Machine Learning For Trading: MC3-P3

Andrew Cassidy, *acassidy6@gatech.edu, Georgia Institute of Technology*

This paper concerns using technical indicators to predict the future price of stocks. I cover 3 technical indicators: Simple Moving Average (SMA), Momentum, and Bollinger Bands. Next I combine these technical indicators into a set of rules for going long and short on IBM's stock. I hold all positions for exactly 10 days. My rule-based strategy outperforms buying and holding IBM stock between January 1, 2006 and December 31 2009 (in-sample period). I optimized my rule-based strategy on this in-sample period by picking the various indicator parameters and the rule-based thresholds using hill climbing. Additionally, I built a machine learning based strategy and optimized its hyper-parameters using hill climbing on the in-sample period. This strategy outperformed both the market and my rule-based strategy. I tested both of these strategies on an out of sample period between January 1, 2010 and December 31 2010. My strategies do not beat the market in this case. I will discuss my hypothesis for why in more detail in the concluding section.

Technical Analysis

I used 3 technical indicators: Simple Moving Average, Momentum, and Bollinger Bands to predict the future price of IBM's stock. Figure 1, 2, and 3 all display the performance of buying and holding onto 500 shares of IBM's stock in a portfolio with an initial value of $100,000. This buy and hold portfolio will be used for comparison throughout this paper, but in these figures it is displayed to show how price relates to the different indicators. In retrospect I would have liked to pick a set of indicators that is more diverse than my 3. Bollinger Bands aim to provide a similar indicator to that of SMA and momentum.

Simple Moving Average

Simple moving average (sma) is calculated by taking a sliding window of stock data and taking the average. For example, in order to calculate moving average for January 3, 2006 shown in Figure 1, I needed price data from 13 days in December 2005.

The moving average can be thought of a smoothed trajectory of price and a proxy for true value. SMA can be used to find arbitrage opportunities. Periods where price moves above or below the SMA with high momentum (discussed in the next section) and then recrosses the SMA can be thought of as sell and buy opportunities respectfully. While the rule-based and ML-based strategies I implemented buys and holds IBM stock for 10 days, usually crossing the SMA is an indicator of when to exit an arbitrage strategy. SMA is often combined with momentum.

Momentum

The idea behind momentum is similar to the one of physics: "an object in motion tends to stay in

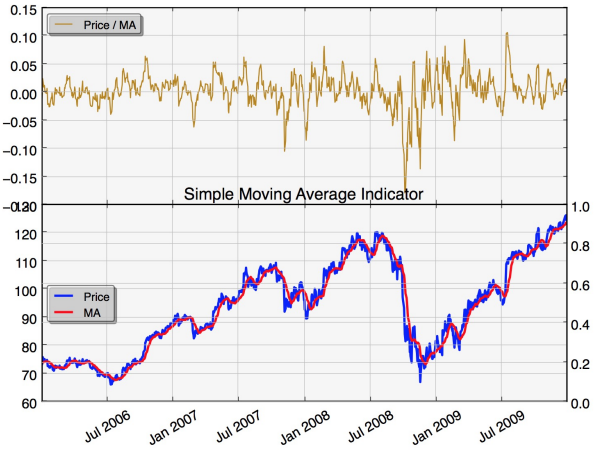

Figure 1: *A simple moving average (sma) with a look-back of 14. The top graph is price's percent deviation from sma and the bottom graph is price and sma. Look-back was one of the hyper-parameters optimized by hill climbing.*

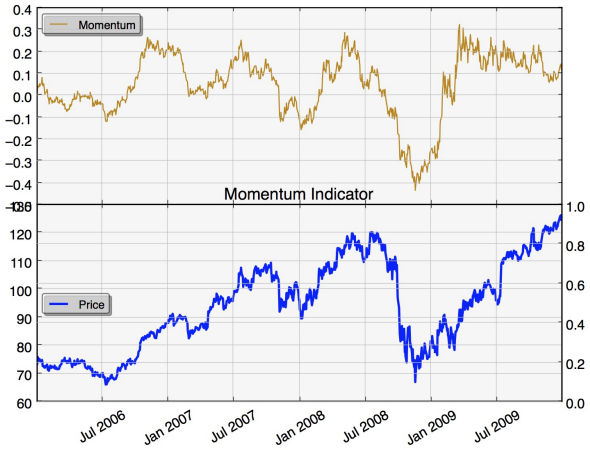

Figure 2: *Momentum with a look-back of 14. On top is momentum and on bottom is IBM's price. A high positive momentum indicates that the stock has been rising for some time and a low negative momentum indicates that the stock has been falling for some time.*

