Recognizing Informed Option Trading

Alex Bain, Prabal Tiwaree, Kari Okamoto

1 Abstract

While equity (stock) markets are generally efficient in discounting public information into stock prices, we believe that in option markets a certain class of informed trading exists which is based on private information that cannot be efficiently discounted into stock prices. This kind of trade, made either by insiders or large institutions with the resources to deduce non-public information about a stock, allows participants to make bets with a limited downside risk and enormous upside potential. We demonstrate the ability to recognize this class of trades using machine learning algorithms and the rich features available for option markets. We present a simple trading strategy that buys a portfolio of selected options and show that it generates outstanding returns.

2 Background

2.1 Equity Options Market

While many people are familiar with debt and equity markets, knowledge of option markets is far more limited. While there are several types of options markets, we will be concerned with the equity options market, which is simply a public market for stock options. The Chicago Board of Options Exchange (CBOE) is the largest and most important equity options market in the United States. The CBOE lists options on over 2,000 stocks. In general, most US equities with market caps of $500 million or more have publicly listed options available to trade.

Although stock market prediction is a classic machine learning problem, hardly any work has been done in the options market. Despite this, we believe that machine learning in the options market can be very fruitful. Because options are expiring derivatives with a theoretically optimal price, options have wonderful features upon which to apply machine learning. Option volumes for many stocks are generally low. The consequence of this is that unusual purchases in the options market have much stronger signals than in the noisy stock market.

2.2 Informed Option Trading

In this paper we focus on informed option trading. When an informed trader wants to make a big bet for a relatively small amount of money, they do it with options. In particular they might buy near term out-of-the-money options; that is, options that are about to expire (perhaps in the next few weeks) which require a large price move in a short amount of time if they are to not expire worthless. These options are inexpensive and the maximum downside risk is simply the purchase price. However, the upside reward can be phenomenal if the options expire in the money.

Note that informed trading does not necessarily mean insider trading. We are looking for large unusual trades that are likely to have been made by hedge funds or institutional money managers with the resources to draw (legal) conclusions about companies. Such trades are denoted informed trades in options parlance, and the managers running the funds are known as informed traders.

While human beings notice unusual option trades, it can be difficult to tell if such trades are indicative of an informed trader with private information about the stock. Indeed, while Cao [1] finds that options are a good predictor of material events, Pan [2] finds that it takes several weeks for stock prices to fully adjust to the information given by options volume. It is here that we will apply machine learning to help us recognize informed option trades that are highly likely to be profitable.

3 Approach

3.1 Stock Option Classification

For this problem, we consider publicly traded equity options. Since call options are more often used

for speculation than puts (which are often used for portfolio protection), we will focus exclusively on calls (although our analysis could be extended to puts). Each stock has a series of listed options that expire on a monthly basis. To limit our scope, we will focus on front month options, i.e. options that expire within the next 30 days.

Given the daily trading data for a particular front month option, we would like to pose the following classification problem:

"Will the underlying stock be trading at a price at expiration above the strike price plus the purchase price of the option?"

We could have also proposed a regression problem that estimates the value of options at expiration. However, since you generally stand to make a large profit by purchasing out-of-the-money options that expire in-the-money, we preferred the clarity of the classification problem.

3.2 Training and Testing Data

We gathered trading data from the Ivy DB OptionMetrics database, a comprehensive options database. Data for the entire year of 2006 was downloaded for the 796 stocks comprising the Russell Midcap Index. We chose to use this list of stocks because the average market capitalization of companies in this index is about $6 billion. Therefore, the options for stocks in this index are liquid, but have relatively low daily option volume.

A large Python framework was built to parse and assemble the data. The data was initially filtered to options expiring in 30 days or less that are at least 5% out-of-the-money. This filtered the data down to 64,801 records. Each record corresponds to trading data for one option on one day. At this point a second key filter was applied, according to these three criteria:

1. The volume for the option exceeds twice the daily average (unusual volume rule)
2. The volume of options traded is at least 500 (institutional-sized blocks rule)
3. The change in implied volatility for the option is positive (institutional buying rule)

The first rule finds options with unusual trading volume. The second makes it more likely that institutions or hedge funds rather than individuals are participating in the trades, since institutions generally buy large blocks of options. The third rule shows aggressive buying instead of hedging or closing positions. These filters condition the data to show only strong, unusual buying with institutional participation. For more intuition behind these rules, see our original project proposal.

These rules filtered our data into 341 candidate options. We split our data into 60% for training and 40% for testing. Each option was labeled as positive if its value at expiration exceeded the purchase price of the option. The training data consisted of 210 options with 22 positive examples, while the testing data had 131 options with 17 positive examples.

