.wpi.edu

Abstract

This paper presents two methods for increasing comprehensibility in technical trading rules produced by Genetic Programming. For this application domain adding a complexity penalizing factor to the objective fitness function also avoids overfitting the training data. Using pre-computed derived technical indicators, although it biases the search, can express complexity while retaining comprehensibility. Several of the learned technical trading rules outperform a buy and hold strategy for the S&P500 on the testing period from 1990-2002, even taking into account transaction costs.

1 Introduction

Genetic Programming (GP) has been applied to a wide range of problems in finance[1]. There have been a number of attempts to use GP for acquiring technical trading rules, both for Foreign Exchange Trading [2][3] and for S&P500 market timing by Allen & Karljalainen [4]. One problem encountered in the latter study was the complexity of the rules produced. The complexity of expression trees produced by GP can easily grow to 100s or even 1000s of nodes. Even if complex rules achieve excellent performance, they may turn out not be applied in practice because they are not comprehensible to human decision makers, for example, in this domain to a portfolio manager. Another problem that is encountered in many machine learning and data mining techniques, including GP is overfitting. This can occur when the learned or evolved model fits the particulars of the training data overly well and consequently does not generalize to new unseen examples. Our study demonstrates a method for GP which promotes comprehensibility and at the same time avoids overfitting.

The remainder of the paper is organized as follows. Previous work and approaches to overfitting avoidance and comprehensibility are presented. The details of the application of GP to our domain are described. The results of our experiments are then presented and discussed.

2 Previous Work and Approaches to Overfitting Avoidance and Comprehensibility

There are basically three approaches to avoiding overfitting: (1) penalizing complexity or biasing toward simplicity, (2) limiting the number of models considered, (3) using a validation data set.

Penalizing complexity or biasing toward simplicity may involve post-pruning of models or generating and hypothesizing models in the order from simple to complex or searching in the space of solutions from general to specific and using some stopping criterion. There have been a number of studies ([5], [6], [7]) where accuracy has not been reduced or has even been improved as a result of simplifying trees by pruning. There have also been theoretical arguments in favor of what has sometimes been referred to as Occam's Razor, namely that simpler models have greater predictive power and lead to less generalization error ([8], [9]).

However, Domingos[10] argues against these theoretical arguments and cites numerous recent empirical studies where simpler models have underperformed. Following Jensen & Cohen [11], Domingos regards the number of models considered rather than the complexity of the models as leading to overfitting. For GP, overfitting is often avoided by limiting the number of generations, and limiting the size of the population would also result in reducing the number of models considered.

A validation data set can be used to directly test generalization errors and thus directly decide between different models. It can be used to cutoff search thus limiting the number of models considered. Such a cutoff can also prevent complexity when the search is biased from simple to complex.

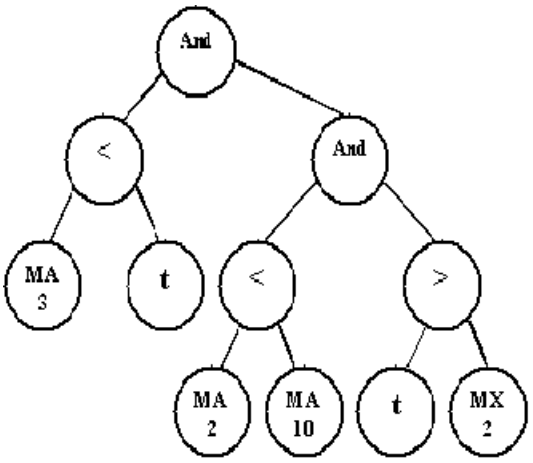
Extensive work has been done the issue of comprehensibility for decision trees. In a number of studies ([12], [13]) it is argued that deeper trees are less comprehensible especially if they are binary. To a certain extent one can equate comprehensibility with the simplicity or conciseness of the model, but consistency with the domain knowledge of the users who must comprehend the model is also an important factor [14]. If a user were to possess a chunk([15],[16]) corresponding to a significant portion of the model, the comprehensibility of the model would be better than if the model contained the same 'size' portion for which the user had no corresponding chunk. For expression trees the possibility of simplification via subsumption relationships is also exists. While doing this by hand may not be practical for large complex trees, building a domain-specific simplifier or using Mathematica or Maple to simplify equations would also be possible.

