The Importance of Simplicity and Validation in Genetic Programming for Data Mining in Financial Data

James D Thomas *
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
jthomas+@cs.cmu.edu

Katia Sycara
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
katia+@ri.cmu.edu

Abstract

A genetic programming system for data mining trading rules out of past foreign exchange data is described. The system is tested on real data from the dollar/yen and dollar/DM markets, and shown to produce considerable excess returns in the dollar/yen market. Design issues relating to potential rule complexity and validation regimes are explored empirically. Keeping potential rules as simple as possible is shown to be the most important component of success. Validation issues are more complicated. Inspection of fitness on a validation set is used to cut-off search in hopes of avoiding overfitting. Additional attempts to use the validation set to improve performance are shown to be ineffective in the standard framework. An examination of correlations between performance on the validation set and on the test set leads to an understanding of how such measures can be marginally benificial; unfortunately, this suggests that further attemps to improve performance through validation will be difficult.

1 Introduction

There has long been a strong interest in applying computational intelligence to financial data. Traditionally such attempts have been on prediction, on identifying near-future price movements. However, another approach develops trading rules which are unconcerned with continual price prediction and wait for certain conditions to trigger buy or sell actions. Recently, economists have had some success using genetic algorithms to generate such trading rules that produce statistically significant out-of-sample returns [7], [1].

We are interested in building a general system of data mining for automatic rule discovery that incorporates these genetic programming techniques. First, we have to examine the genetic programming methodology itself. This paper implements a genetic programming-based system for trading rule discovery and evaluates its performance over real world exchange rate data in the dollar/yen and dollar/dm markets. Given the immense amount of noise in financial markets, we are most concerned with those aspects of our system that allow us to fight overfitting, specifically rule complexity and validation methodologies.

2 Previous Work

The efficiency of financial markets has long been a key interest of financial economists, as literally all of modern financial theory depends on the assumptions of efficient markets. In this framework, financial markets are thought of as random processes, whose dominant component is a potentially time-varying Gaussian perturbation [3]. But recently a line of inquiry has opened up exploring the excess returns provided by simple moving average trading rules. The pioneering work was done by Brock, Lakonishok, & LeBaron [8], demonstrating that a simple moving average trading rule produced significant excess returns when applied to the S&P 500, and LeBaron [5] intriguingly extends these techniques to foreign exchange markets. Researchers have stepped beyond traditional moving average rules to examine the ability of genetic programming [4] to automatically generate more complex trading rules. Allen & Karjallainen [1] explored this approach on the S&P 500 and found rules that earned excess returns over a buy-and-hold strategy. Neely et al [7], [6] used similar methodology in investigating foreign exchange markets and again showed excess returns.

While the economists are interested in addressing issues of market efficiency, computer scientists are in-

*Thanks to Blake LeBaron, Andrew Moore, Matt Glickman, Bryan Routledge, and John Miller. Sponsored by NSF grant IRI-9612131 and ONR grant N-00014-96-1-1222. Copyright 1999, American Association for Aritificial Intelligence (www.aaai.org). All rights reserved.

terested in optimizing the search algorithms used to produce the best results possible. While the results of Allen & Karjalainen and Neely et al are strong, they use GP as a black box to answer economic questions about market efficiency, with little thought as to the design of that black box. This paper attempts to open the black box to examine how the genetic program is doing its job and how to get it to work better.

As such, little space will be devoted towards proving the economic credentials of this work; the focus will primarily one of comparison of excess returns across different variants of the basic algorithm.

3 The system

3.1 Task and Data

We are interested in the profitability of trading rules; specifically, rules that look at past data and give us a 'buy' or 'sell' signal. For data, we used the average of closing bid and ask prices for dollar/dm and dollar/yen foreign exchange price series, dating from 2/2/1979 to 12/29/1995. This gave us 4324 price quotes. We somewhat arbitrarily took the first 1000 datapoints for training, the next 500 for validation, and we used the next 2750 for a test set (we threw away 74 datapoints because we cut the test set up into 250 point slices, for an experiment we describe in section 4.3).

3.2 Fitness

The profitability of a trading rule is measured in terms of excess returns. Excess returns are a measure of how much money we gain by trading, minus the cost of the capital we have to borrow to trade with.

Consequently, we use excess returns as our measure of fitness. We measure the log excess returns (in the finance literature, returns are traditionally converted into log form for easy manipulation) of a long position (borrow dollars and buy foreign currency) as follows. Denote the exchange rate at time t by S_t, the domestic interest rate by i_t and the foreign interest rate by i_t^f. Then our returns are the gains from trading plus the differential in interest rates:

$$r_t = lnS_{t+1} - lnS_t + ln(1 + i_t^f) + ln(1 + i_t)$$

For a short position (borrow foreign currency and buy dollars) we simply take the negative of this. We make no provisions for transaction costs.