motion". If a stock has been rising for the last few trading days, we should expect that rise to continue and vice versa for a stock that is falling. Momentum for a stock on a particular day is calculated by looking at the price of the stock X (look-back) days in the past and calculating the the percent increase of today's price compared to the look-back price. In figure 2 X=14.

Momentum is often combined with SMA. I combined these two indicators in my rule-based strategy. The idea is that when a stock has deviated above the SMA with a high enough momentum we should sell or buy the stock. I will discuss this in more detail in my rule-based strategy section.

Bollinger Bands

Bollinger Bands are an indicator that combines both SMA and a stock's variation. SMA is calculated in the same fashion as previously described and variation is calculated using a similar sliding window, but instead of taking the average we take the standard deviation. Next then we add or subtract two standard deviations from the SMA and these represent our top and bottom Bollinger Bands. In figure 3, I have highlighted how Bollinger Bands can be used to find arbitrage opportunities.

The idea behind Bollinger Bands is that when a stock leaves one of the Bollinger Bands we either buy or sell the stock and then sell or buy the stock when it re-intersects the Bollinger Band or the SMA. The premise is that when a stock leaves one of the Bollinger Bands it will eventually return to its normal variation and was either being overbought or oversold.

In figure 3 I have highlighted the arbitrage opportunities. Green vertical lines indicate when a price has deviated below the lower bollinger band is returning back while red vertical lines indicated when a price has deviated above the top bollinger band is returning back. These are arbitrage opportunities to buy and sell stock respectfully.

Rule-Based Trading

I used 2 of my indicators to develop an automatic rule based trading strategy: SMA and momentum. I generated a high performance trading strategy using only these two and ran out of time to add in bollinger bands. My rule for trading is quite simple and is shown in figure 4. The premise of my rule based system is that large movements away from the SMA indicate buy or sell opportunities. My strategy does not take into account arbitrage opportunities that bollinger bands indicate. I ended up using bollinger bands in my ML-based strategy and that's one hypothesis as to why my ML-based strategy outperformed my rule-based strategy.

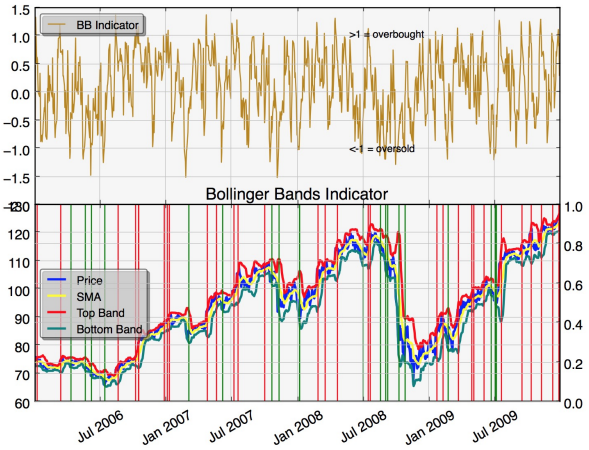

Figure 3: *Bollinger Bands combine aspects of momentum and sma. The top graph represents bollinger bands percentage. The bottom graph shows prices, sma, and the top and bottom bollinger bands. The green vertical lines are buying opportunities and the red vertical lines are selling opportunities.*

```
buy = (sma > sma_upper) & (m > m_upper)
sell = (sma < sma_lower) & (m < m_lower)
```

Figure 4: *My rule based strategy looks at the values of the sma and momentum for a particular time and compares them to thresholds (sma upper, sma lower, m lower, m upper). If both values are above or below a threshold, my algorithm buys or sells the stock respectfully. The variable m represents momentum.*

