Making Predictions in Extremely Volatile Markets

Alex Contryman

alexwc@stanford.edu

Introduction

Algorithmic trading, which inherently relies on machine learning, has become increasingly prevalent in the stock market in recent years. One question regarding algorithmic trading is how volatility of the financial assets being traded affects algorithm performance. In general, we expect that learning algorithms will perform more poorly for assets with more volatile price history.

In this report, we study the performance of two learning algorithms for financial assets of varying volatility. The specific goal is to predict whether the next-day closing price of a particular stock will be higher or lower than the current closing price. This simple prediction enables a straightforward trading algorithm to make buy and sell decisions based on the next-day prediction. Both price prediction and trading algorithm performance are studied as a function of stock volatility.

Data

The daily opening, high, low, and closing prices along with trading volume of 28 randomly chosen stocks of varying volatility were acquired from Yahoo Finance for the period 1/1/2000 to 11/1/2014. The volatility for this period was calculated by first calculating the daily logarithmic return

$$R = \ln\left(\frac{close_{i+1}}{close_i}\right)$$

Where $close_i$ is the closing price for day *i*. The annual volatility is given by

$$\sigma = \frac{\sigma_{SD}}{\sqrt{\frac{1}{252}}}$$

Where σ_{SD} is the standard deviation of daily logarithmic returns and we assume 252 trading days in a year to convert to annualized volatility. The distribution of annualized volatilities of the chosen stocks are shown in Figure 1.

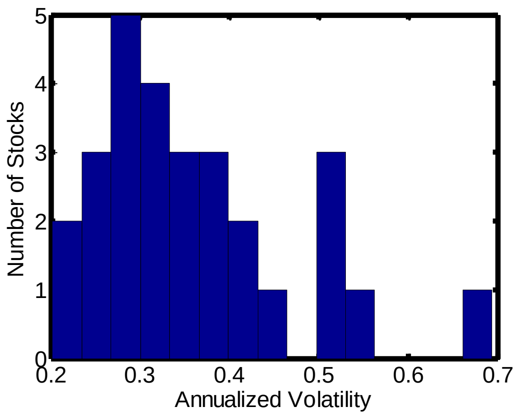

Figure 1. Distribution of volatilities of chosen stocks.

Features

Features were calculated using percent rate of change (PROC):

$$F = \frac{V_i - V_{i-d}}{V_{i-d}}$$

Feature *F* is the PROC of value *V* from its value *d* days ago. PROC features were calculated for the daily opening, high, low, and closing prices along with volume for $d = 3$, 10, 30, 100, and 300 days for a total of 25 features per individual stock. PROC was chosen for the features because it is independent of the underlying values which hopefully results in a more universal feature. Any feature which has price dependence might end up training algorithms on specific historical events that aren't necessarily relevant to the present (i.e. buy when the closing price hits $50/share).

Algorithms

Two algorithms were used to predict up/down closing price movement: logistic regression and support vector machine (SVM). The SVM uses L^1 regularization and a Gaussian kernel.

For training, the most recent two years of price data was excluded in order to save for final testing. From the remaining data, the algorithms were trained on the first 80% of the data, with the last 20% of the data used for cross-validation. Features were chosen using forward search with a maximum of 12 features. Figure 2 shows an example of how the percent correct ($1 - \text{Error}_{\text{cross-validation}}$) evolves as features are added during forward search.

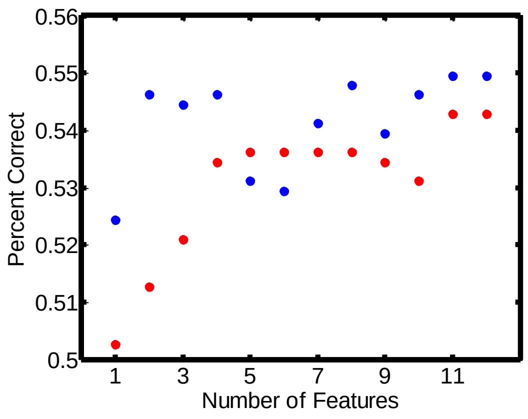

Figure 2. Evolution of cross-validation error during forward search as features are added for an example stock. Blue and red dots correspond to SVM and logistic regression, respectively. The y-axis is plotted as percent correct, which is defined as $1 - \text{Error}_{\text{cross-validation}}$.