3.3 Features

The rich properties of stock options afford us a number of potential features to use for machine learning. With our Python framework, we built a base set of 13 features designed to measure the pattern of buying for the candidate option. We implemented feature selection using backwards search and dropped the first feature (see next section). For some learning algorithms, we normalized our data by rescaling features to [0, 1] or to a z-score. Our final features were:

1. How far out-of-the-money the option is as a % of the stock price
2. Price of the option (best closing offer)
3. Implied volatility of a standardized 30-day at-the-money option
4. Daily change in implied volatility
5. Volume / total call volume for the stock
6. Volume / one month average total call volume
7. Total call volume / total put volume
8. Average total call volume / average total put volume
9. Total call volume / average total call volume
10. Change in open interest from todays trades
11. Change in open interest / volume
12. Change in open interest / total open interest

Below is an example of what the features would be for the Vertex Pharmaceuticals (NASDAQ:VRTX) November 35 call on October 24, 2006 right before the stock exploded from $33 to $45. This option was identified by several of our models as likely to be profitable.

VRTX Nov 2006 $35 call on October 24, 2006

Percentage out-of-the-money	5%
Option price	$1.90
Implied volatility	0.72
Change in implied volatility	0.02
Volume / total call volume	0.39
Volume / one month avg vol	4.17
Total call volume / tot put vol	7.53
Average call vol / avg put vol	0.34
Total call vol / avg call vol	10.6
Change in open interest %	0.48
Change in open int / volume	0.55
Change open int / tot open int	0.13

Figure 1. Example Features for an Option

For this example, VRTX closed on October 24, 2006 at $33.32. Buyers came in and bought four times the average number of calls, paying up for them (implied volatility went up). The overall number of calls they bought at all near-term strikes was 10.6 times the average. VRTX reported earnings and discussed its new drug telaprevir two days later.

3.4 Feature Selection: Backwards Search

Remove:	None		1	
Remove Feature #	True +	False +	True +	False +
1	5	13	X	X
2	4	11	5	13
3	3	12	4	14
4	2	5	2	6
5	4	11	5	13
6	4	8	4	10
7	3	14	4	13
8	3	11	5	10
9	3	12	2	11
10	4	10	4	10
11	3	13	4	13
12	3	11	5	14
13	4	10	5	13
Remove:	1, 8		1, 8, 2	
Remove Feature #	True +	False +	True +	False +
1	X	X	X	X
2	5	10	X	X
3	4	11	4	11
4	1	1	1	1
5	5	10	5	10
6	3	9	3	9
7	4	12	4	12
8	X	X	X	X
9	4	8	4	7
10	3	9	4	10
11	3	9	3	9
12	5	11	5	11
13	5	10	5	10

Figure 2. Feature Selection Process with Backwards Search

In order to find an optimal subset of features to use, we implemented a feature selection algorithm using a backwards search. Starting with 13 features and using our Bayesian logistic regression model (see section 4), we found the baseline performance of our algorithm on a dataset. Running the same algorithm, removing one feature at a time to form a subset of 12 features, we found the optimal feature to remove to improve our performance. After removing this feature, we continued the process by removing one optimal feature at a time and decreasing the size of our feature subset by one until there was no longer any recognized improvement. Figure 2 shows the iterations of this algorithm until convergence. This optimal subset contains 10 of our original features. We can see from the last few iterations of the backwards search that the overall error will remain the same if several of the features are kept or discarded. This implies that these features most likely will not decrease the effectiveness of the learning algorithm, but will not improve it either.

3.5 Learning Algorithms

For the classification problem we evaluated several learning algorithms including SMO, SVM_light, logistic regression, and boosted decision trees. We replaced logistic regression with $l2$-regularized Bayesian logistic regression, which gave us much better results.

For the support vector machines, we found that a polynomial kernel clearly gave us the best results. We performed a grid search over the regularization parameter C and the kernel polynomial degree d using SMO to optimize the SVM parameters (see the results section).

We evaluated boosted decision trees using GentleAdaBoost. Boosting did not perform very well, but we include some of our results using Boosting for comparison.

Each training and testing example is a pair $(y^{(i)}, x^{(i)})$, where $y^{(i)}$ indicates whether the stock price was above the option strike price + the total cost of the option at expiration, and $x^{(i)}$ is a vector of our 12 option features.