As I discussed in my technical indicators when a stock deviates from its average with high momentum this can indicate an arbitrage opportunity. It can also indicate that a stock is on the rise or the fall. That's the premise of my system. In order to "tune" my system I ran a program called twiddle from my Artificial Intelligence for robotics class. You can run twiddle.py to see the results. Twiddle is a version of high climbing. Twiddle varies the parameters for SMA and momentum look-back as well as the greater than and less than thresholds for buying and selling, respectfully. These thresholds are shown in figure 4. Twiddle adjusts one parameter at a time and looks at the final portfolio value at December 31 2009. If adjusting the parameter increased the final portfolio output, twiddle will further the step of increase or decrease on the next iteration. If twiddling the parameter caused the final portfolio value to go down, twiddle down-throttles the step of increase or decrease it will try next time. By varying all the parameters in a sophisticated way, I optimized the parameters for my indicators and rules and got the results shown in figure 5.

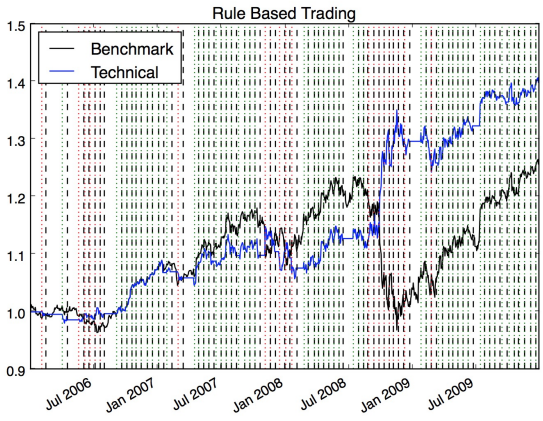

Figure 5: *My rule based strategy compared to the benchmark. Red dotted lines represent shorting a stock and green dotted lines represent going long. Black dashed line represent exiting a position*

My rule-based strategy beats the benchmark. This is largely because it learned to short IBM during the financial crisis of 2008. In the next section, I document how adding Bollinger Bands and letting a ensemble of decision trees (random forest) decide the trading rules further beat the market.

ML-Based Trading

Decision trees for classification learn a combination of rules that indicate a particular class. A very simple decision tree could learn the same rules that I specified in my rule-based section. However, my rule-based section made explicit decisions about how to combine SMA and momentum to determine a class: 'buy' or 'sell'. By letting a ML algorithm learn these rules and potentially more complicated conjunctions and disjunctions I beat the market and my rule-based system for the in-sample period. Additionally, my ML algorithm incorporated Bollinger Bands.

My algorithm learns to 'buy', 'sell', or do 'nothing' by looking at the different technical indicators compared to the predefined class. In order to determine the 'buy', 'sell', and 'nothing' class of what position to take at a particular time, I looked at the 10 day return on an investment (ROI). The threshold for

flagging what is a buy or sell was another parameter to twiddle. You can run ml twiddle.py to see a similar routine to my rule-based twiddle. In this version of twiddle in addition to the previous discussed parameters and the ROI flagging thresholds, I also twiddle the leaf size, number of bags to the ensemble learning. It did not surprise me that my leaf size parameter is around 50. The boolean clauses of my rule-based system performed very well, so the ML-based system probably does not need to learn a super complicated clause either. The results using my optimized parameter set are shown in figure 6.

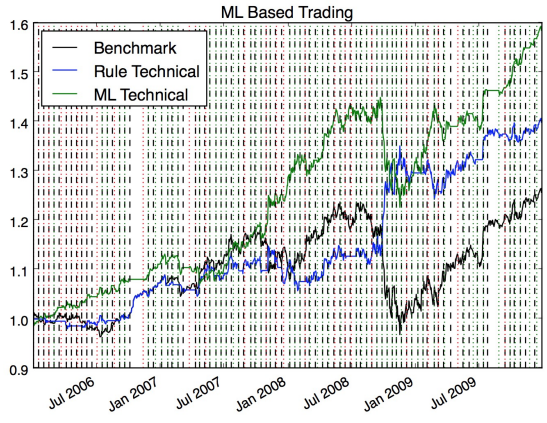

Figure 6: *My ML based strategy compared to the benchmark and my rule based strategy. Red dotted lines represent shorting a stock and green dotted lines represent going long. Black dashed line represent exiting a position*