This training algorithm was repeated for each of the 28 stocks chosen, for a total computational runtime of about 6 hours.

Results

I. Prediction

Figure 3 summarizes the prediction performance for all stocks. Blue and red correspond to SVM and logistic regression results. The overall performance between the two algorithms was quite close, but SVM does do slightly better overall. For the remainder of the analysis, we focus only on SVM.

It is significant that the algorithms can only make predictions 5-10% better than simply choosing at random. It is possible that better or more numerous features would improve their performance, but there will likely be some fundamental performance limit due to the at least partially random nature of the stock market.

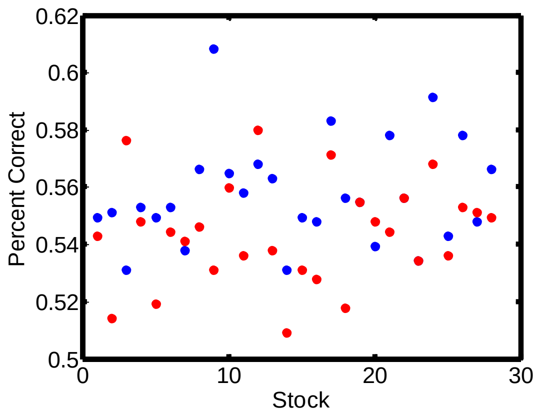

Figure 3. Performance of algorithms for all stocks. Blue and red markers correspond to SVM and logistic regression.

Figure 4 explores the relationship between algorithm performance and number of features used for each stock. Each blue dot represents the number of features that the forward search algorithm gave the best algorithm performance for a particular stock. The red trend line is a least-squares fit to the plotted data, which seems to weakly indicate that prediction performance is better when more features are used. This suggests that adding features and increasing the feature limit could improve algorithm performance.

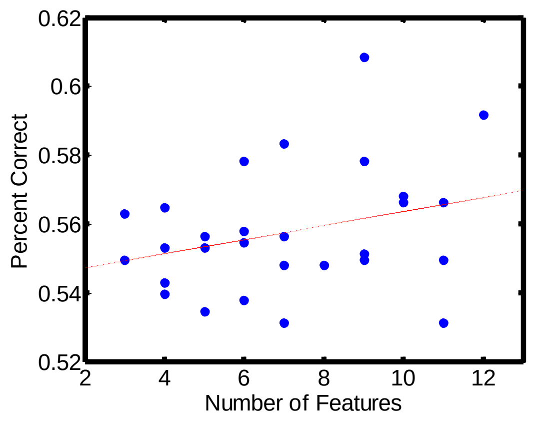

Figure 4. Algorithm performance as function of number of features used.

Figure 5 shows the relationship between algorithm performance and stock volatility. There seems to be no strong correlation between stock volatility and algorithm performance, which runs against what was expected.

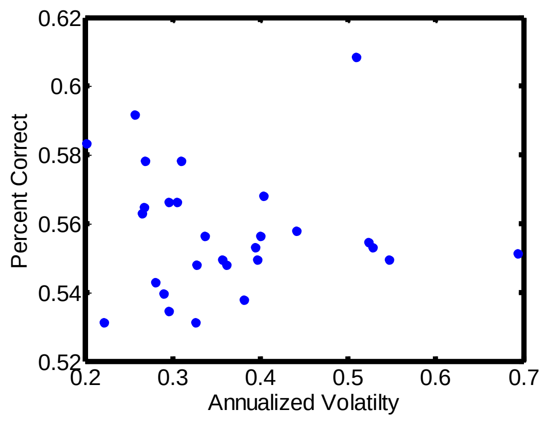

Figure 5. Algorithm performance versus volatility.

II. Simulated Trading

To further test the algorithm and get a sense of real world performance, the predictions were used to run an extremely simple trading algorithm using the final two years of data excluded from training. The trading algorithm consists of two rules: if the stock's next-day closing price is predicted to be higher than the present day closing price, buy as many shares as possible at the present day closing price. If the stock's next-day closing price is expected to be lower, sell all shares.

