AUTOMATED STOCK TRADING USING MACHINE LEARNING ALGORITHMS

Tianxin Dai, Arpan Shah, Hongxia Zhong

November 16, 2012

1. Introduction

The use of algorithms to make trading decisions has become a prevalent practice in major stock exchanges of the world. Algorithmic trading, also called automated trading, black-box trading, or algo trading, is the use of electronic platforms for entering trading orders with an algorithm deciding on aspects of the order such as the timing, price, or quantity of the order, or in many cases initiating the order without human intervention [Wikipedia]. Competitors across different companies try to edge out each other with more and more statistically advanced algorithms that use more data, and quicker execution to beat out the competition. Machine Learning has therefore been central to the process of algorithmic trading which provides an interesting problem with numerous variables.

In particular, variations between stock exchanges, between stocks and between days makes it difficult for one strategy or a single algorithm to be devised that works across the board. Regardless of this known limitation due to the inherant randomness of the markets in this project different strategies are able to still monetize on the exchange with advanced algorithms. The algorithms that are utilized range from various supervised learning algorithms such as regressionm, SVMs (Support Vector Machines) to unsupervised algorithms such as Random Forest. One of the most complicated challenges is choosing the right set of features to model the algorithm with. Feature selection from the vast amounts of real-time data such as 'Trade Volumes' , 'Bid-Ask ratios', 'High-Low variations' and getting the features right becomes crucial in having a successful trading algorithm.

2. Objectives

We aim to devise a machine learning algorithm by starting from a very basic model and successively improving it and the algorithm. We plan to focus on feature selection to approach a solution that is able to trade effectively on a selected universe of stocks. To achieve this objective, we plan on emphasizing our attention into choosing and validation of features using strategies such as **Filter Feature Selection** and **Forward search.** The emphasis on feature selection for this problem comes from the fact that "Not all features have the same predictive potential"[1]. In particular, given the nosiy nature of Stock Ticker data it is important to make sure that over-fitting doesn't occur as that would increase the influence of the noise and lead to poor predictive behavior. Under-fitting is equally dangerous in this scenario as it would provide an equally poor performance.

3. Data: Training and Testing

Definition and Collection

After speaking to a TA(Keith) from CS246 who had worked on similar problems and reading a bit about stock universe selection, we decided to initially look at a very conservative set of stocks as we didn't want to first have an algorithm and then find the stocks to which it would fit. We selected data based on the following criteria:

a. Price between 10-30 dollars in the last thirty days

b. Membership in the last 300 of S&P500. (so not the 200 largest names because these we found had maximum speculative buying).

c. ADV in the middle 33% - To ensure that the stocks being picked were not being traded too few times or too often to reduce errors due to incredibly high-volume or low volumes of trade

We utilized the Bloomberg terminals available to Stanford students for research and academic purposes, and spent some time learning the system to be able to query for the stocks that would fit into our desired universe. After choosing our target stock universe we discovered a list of 26 stocks to test our classification algorithms on[1]. We generated 30 different lists of stocks for a 30-day period to evaluate our algorithm's performance, and we always select on day $t-1$ the universe to trade on day t in order to avoid look-ahead bias.

Pre-Processing

Upon obtaining the data in an excel spreadsheet, we created matlab scripts to be able to correctly parse the data and store it into quick .MAT files and massaged fields in the Excel spreadsheet such as date and time to be able to quickly process the ticker-data as a time-lapsed field and be able to utilize them effectively across our matlab scripts.

4. Building Models and Experimentation

Mean Reversion

After speaking to Keith, The first strategy we utlized was a naive strategy meant to familiarize ourselves with the data and test a very simple model - Mean Reversion. In this strategy, we looked at every tick in the trade data looking at the previous two prices for every tick. We predicted at every step that an uptick follows a downtick, and a downtick follows an uptick, i.e if $P(t-1) > P(t)$ then $P(t+1) < P(t)$ and if $P(t-1) < P(t)$ then $P(t+1) > P(t)$. If $P(t-1) = P(t)$, then we carry our previous forecast forward. If we were predicting a downtick for $P(t)$ and got $P(t-1) = P(t)$, then we would continue forecasting downtick for $P(t+1)$. Although this strategy is indeed naive, it is still based on the behvior of the stock-market because on the tick level the prices bounce between the bid and the ask price[2].

