Integrating Genetic Algorithms and Text Learning for Financial Prediction

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

This paper takes two approaches to prediction of financial markets using text data downloaded from web bulletin boards. The first uses maximum entropy text classification to predict based on the whole body of text; the second uses a genetic algorithm to learn simple rules based solely on numerical data of trading volume, number of messages posted per day and total number of words posted per day. While both approaches produce positive excess returns in some cases, it is found that integrating the two predictors together produces far superior results. Furthermore, aggregating multiple GA trials to build single predictors increases performance even more.

1 Introduction

There has long been a strong interest in applying computational intelligence to financial data. Traditionally such attempts have been concerned with forecasting the future based on past price data. However, recently a new source of data has become available: text. Not only have large amounts of financial text data become available on the web, but the AI community has seen an explosion of development of the tools needed to automate its analysis. This paper aims to integrate novel methods of text analysis with existing methods of using genetic algorithms for learning simple rules based on numerical data.

The format of this paper is as follows. In the next section, we describe the task of learning profitable trading rules and the data we used – text downloaded from internet stock chat boards with corresponding daily closing stock prices and volume figures. The next two sections present two competing approaches: a text classification algorithm for prediction, and a simple genetic rule learner based solely on numerical data (including measures of the volume of text generated). The following section attempts to integrate the two approaches with strong performance increases, and then we present a brief summary and extensions.

2 Task and Data

We are interested in the profitability of trading rules rather than accuracy of prediction itself – the economics literature is concerned with market efficiency instead of prediction accuracy, and the ability of a trading strategy to make consistent excess profits is a strong sing of market inefficiency. Therefore, although are algorithms will give us 'up' or 'down' predictions, we will map these predictions onto a trading strategy. Specifically, if our algorithms predict up, we buy or hold the stock (depending on if we already own it). If our algorithms predict down, we sell the stock or hold a cash position.

Our measure for the fitness of such a trading rule will be excess returns – the measure of how much money we gain by following the trading strategy dictated by our algorithms instead of merely buying and holding the stock over the appropriate period (this is one of the standard measures of market inefficiency in the literature). Calculating the daily log excess returns is simple (in the finance literature, returns are traditionally converted into log form for easy manipulation). Denote the price of a stock at time t by $P(t)$. Let $S(t)$ denote an indicator variable for our prediction signal: $S(t) = 1$ if we are issued a 'buy/hold' signal, $S(t) = 0$ otherwise. Then our excess returns are simply the returns from our trading strategy minus the returns for holding the stock:

$$r(t) = S(t)*(lnP(t+1)-lnP(t))-(lnP(t+1)-lnP(t))$$

This simplifies to

$$r(t) = (S(t)-1)*lnP(t+1)-lnP(t))$$

We make no provision for transaction costs.

For data, we are specifically concerned with using text

[*]Thanks to Blake LeBaron, Andrew Moore, Matt Glickman, Bryan Routledge, John Miller, and Somesh Jha. Sponsored by NSF grant IRI-9612131 and ONR grant N-00014-96-1-1222

information for prediction. As such, we took the forty most popular discussion boards on the financial website www.ragingbull.com for the week of November 18, 1999. We eliminated all boards not devoted to a specific stock, all boards whose stocks were not trading on NASDAQ or the NYSE as of Jan 1, 1999, and those boards with stock prices of less than \$1 on Jan 1, 1999. This left us with 22 stocks and the accompanying text from their bulleting boards.

We downloaded every post for each one of the bulletin boards from January 1, 1999 to December 31, 1999, and downloaded daily closing prices and trading volume for each stock from the quote server at fiance.yahoo.com. For each day t, we aggregated all text produced after market close on day $t-1$ and before market close on day t[1]. This data provided the raw material for our prediction algorithms – text produced before close on day t could be used to predict whether the closing price on day $t+1$ would be higher or lower, and an appropriate trading strategy implemented.

This one year of data gave us 252 training days. In all the results that follows, we use the first 52 days as the start of the training set and report results averaged over the last 200 trading days.

3 Text Classification

For text classification, we used the rainbow package developed by McCallum [4], which provides a variety of potential classification methodologies. Space constraints limit us to presenting a high-level overview of the technique; a detailed explanation can be found in [6]. Since the focus of this paper is on the integration of the text classification with genetic methods using other data, we feel that this is not a problem.

In general, text classification techniques work as follows. The entire training corpus is lexed, and the number of occurances of each word is calculated for each document, producing a large matrix. This can be visualized as an n-dimensional space where each dimension corresponds to a specific word in the corpus (here, n is the number of unique words in the corpus). Each document's position on a specific dimension is determined by the number of occurances of the word corresponding to that dimension in the document. Since we now have a set of labelled examples in n-dimensional space, we are left with a simple classification problem, to which various algorithms like naive bayes classification [3] can be applied. Maximum entropy text classification is simply the application of maximum entropy techniques to this specific classification framework [6].