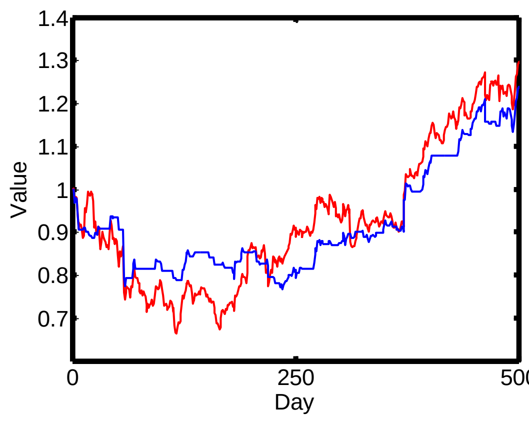

Figure 6. Simulated trading performance for a particular stock. The red line is the stock's closing price, normalized to its day 1 value. The blue line represents the trading algorithm's currently held value of cash and stock.

An example of the trading algorithm's performance for a single stock can be seen in Figure 6. The normalized stock price is plotted in red and the combined value of cash and stock held by the trading algorithm is plotted in blue. Overall the algorithm does slightly worse over this period than a simple buy and hold strategy; however its performance is less volatile than just the stock price. Decreased volatility is something valued by investors, and one way to quantify volatility-adjusted performance is the Sharpe ratio[1]:

$$S_a = \frac{E[R_a - R_b]}{\sigma_a}$$

R_a is the return of the asset, R_b is the risk-free return (which we set to zero), and σ_a is the standard deviation of the asset returns. The Sharpe ratio rewards large returns, but punishes huge swings in returns.

In Figure 7, the Sharpe ratio of the algorithm's performance for each stock normalized by the Sharpe ratio of the stock's price is plotted versus stock volatility. The trading algorithm manages to have a larger Sharpe ratio than the stock for every stock tested, which is a great performance result. Additionally, there seems to be a slight negative trend with volatility, suggesting that the algorithm performs worse for more volatile stocks.

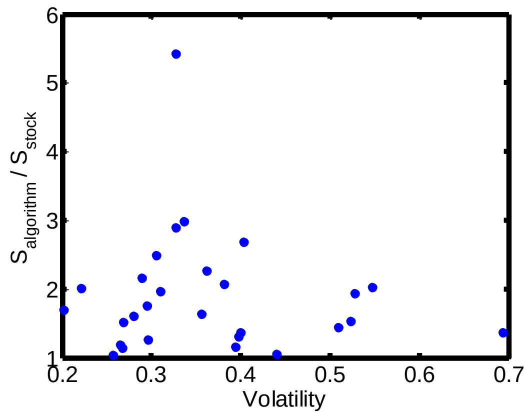

Figure 7. Normalized Sharpe ratio versus volatility. The Sharpe ratio of the trading algorithm normalized by the Sharpe ratio of the underlying stock is plotted.

Conclusions

SVM and logistic regression were able to predict daily up/down stock price swings with moderate success. Analyzing the results of the forward search feature selection algorithm, we found that using more features or allowing a larger number of features in the algorithms might improve performance. Using a simple simulated trading algorithm, we found that the algorithm did not outperform the simple buy and hold strategy in an upward trending market, but it did reduce volatility. By measuring risk-adjusted performance using the Sharpe ratio, the trading algorithm outperformed buy and hold for every stock tested.

Risk-adjusted algorithm performance seems to get slightly worse with higher volatility. One way to combat this trend would be to try adding features more sensitive to quick changes in price. In this study, only PROC feature were used which roughly corresponds to a first derivative of price. Features created with higher order derivatives could be used to try to achieve better high-volatility performance.

References

1. William F. Sharpe, "Mutual Fund Performance," Journal of Business, Vol. 39, No. 1, Part 2, January 1966, pp. 119-138.

Acknowledgements

Thanks to Joel Thompson who helped develop and refine the idea for this project.