Out of sample performance

I tested both of these strategies on an out of sample period between January 1, 2010 and December 31 2010. My strategies do not beat the market in this case. Figure 7 and Figure 8 are the same expect that Figure 7 shows my rule-based trades and Figure 8 shows my ML-based trades. Why don't I beat the market? Great question. I think there are two possibilities. First, the financial crisis happened during my in-sample training period. This coupled with a twiddle program that was hell-bent on beating the market made both of my algorithms pretty short heavy (both my algorithsm short stocks a lot). In fact it seems I only lose money on the out of sample periods when I am shorting for both my rule based and ml based strategies.

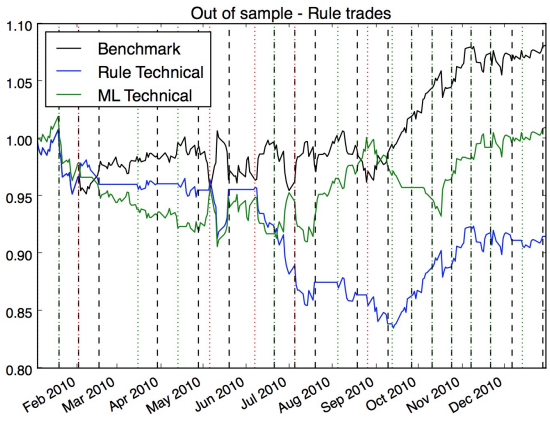

Figure 7: *Performance of my rule based and ml based strategies compared to the market for the out of sample period. Trades are displayed for the rule based strategy*

Second, my out of sample period is a much shorter time frame than my in-sample period and maybe both of these strategies are just better over the long haul. My first point is the most likely and my second is just a conjecture. So how would I fix this? Well traditionally in ML we use a hold out test set to correct for over-fitting. For example, when fitting a neural network we generally see test and training set performance decline up until a certain over-fitting point. After that over-fitting point, we see training set error continue to go down and testing set error go up. I could perform a similar procedure for these trading strategies. Twiddle could run and optimize the hyper-parameters with respect to maximizing the payoff at the end of the in-sample period, but only up to the point where performance on the out of sample period also increases.

Conclusion

This project was awesome and it should be more than 15 percent of the grade. I put so much more time into this project than others. Anyways what did I learn? I learned how to read and interpret technical indicators. I also learned how to combine them into a trading strategy and that you can use classification algorithms to learn these trading strategies for you. What would I do differently in the future? 1) I'd code up a more diverse set of technical indicators and

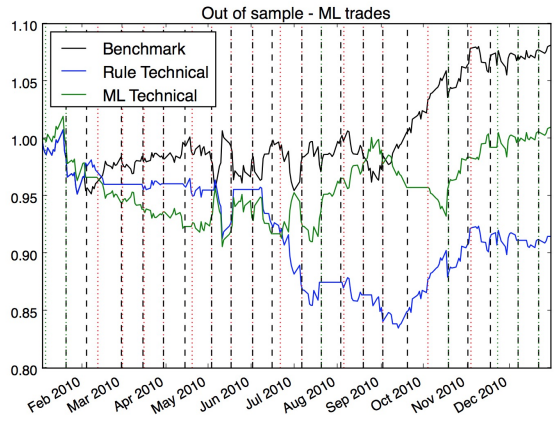

Figure 8: *Performance of my rule based and ml based strategies compared to the market for the out of sample period. Trades are displayed for the ml based strategy*

2) I'd only optimize my parameters on the in-sample period up to the point that it also increased my out of sample performance. Likely both my rule-based and my ML-based strategy were over-fit to the in-sample training data. Furthermore, the in-sample trading data has a period of serious stock decline related to the financial crisis. A longer in sample period with a diverse set of market conditions (booms, busts, bubbles, recessions, etc.) would potentially deliver a more robust algorithm. Finally, instead of optimizing for the return at the end of the in-sample period I could have optimized for the average over the whole in-sample period.