Starting at trading day 53, we algorithm used the as follows. For day t, we inserted the text for days $[1 \ldots t-1]$ to the training set, trained the classifier, and had it give probabilities for 'up' or 'down' based on the text for day t. If the 'up' probability was greater than .5, we issued a 'buy/hold' signal; otherwise, a 'sell' signal. As we moved to day $t+1$, we added the text from day t to the training set and retrained the classifier.

[1]For simplification data produced on non-trading days was discarded

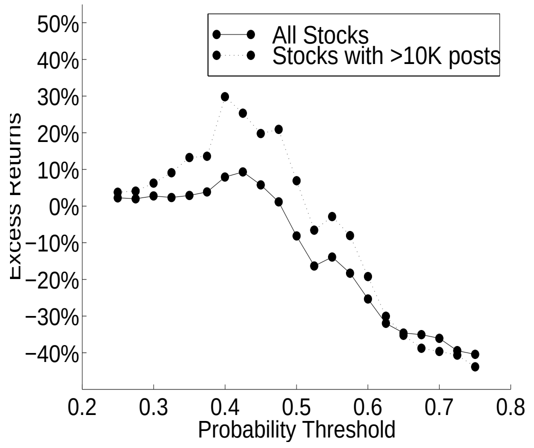

Figure 1: Excess returns plotted against probability threshhold

Results, averaged over the last 200 trading days, are presented in the table below. We split the results into two clases, presenting the excess returns (and p-values for a two-tailed ttest against the null hypothesis of a buy-and-hold strategy) for all 22 stocks as well as those for which we only had more than 10000 posts over the year. The amount of data (measured by the number of posts in the dataset) for each stock varies greatly, from SNMM (158K posts) to EMC (2.8K posts). One would expect that the more posts we have from a stock, the better we are able to classify it, and this seems to be true in the table below. Excess returns over all stocks are negative, but excess returns over those stocks with more than 10K posts (12 stocks out of 22) provides us with small excess returns. These results are promising, but not yet satisfactory.

Performance by Maximum Entropy text classification	
Stock:	Excess Rets
All stocks (22 stocks)	-8.11%
>10K posts (12 stocks)	6.91%

The maximum entropy text classification algorithm doesn't provide a simple 'yes/no' answer for each document; it outputs a probability. Although it is conventional to classify those documents with a predicted 'up' probabilities of greater than .5 as being in the 'up' class, there is no rule in stone saying that we must do so. The plot in figure 1 plots the excess returns generated by the a trading strategy where the probability threshhold used for predicting an 'up' day is allowed to vary from .5 along the x-axis.

If one examines the returns produced from trading rules that use a threshhold approximately between .4 and .5, one finds that large excess returns are produced – between 20% and 30% in the >10K posts stocks case, and over 5% in the all stock case – a significant improvement from negative excess returns. This would indicate that the text classification algorithm is in fact extracting useful information, but that our approach of using .5 as the threshold is wasting most of it. Unfortunately, it would be unacceptably ad hoc to simply go back and fiat a trading rule that uses a lower probability threshold. The key to exploiting this information will be the integration of the text classification technique with other predictors, a topic we explore below in section 5.

4 Genetic Methods for Learning Trading Rules

There has been a recent interest in using genetic techniques to learn simple trading rules in financial markets based on past price data [1, 5], primarily motivated by a desire to explore the efficient market hypothesis. These techniques have proved surprisingly effective, and have proved to be an interesting starting point for further research [7].

Since we are interested in exploring the value of the text data, we will not use past prices as the input data for our rule learner: instead, we will use trading volume and two numbers derived from a given day's text: the number of messages posted in a day, and the total number of words posted in a day.

The rules that we want to learn, that map past information onto 'buy/hold' and 'sell' signals, have the following form: let v be today's volume (although it could be total messages or total words), $ma(v, n)$ be the n day moving average of past volume values, and k be an arbitrary constant in the range [0,2].

Then each rule has the following form:

if $v > k * ma(v, n)$, issue a 'buy/hold' signal.

Although these rules seem almost trivially simple, they are similar in spirit to those of technical analysis [2], and searches over combinations of rules of this form is the standard approach in the prediction literature discussed in the introduction.

Since recent work [7] has emphasized the importance of simplicity in dealing with the incredibly noisy nature of financial data, we will limit our search to extremely simple combinations of rules: two rules, logically combined with either 'and' or 'or', and also potentially negated.

To search over this rule space we used an extremely simple GA. We had a population of 10 rules, deterministic tournament selection, no crossover, and Gaussian noise applied for mutation. For a fitness function we measured the excess returns that would have been produced by the given rule over the training set. The actual parameter space is composed of the potential parameters of k, n (the size of the moving average window) and which datastream the rule applies to (trading volume, daily message volume, or daily word volume).