3 Genetic Programming for Comprehensible Technical Trading Rules

To achieve comprehensibility in technical trading rules we use two techniques. First, we use derived technical indicators. These are derived from the raw data indicators using various arithmetic operations, and are known to, and used by, experts in technical trading. These could conceivably be generated by GP as parts of more complicated technical trading rule expression trees. GP is particularly amenable to this use of domain knowledge. By starting with these derived indicators we bias the search, but we also will produce more comprehensible trees. Second, we use a complexity penalizing factor [17] in GP's objective fitness evaluation function. As discussed above, reducing complexity of models has sometimes led to fewer and sometimes to more generalization errors. We therefore compare the performance of the models generated by GP with vs. without the complexity penalizing factor.

For each technical trading rule, non-leaf nodes in the expression tree allowed the logical operators (AND, OR, NOT) and the arithmetic comparison operators (>, <). The leaf nodes include the following types of technical indicators.
Prices: Opening, Closing, High, Low of the current month.
Moving Averages: 2,3,6, and 10 month.
Rate of Change: 3 and 12 month.
Price Resistance Markers: two previous 3-month moving average minima and two previous 3-month moving average maxima.
Trend Line Indicators: a lower resistance line based on the slope of the 2 previous minima and a upper resistance line based on the slope of the 2 previous maxima.

Below is a sample trading rule: *if the 3 month moving average is less than the lower trend line and the 2 month moving average is less than the 10 month moving average and the lower trend line is greater than the second previous 3 month average maxima*. If you are out of the market and the rule becomes true, buy. If you are in the market and the rule becomes false, sell.



We used a standard GP algorithm[18]. The software for this work used the GAlib genetic algorithm package, written by Matthew Wall at the Massachusetts Institute of Technology[19]. We used a population of 500 trees and ran for 100 generations. It was a steady state GA with half the population being replaced each generation. Given the difference in types of the nodes (boolean-valued vs. real-valued) there were constraints on the nodes that could be exchanged during crossover.

We used S&P500 data from January 1954 through December 2002. With the exception of the Prices, all the technical indicators are derived. The latter were pre-processed, and the need to assure the two previous minima and maxima required data from 1954-1959. We trained on data from 1960-1990 and tested on data from 1991-2002. For testing, we assumed transaction costs of .5% for each buy or sell. For months when we were out of the market, we credited the interest rate on 3-month T-bills.

4 Results and Discussion

Below are tables comparing the technical trading rules generated by the GP without the complexity penalizing factor and with it. Each table includes the best technical trading rule generated by each of ten runs of 100 generations. The.last row gives the average values. It is obvious that the size in terms of the number of nodes as well as the depth of the trees is much larger when the factor is not used.

In considering the performance of the trees, one should note that with a buy and hold strategy if one invested $1000 in the S&P500 at the beginning of 1960, it would have grown to $5457 at the end of 1990. This was the in-sample period on which the GP was trained. For the out-of-sample period, $1000 would have grown to $2638. Looking at the averages, on the in-sample period the returns of the technical trading rules generated without and with the factor exceeded the buy and hold; without the factor it was more than 3 times the buy and hold return, whereas with the factor it reduced to just over 2 times the buy and hold. The large trees without the factor were able to fit the in-sample data better. When we look at the performance on the out-of-sample period we see that on average the technical trading rules learned without the factor perform worse than those with the factor. It appears that without the factor the technical trading rules were overfit to the training data. In fact, the average performance of those rules was worse than a buy and hold strategy. In contrast, the rules learned with the complexity penalizing factor were less overfit and their average performance exceeded the buy and hold strategy for the out-of-sample period.