3.3 The algorithm

The system follows a standard genetic programming framework. Our data structures deliberately mimic those of traditional moving average trading rules. These rules usually involve comparisons between moving average, max, and min statistics computed over past price signals. A typical rule might be:

If the current price is greater than the 150-day moving average of past prices, take a long position; otherwise take a short position.

This is an example of a '150 day moving average' rule.

In order to allow for the representations of such rules (and more complex variants), we set up our trees with the following kinds of nodes:

- Leaf nodes all consist of a function of the current price series: moving averages, maxes, or mins with time windows varying from 1 to 250.
- Higher level nodes consist of two types: comparison nodes ($<$ and $>$) and
- logical nodes (logical and, xor, not).

We set up our operations on the trees (crossover and mutation) so that only structures that 'make sense' were created: where 'make sense' means that comparison nodes always have leaves as children, and logical nodes all had comparison nodes as children. This essentially allows arbitrary logical combiations of traditional moving average rules while ruling out potentially absurd combinations like less-than comparisons between logical values.

For reasons of computational efficiency, we precomputed the min, max, and moving average functions over 18 different windows of sizes ranging from 3 to 500. By caching these results, we greatly speeded up fitness computations.

The high-level algorithm goes as follows:

- Split data into training, validation, and testing sets
- Generate beginning population of rules
- Pick two rules (weighted by fitness) and perform crossover
- Pick one rule at random and mutate it.
- For each of the three new rules generated by crossover and mutation, pick a single existing rule and compare fitness on the training set; if the new rule has greater fitness, keep it.
- Keep track of the average rule-fitness over the validation set (excess returns generated by that rule)
- Repeat until average rule fitness over the validation set starts to decline
- Evaluate rules over test set.

There are two key differences between this algorithm and a traditional genetic programming approach. The first is the lack of a traditional generation structure; instead of the entire population undergoing crossover and mutation, it is one operation at a time. There is a good reason for this; as we will see later, search in this domain progresses very quickly, and in order to examine what's happening, finer resolution than the level afforded by a traditional generation structure is needed.

The second is the presence of the validation set, and its use in determining when to end search. In contrast to many problems to which genetic methods are applied, we are looking for the ability to generalize; and if we purely optimized over the training set we would tend to 'overfit' the data. What we use is similar to methods traditionally used by neural networks and other machine learning methods [2].

4 Experiments & Empirical Results

We are interested in understanding how variants of our basic system function comparatively. In particular, we want to address the folloing questions:

- How does rule complexity of affect performance?
- Can we use additional validation methods to protect against overfitting?
- Should we try to accound for nonstationarity?

The sections that follow describe our empirical investigations of these questions applied to our basic system. We ran simulations over both the dollar/yen and dollar/dm data and evaluated excess returns over the training sets. All simulations are averaged over 30 trials.

4.1 Rule complexity

Intuitively, controlling the size of the trees representing the rules offers a tradeoff: bigger trees means more representational capability at the cost of potential overfitting. To test this, we varied the parameter that controlled the maximum depth of the tree, allowing trees of depths 5, 3, and 2.

The table below presents the results, in terms of annualized percentage returns for the system with differing maximum tree depth sizes.

Depth:	$/yen	$/DM
5	3.94%	.85%
3	2.83%	.61%
2	8.10%	1.14%

The results here are striking; in the dollar/yen case, tree size of 2 is clearly superior, and looks mildly superior in the dollar/DM case. Eight percent excess returns are pretty impressive, and are in line with some of the better results from traditional moving average rules [8]. It is worth thinking about what this means – when limiting trees to depth 2, the only sort of rule that is possible is that of a simple comparison – comparing one moving average, max, or min statistic against another. In the battle between overfitting and representational power, representational power clearly loses. For the dollar/yen case T-test values comparing the means of the depth 2 case with the depth 3 and 5 cases have significance levels below .001; for the dollar/dm case the results do not show statistical significance.

In the following sections, we discuss many variations on the original algorithm; however, the pattern above – tree depth 2 providing the best results– is so pervasive we will not argue much more for it.

4.2 Validation

In our standard model, the validation set is used only to determine when to cutoff search. However, there are additional possible ways to use this information that might help increase returns. Neely et al [7], [6] used the following methodology: after he stopped search on the genetic algorithm, he looked at the rule with the best performance over the training set. He then examined that rule's performance on the validation set. If that rule produced positive returns, he kept it, otherwise he threw it out and started over.