4 Experimental Results

4.1 Parameter Optimization

We ran a simulation to find the optimal value of the degree 'd' for the polynomial kernel and of the error penalization parameter, 'C', for the SMO algorithm. The degrees of the polynomial chosen ranged from 0 to 50, and we found that in general the SMO ran faster for higher degree polynomial kernels and slower when C was increased. Also, SMO ran significantly faster on the normalized data compared to the raw data. The 'max-

passes' value of 1000 was found to be good enough to get consistent results and tolerance of 0.0001 was used. The following table summarizes the simulation result:

Degree (d)	C	Error	Positives	True Positives	False Positives	False Negatives
1	All	12.98%	17	0	0	17
2	0	12.98%	17	0	0	17
	0.3	12.98%	17	0	0	17
	0.5	12.98%	17	0	0	17
	1	12.98%	17	0	0	17
	2	12.98%	17	0	0	17
	5	12.21%	17	2	1	15
	10	13.74%	17	2	3	15
	20	18.32%	17	2	9	15
	100	17.56%	17	3	9	14
3	0	12.98%	17	0	0	17
	0.3	12.98%	17	0	0	17
	0.5	13.74%	17	0	1	17
	1	13.74%	17	2	3	15
	2	16.03%	17	2	6	15
	5	16.79%	17	3	8	14
	10	17.56%	17	3	9	14
	20	20.61%	17	3	13	14
	100	18.32%	17	3	10	14
15	0	12.98%	17	0	0	17
	0.3	19.85%	17	4	13	13
	0.5	19.85%	17	4	13	13
	1	**22.90%**	**17**	**5**	**18**	**12**
	2	25.19%	17	3	19	14
	5	26.72%	17	3	21	14
	10	26.72%	17	3	21	14
	20	26.72%	17	3	21	14
	100	26.72%	17	3	21	14
30	0	12.98%	17	0	0	17
	Rest	32.06%	17	5	30	12

Figure 3. Grid Search for Optimal C and d

The best d and C parameters were found to be 15 and 1 respectively based on the accuracy, number of true positives, and number of false positives. It is important for the algorithm to have as many true positives as possible since it would allow us to profit from these trades while minimizing false positives as these trades would lose money. Figure 4 shows the number of true positives and false positives using a polynomial kernel of degree 15 for various values of C.

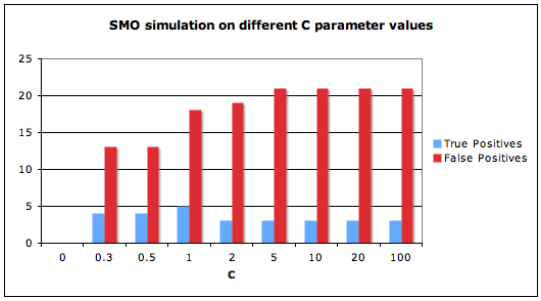

Figure 4. Search for Optimal SVM Polynomial Kernel

Figure 5 shows the accuracy, precision, and recall for different values of C, while degree is held constant at d = 15. From this figure, we can clearly see that precision and recall are both maximized around C = 1, while accuracy is still fairly good.

Likewise, Figure 6 shows the corresponding accuracy, precision, and recall for various values of degree, with constant C = 1. From this graph, it can be seen that a degree around 15 minimizes the tradeoff of low precision and low recall. Overall, it averages out these ratios, and therefore provides a decent model.

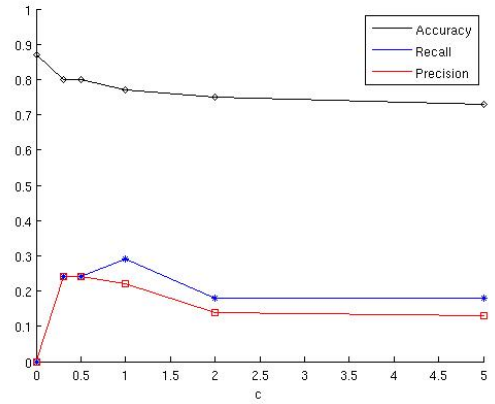

Figure 5. Precision, Recall, Accuracy for SVM parameter C

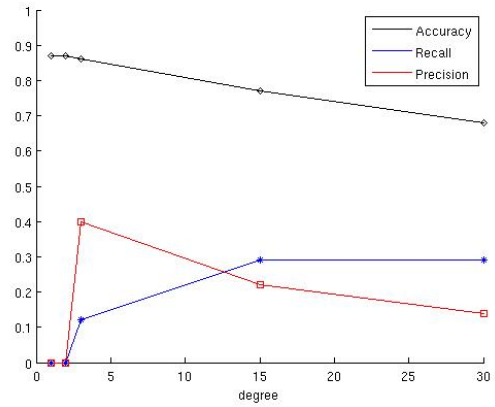

Figure 6. Precision, Recall, Accuracy for Kernel degree d

We ran a similar simulation for SVM_light and Bayesian logistic regression to obtain the best 'd' and 'C' parameters. For SVM_light, we also used 'j' parameter which is a ratio of cost on false positives to false negatives.