We applied the genetic algorithm as follows: at time t, evolve the population for 10 generations, then allow the final population of rules to vote on the next day, and calculated the returns accordingly.

For computational efficiency reasons, we did not update the training set every trading day; rather, we updated every 25 days. The nature of the genetic algorithm gave us a unique advantage, however – once we had our initial population, we carried it over each time we retrained. That is, since it is likely that rules that scored well during the first 100 days would score well during the first 125 days, starting with the old population seemed like the ideal population seeding. This is why we used so few generations each time we retrained – since we already had a partially trained population, we were afraid of excessive overfitting.

The are the excess returns produced by the algorithm, averaged over 200 trials, are presented in the table below:

Performance by Genetic Rule Learner	
Stock:	Excess Rets
All stocks (22 stocks)	-6.22%
>10K posts (12 stocks)	5.948%

These results are similar to those produced by our text classification algorithm above – some positive excess returns for the >10K posts case, but negative excess returns in the general case. Again, promising, but not quite what we'd hoped for.

5 Integration

So far we have seen two promising methodologies for building trading rules over this dataset. The next step is to attempt to integrate them. Remember that both of our algorithms give us numbers which we compare to a threshold of .5 – for the text classification example, we have an explicit probability calculation; for the genetic approach, we have the percentage of the population that issues a 'buy/hold' signal.

Our approach is to simply take a weighted average of these two predictors and compare them against our threshold of .5. However, we have to be careful not to weight them evenly – since the maximum entropy probabilities are typically in the range of [.4, .6], and the percentage of the population that predicts up is usually close to zero or one, combining them evenly would leave us with a predictor heavily dominated by the genetic rule learner.

As a result, we settled on a weighting of .9 for the text classification predictor and .1 for the genetic simple rule learner. The results are presented in the table below.

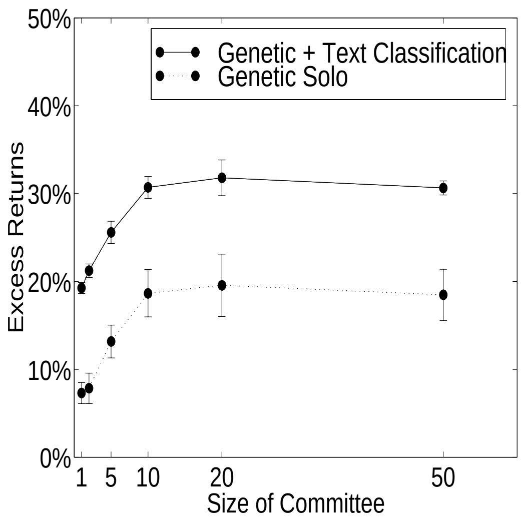

Figure 2: Excess returns plotted against number of GA runs averaged together

Performance by Integrated Approach		
Stock:	Excess Rets	std. dev.
All stocks (22 stocks)	2.88%	4.75
>10K posts (12 stocks)	19.26%	8.84

These results are much stronger. The >10K case is strongly statistically significant with a p-value of less than .0001 (against a null hypothesis of zero excess returns), and the all stock case is at least positive.

5.1 Averaging Multiple GA runs

Financial data is notoriously noisy; as one way to combat this we tried aggregating multiple GA runs into single predictors. Instead of treating each run of the genetic algorithm individually, we averaged the predictions over n runs and used this number to feed into our integrated predictor.

The results are plotted in figure 2, excess returns plotted against the number of GA runs we aggregated to form the predictor, n, along with standard error bars. We present only the results from the >10K post stocks here, and the plot contains both the results from the integrated approach as well as the stand alone genetic rule learner approach.

The results here are clear: aggregating many GA runs into a single predictor strongly improves performance, both with the GA by itself and as part of an integrated approach – in the integrated approach, the excess returns exceed 30% for committee sizes of 10 and larger.

6 Conclusion

This paper has described two ways of data mining a new source of data for financial prediction, web bulletin board postings. We presented both a straightforward maximum entropy text classification algorithm as well as a genetic algorithm for learning simple rules based solely on numerical data (including both traditional technical measure of volume as well as counts of messages and words from the text dataset).

We have shown that by both judiciously integrating our predictors together, and by averaging our predictions over many trials of the genetic algorithm, we can improve the excess returns from less than 10% to over 30% on the class of stocks that gives us the most text data to analyze.

The next obvious step is to extend this approach to other sources of text – other web bulletin boards like those on yahoo.com, but also financial news stories. However, given the extreme amounts of noise in financial data, we feel that integration between multiple data sources and techniques is key to progress in data mining for finance, and the techniques presented in this paper are a key first step.

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

[3] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[4] Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.

[5] 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.

[6] Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.

[7] James D Thomas and Katia Sycara. The importance of simplicity and validation in genetic programming for data mining in financial data. In *Proceedings of the joint GECCO-99 and AAAI-99 Workshop on Data Mining with Evolutionary Algorithms: Research Directions*, 1999.