Thus inspired, we added tried two additional validation techniques. One was very similar to Neely's; once search was stopped using the method described above, we isolated the rule that performed best on the training set and examined its performance on the validation set, keeping it if it produced positive returns. We label this approach the 'single-rule' approach. Our second approach is similar, but instead of examining a single rule, we examine the performance of every rule in the population over the validation set, and keep all that produced positive returns. We label this approach the 'multiple-rule' approach. We present the results before for the two additional methods (we still use the validation set to determine when to cut off search). The first table covers the dollar/yen market; the second, dollar/dm:

	dollar/yen		
Depth:	standard	single	multiple
5	3.94%	1.22%	3.37%
3	2.83%	2.59%	1.54%
2	8.10%	6.85%	6.98%

	dollar/dm		
Depth:	standard	single	multiple
5	.85%	-.39%	.31%
3	.61%	.23%	.22%
2	1.14%	.94%	1.22%

The results here are surprising. In only one case (multiple-rule with tree depth of 2 on the dollar/dm) did the additional validation help at all, and that difference is marginal at best. In every other case, the additional attempt at validation made things worse. The differences are not statistically significant, but the fact that the additional validation provides no help is unexpected.

4.3 Time dependence

It is widely accepted that financial data is highly nonstationary, although pinning down exactly what that means is difficult [3]. But in our system, rules are used almost ten years after the data that generated them. We wanted to explore if this made a difference by reworking the validation system: instead of using a fixed window immediately after training, we used a 'sliding' window. We sliced the test set into chunks of 250 datapoints. For the first 250 points, we used the original 500 point validation set. Then, for the next 250 steps, we moved the validation set forward 250 steps, and so on throughout the data, so that the validation set was always the 500 datapoints immediately preceeding the test set. The results are presented below:

	dollar/yen		
Depth:	standard	single	multiple
5	1.25%	.03%	2.16%
3	3.07%	1.21%	3.07%
2	6.88%	6.86%	7.08%

	dollar/dm		
Depth:	standard	single	multiple
5	.42%	-1.00%	.72%
3	.39%	-1.11%	.78%
2	1.30%	1.10%	1.51%

This table is qualitatively similar our results with a fixed validation set, presented in section 4.2, with a couple of minor differences: performance on the dollar/yen market with depth 2 tree is poorer in the standard case. In the dollar/dm markets, returns for the depth 5 and 3 trees is inferior, but the depth 2 tree shows small improvement. But the important thing is that allowing for the validation set to 'keep up' with the test set produces no significant improvements.

4.4 Why can't we validate better?

We find the result that using only rules that produced positive results on the validation set resulted in poorer test set performance very puzzling. This subsection we proposes a possible explanation. By no means do we have a definitive answer; what we do here is intended to be suggestive.

We looked at correlation between rule performance on the validation set and rule performance on the test set. The inution is that if using the validation set to weed out ovefitted rules worked, we would expect to see a positive correlation, indicating that poor performance on the validation set indicated poor performance on the test set, and vice versa. However, since using the

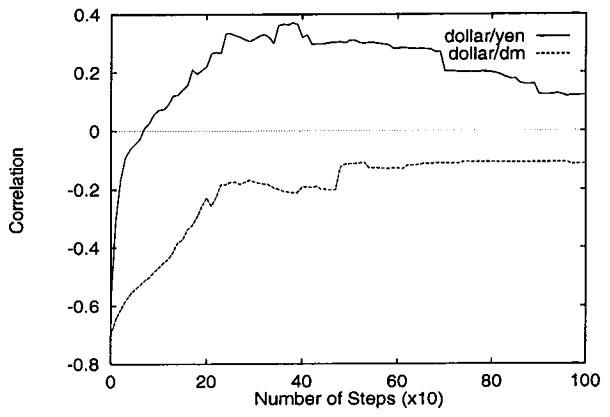

Figure 1: Correlation between performance on validation and training sets

validation set to weed out poor performing rules is actually counterproductive, we expect to see the opposite. Figure 1 shows a graph of correlation across number of steps for the tree depth 2 case, both dollar/yen and dollar/dm.

Strangely, at the beginning of search, with completely random rules, the correlations are strongly negative; as search continues, the correlations rise. In the dollar/yen case, the correlation becomes positive, and in the dollar/dm case, they stay negative, but close to zero. This measurement of correlation relates to our concerns of validation as follows. In the dollar/yen case, the median correlation between rule fitness on the validation set and fitness on the test set at the point of search cutoff over the 30 trials is 0. This suggests that in half of the cases, there was zero or less correlation between validation set performance and test set peformance. For the dollar/dm case, the median was -.4412, indicating that there was a negative correlation in most cases.

Given this information, we should not be surprised to find that using the validation set to identify bad rules will fail as often as not at the point of search cutoff.

However, given that the correlation rises as search progresses, one might expect the ability of the multiple-rule validation technique would improve accordingly. Figures 2 and 3 plot the excess returns against number of search steps taken for the tree depth 2 case over the validation set and over the test set – for both the standard approach and the multiple-rule approach.