Without Factor			
Size	**Depth**	**1960-1990**	**1991-2002**
224	19	15195	3491
542	88	13856	3300
530	45	20265	1469
41	11	16832	2023
200	38	12818	1835
967	87	14984	1834
178	28	19605	1626
45	11	16367	1977
596	68	23702	2089
111	20	17637	1764
343	*42*	*17126*	*2141*

With Factor			
Size	**Depth**	**1960-1990**	**1991-2002**
15	5	10976	3128
15	5	13690	2006
3	2	8762	3377
15	5	13981	2003
3	2	8762	3377
12	5	14788	1685
15	4	15078	2096
3	2	8762	3377
15	5	9128	3697
3	2	8762	3377
10	*4*	*11269*	*2812*

With respect to the individual rules generated, it should be noted that while two of the ten rules generated without the factor exceeded the buy and hold return in the out-of-sample period, six of the ten rules generated with the factor exceeded it.

At least for this domain, it appears that reducing complexity in the learned models does increase performance on the unseen testing sample. Given the smaller size of the trees generated with the factor, comprehensibility is also clearly improved. We conclude that for this domain when a complexity penalizing factor is combined with the use of derived technical indicators, we can produce comprehensible rules and avoid overfitting. In addition, the performance of these learned rules can exceed that of a buy and hold strategy, which in the past has not proved easy to do [4].

References

[1] Chen, S.-H. 2002. *Genetic Algorithms and Genetic Programming in Computational Finance.* Boston, MA: Kluwer.

[2] Neely, C., Weller, P., & Dittmar, R. 1997. Is Technical Analysis in the Foreign Exchange Market Profitable? A Genetic Programming Approach. *Journal of Financial and Quantitative Analysis* 32:405-26.

[3] Thomas, J. & Sycara, K. 1999. The Importance of Simplicity and Validation in Genetic Programming for Data Mining in Financial Data. *Proceedings of the joint AAAI-1999 and GECCO-1999 Workshop on Data Mining with Evolutionary Algorithms*, July, 1999.

[4] Allen, F. & Karjalainen, R. 1999. Using genetic algorithms to find technical trading rules. *Journal of Financial Economics* 51:245-271.

[5] Buntine, W. & Nibblet, T. 1992. A further comparison of splitting rules for decision-tree induction. *Machine Learning* 8:75-86.

[6] Clark, P. & Nibblet, T. 1989. The CN2 Induction Algorithm, *Machine Learning* 3:261-283.

[7] Mingers,J. 1989. An empirical comparison of pruning methods for decision tree induction. *Machine Learning* 4:227-443.

[8]A. Blumer,A., Ehrenfeucht, A., Haussler, D., & M.K.Warmuth, M.K. 1987. Occam's Razor. *Information Processing Letters* 24:377-380.

[9] Rissanen, J. 1978. Modeling by shortest data description. *Automatica* 14:465-471.

[10] Domingos, P. 1999. The Role of Occam's Razor in Knowledge Discovery. *Data Mining and Knowledge Discovery* 3:409-425, 1999.

[11] Jensen, D. & Cohen, P.R. 2000. Multiple Comparisons in Induction Algorithms. *Machine Learning,* 38:309-338.

[12] Quinlan J.R. 1987. Simplifying decision trees. *Interational Journal of Man-Machine Studies* 27:221-334.

[13] Shepherd, B. An appraisal of a decision-tree approach to image classification. In *Proceedings of the Eight International Joint Conference on Artifical Intelligence*, 496-501. Philadelphia , PA: Morgan Kaufman.

[14] Pazzani, M., Mani, S., & Shankle, W.R. 1997. Beyond concise and colorful: Learning Intelligible Rules. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, 235-238. Newport Beach, CA: AAAI Press.

[15] Miller, G.A. 1956. The magic number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review* 63:81-97.

[16] Laird, J.E., P.S. Rosenbloom, A. Newell. 1986. Chunking in Soar: The Anatomy of a General Learning Mechanism. *Machine Learning* 1:11-46.

[17] Bojarczuk, C.C., Lopes, H.S., & AA Freitas, Data Mining with Constrained-syntax Genetic Programming: Applications in Medical Data Sets. In *Proc Intelligent Data Analysis in Medicine and Pharmacology - a workshop at MedInfo-2001*, London, September 2001.

[18] Koza, J. R. 1992. *Genetic Programming: On the Programming of Computers by means of Natural Selection*. Cambridge, MA: MIT Press.

[19] http://lancet.mit.edu/ga/