As expected, the error percentage of this strategy was pretty high and we had an accuracy of only 11.2% on our data.[2]

Regression

The next strategy we explored was implementing a regression model with features based on the rolling 5-minute high-low spread. In this model we used dynamic programming to cache high and low for five 1-minute-buckets and calculate the 5-minute high-low based on those buckets. Note that this is only an

[1] List Attached in Appendix

[2] Code for strategy attached in Appendix

approximation since our window could range from 4 minutes to 6 minutes, but is still good enough for learning. Then we classify each tick as a big move if the move is more than 50% of the rolling 5 minute high-low spread, and a small move if it is less than 50% of the spread. And we regress the percentage move as a linear function of large move and small move. Specifically,

$$Y = \theta_0 + \theta_1 x_{large} + \theta_2 x_{small}$$

$$\text{where } x_{large} = \begin{cases} 1, & p(t+1) - p(t) > 0.5\Delta HL \\ -1, & p(t+1) - p(t) < -0.5\Delta HL \end{cases}, x_{small} = \begin{cases} 1, & p(t+1) - p(t) < 0.5\Delta HL \\ -1, & p(t+1) - p(t) > -0.5\Delta HL \end{cases}$$

5. Further Work Planned Before Final Submission

The next steps we plan to undertake should improve our accuracy even further. We plan to implement a SVM model for this classification problem. In addition, significant time will be spent on choosing, and optimizing the feature set for the set of stocks that have been chosen as the universe. The methods of analysis and improvement to decide and choose features will be used and then mock trading sessions will be run to discuss the efficacy of the algorithm designed in the end.

6. Reference

[1]Winning the Kaggle Algorithmic Trading Challenge with the Composition of Many Models and Feature Engineering,IEICE Transactions on Information and Systems, vol.E95-D, no.10, 2012

[2]CS246 Hw4 Handout

7. Appendix

Sample Universe of Stocks Fitting Criteria:

Stock Ticker	Origin
CNP	US Equity
SLM	US Equity
KIM	US Equity
XL	US Equity
TXT	US Equity
LNC	US Equity
TSN	US Equity
NWL	US Equity
CBG	US Equity
NYX	US Equity
UNM	US Equity
HRB	US Equity
GCI	US Equity
CVC	US Equity
SAI	US Equity
ZION	US Equity
JBL	US Equity
IGT	US Equity
SEE	US Equity
WPX	US Equity
TER	US Equity
GME	US Equity
THC	US Equity
APOl	US Equity
PBI	US Equity

Mean Reversion:

```
% Implements mean-reversion and evaluate accuracy
    % 1 is uptick, and -1 is downtick
    % Initialize our first forecast to be an uptick
    function accuracy = mean_reversion(price)
    prev_forecast = -1;
    accuracy = 0;
    for i=2:numel(price) - 1,
    % If previous tick is uptick, then forecast downtick
    if price(i-1) < price(i)
    forecast = -1;
    % Forecast uptick if prev tick is downtick
    elseif price(i-1) > price(i)
    forecast = 1;
    % Otherwise, retain our previous forecast
    else
    forecast = prev_forecast;
    end
    % Now look at the actual tick and update our accuracy score
    if (price(i+1) > price(i) && forecast == 1) || ...
    (price(i+1) < price(i) && forecast == -1)
```

```
accuracy = accuracy + 1;
elseif (price(i+1) ~= price(i))
accuracy = accuracy - 1;
end
end
% report accuracy
accuracy = accuracy/(numel(price) - 2);
```