Examining these graphs roughly confirms are expectations. In both cases, around step 230, the multiple-rule method starts to produce superior performance; and, this is approximately when the correlation graphs for each level off. The match isn't perfect: the dollar/dm correlations are still negative, despite some gains made in the test set performance, but it is suggestive. In addition, the test set performance in the dollar/yen case

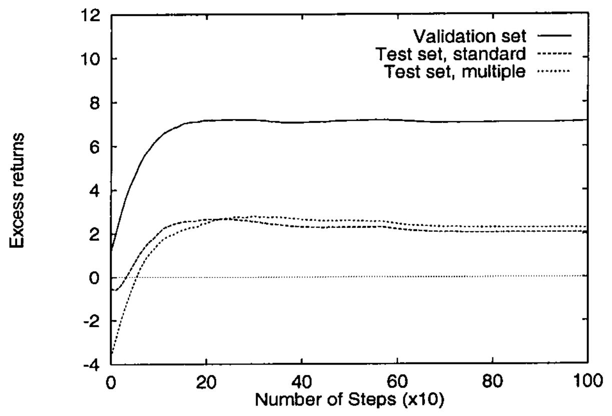

Figure 2: Excess returns for rules on validation and test sets for dollar/yen

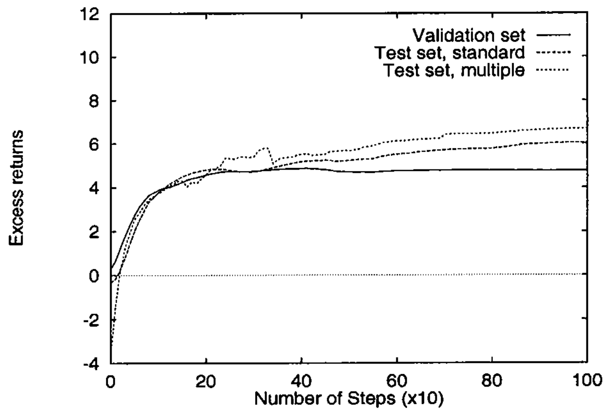

Figure 3: Excess returns for rules on validation and test sets for dollar/dm

shows no sign of overfitting; these two facts suggest that additional criteria for identifying a cutoff point for search need to be investigated.

5 Conclusions & Future work

What can we conclude from the results here? First, it is possible to extract rules with out-of-sample excess returns, at least in some domains: we get very clear results from the dollar/yen market but not from the dollar/dm market. However, some caveats apply.

The most important issue here is that of tree size. Given that trees limited to a depth of two outperformed larger trees, we must wonder if the full representational power offered by the genetic programming framework may be a hindrance rather than a help (at least with the sparse feature set presented here). Given how fast the system finds good solutions (in less than ten generations, usually) and the simplicity of such solutions, a more enumerative search method may be preferrable. Still, we see promise in the genetic programming framework, for its potential ability to integrate multiple streams of data – an active research area is using genetic programming to integrate data about interest rates and even text information into the automated rule discovery mechanism. Future work lies in attempts to find a way to encourage simplicity rather than mandate it. Neely [7] suggests that using high costs during training would encourage simpler strategies that didn't trade as much and provides some encouraging, but highly preliminary results along those lines. Adjusting the fitness mechanism to bias towards simplicity is another potential approach.

The validation issues raised by these experiments are puzzling. It seems that neither allowing the validation window to move with the test set, nor trying to use information from the validation set to select better rules provides much gain. However, a closer look shows that gains might be found in a new criteria for stopping search. We examined the relationship between validation set fitness and test set fitness and found correlations that are often negative. Unfortunately, this would seem to make use of more validation information for performance improvement difficult. It seems likely there is a sort of regime change occuring in the data – understanding and taking advantage of this structure in the data is a key to future research.

References

[1] Franklin Allen and Risto Karjalainen. Using genetic algorithms to find technical trading rules. Technical report, Rodney L. White Center for Financial Research, 1995.

[2] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[3] John Y. Campbell, Andrew W. Lo, and A. Craig MacKinlay. *The Econometrics of Financial Markets.* Princeton University Press, 1997.

[4] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[5] Blake LeBaron. Technical trading rule profitability and foreign exchange intervention. *Journal of International Economics*, forthcoming 1998.

[6] Chris Neely and Paul Weller. Technical analysis and central bank intervention. Technical report, Federal Reserve Bank of St. Louis, 1997.

[7] Chris Neely, Paul Weller, and Rob Dittmar. Is technical analysis in the foreign exchange market profitable? a genetic programming approach. Technical report, Federal Reserve Bank of St. Louis, 1997.

[8] Josef Lakonishok William Brock and Blake LeBaron. Simple technical trading rules and the stochastic properties of stock returns. *Journal of Finance*, 47(5):1731–1764, 1992.