4.2 Testing Results

After coming up with the optimal parameters, we ran the learning algorithms and obtained the outputs. In the optimal case, the learning algorithms obtained five true positives - albeit not all the same - and around 15 false positives.

Model	Training			Testing		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
SMO	99.05%	95.45%	95.45%	82.11%	21.74%	55.56%
SVM_LIGHT	91.43%	55.56%	90.91%	83.08%	31.25%	31.25%
BAYESIAN LR	90.00%	53.33%	36.36%	82.81%	27.78%	35.71%
BOOSTING	100.00%	100.00%	100.00%	82.68%	15.38%	35.71%

Figure 7. Training and Testing Precision, Recall, Accuracy

4.3 Trading Strategy

We then came up with two different investment scenarios. In the unweighted case, we invest an equal amount of money in each of the output recommendations from each learning algorithm. In the weighted case, we add more weight to the outputs that different learning algorithms all recommend and less to the ones that are unique to each of our learning algorithms. For instance, if SVM_light, SMO, and Bayesian logistic regression all recommended buying option 'A', then we assigned 3 times more weight to it than to an option recommended by only one algorithm. We then calculated the returns which are shown in the table below. In the weighted case, the options recommended by Bayesian logistic regression had the most returns, while the SVM_light recommendation performed the best for the unweighted case. Some of the weighted returns more than double the initial investment. Investing in random options produced negative returns.

Model	Unweighted	Weighted
SMO	18.04%	46.31%
SVM_LIGHT	88.48%	121.57%
BAYESIAN LR	67.39%	136.30%
RANDOM (19)	-45.97%	

Figure 8. Returns from our Options Trading Strategy

5 Summary

The goal of this project was to detect informed option trades using machine learning techniques. We used four different learning algorithms - SMO, SVM_light, Bayesian logistic regression, and Boosting - to learn from 796 stocks worth of data from 2006. We wrote scripts in Python to process large amounts of data obtained from the database and filtered them to options showing strong unusual buying. We normalized the data, and ran various simulations to obtain the optimal set of parameters for different learning algorithms. Our algorithms learned from the data and detected many trades that were profitable. As discussed in the results section, investing in the options recommended by our algorithms generated significant returns. To be more confident on the ability of our algorithms to learn, we need to test it on larger and varied test sets over longer time periods to see if the positive returns can be generated consistently. We will also be looking to find the best set of features and analyze the training set to improve on the classification error.

6 Future Work

There are many ideas that this project could have employed, some of which are:

1. Our work used information from the past trading days by looking into 20 trading day moving-averages, but we could employ more a sophisticated time-series analysis to create better features.
2. We used end-of-day summary data which, for instance, did not look into the number and size of trades during the day. Because of this, we might have missed crucial insights on whether the volume was created by one large buyer or a few small buyers.
3. We used backward search for feature selection, but could experiment with other feature selection algorithm.
4. We could use the characteristics of the underlying stock as our features to link the option price movement to that of its underlying asset.
5. We could explore the data on put volume and the downward movement of the underlying stock to detect if information was leaked prior to bad news (stock decline).

7 Related Work

A large number of machine learning techniques have been applied to the stock market. We mention some particularly relevant papers here. Our paper is in the spirit of Choudhry and Garg [3] but using options features instead of technical analysis. The use of SVMs specifically for financial time series forecasting was studied by Tay and Cao [4].

A fantastic paper on detecting insider trading using machine learning techniques was given by Donoho [5]. Some of our features are based on his ideas, although we are more focused on detecting institutional buying while he focuses on insider trading.

A small number of recent papers have focused on the options market. Recently, Audrino and Colangelo [6] used regression trees to predict implied volatility changes and demonstrated a successful trading strategy with their results.

Interestingly, informed trading has been extensively studied by game theorists, with many non-intuitive results. The pioneering work was done by Kyle [7].

8 References

[1] Cao, Chen, and Griffen. Informational Content of Option Volume Prior to Takeovers. Journal of Business, 78 2005, pp. 1073-1109.

[2] Pan and Poteshman. The Information in Option Volume for Stock Prices. Review of Financial Studies, 19 2006, pp. 871-908.

[3] Choudhry and Garg. A Hybrid Machine Learning System for Stock Market Forecasting. World Academy of Science, Engineering, and Technology, 39 2008, pp. 315-318.

[4] Cao and Tay. Financial forecasting using support vector machines. Neural Computing Applications, 10 2001, pp. 184-192.

[5] Donoho. Early Detection of Insider Trading in Options Markets, 2004.

[6] Audrino and Colangelo. Option trading strategies based on semiparametric implied volatility surface prediction. August 2009 Discussion Paper No. 2009-24.

[7] Kyle. Continuous auctions and insider trading. Econometrica, 53 1985, pp. 1315-1335.