Exploring Techniques in Surprised Learning

Kam Wing Chung (Kenneth)

Department of Computer Science

Georgia Tech University

Kchung42@gatech.edu

Purpose

The purpose of this assignment is to explore decision trees, neural networks, boosting, support vector machines and k-nearest neighbor in supervised learning. All of above supervising learning techniques will be based on Wine Quality Data Set found on (<http://archive.ics.uci.edu/ml/datasets/Wine+Quality>). All of above supervising learning techniques will be using R and R packages. R code, dataset and README.txt can be found the same package as this document.

1 Wine Quality Data Set

One of the major focuses of research in finance is the pricing of derivatives. There are many existing models in finance for predicting the price of an option, most of which revolve around the Black-Scholes model; however, these models tend to involve highly complex mathematics and often make many assumptions about the underlying characteristics of the market.

An option is a type of derivative which gives the owner the right, but not the obligation, to buy or a sell a stock at a specified price (called the strike price) and at a specified future date. A stock option that gives the right to buy a stock at a certain date is called a “call option”. A stock option that gives the right to sell is called a “put option”. There are different styles of options available to investors. Two of the most common are European and American style options. Owners of European style options can only execute their options on the specified expiration date. American style options, on the other hand, allow execution at any time up until the expiration date.

Suppose for example that a certain stock, call it S, is trading at $100 a share. If we as investors think that the price will increase we could buy a call option with a strike price of $105 that expires in one month. This would give us the right, but not the obligation, to purchase stock S in one month’s time for $105 a share. If the stock price in one month’s time is greater than $105, let us assume it is selling for $110, we can now buy it for $105 and then turn around and sell it for a $5 profit. On the other hand, if the stock price in one month is less than $105 we simply choose not to exercise our option and only lose the amount we initially paid for the option contract. Finding the true value of an option and what its fair price continues to be an active area of research.

The Black-Scholes model of option pricing is based on the solution to a set of partial differential equations. The inputs to the model are the price of the stock, the strike price, the risk-free interest rate, the volatility, and the time to expiration. The output is the theoretical value of the option. In the derivation of the Black-Scholes option pricing model several assumptions are made [Black *et al*., 1973]:

* The underlying security does not pay a dividend
* The stock follows a geometric Brownian motion with constant drift and volatility
* All securities are infinitely divisible
* There are no restrictions on short selling
* There are no transaction costs or taxes

While these assumptions are useful in developing a rigorous closed form solution, many of the assumptions are unrealistic. Most stocks pay dividends and it is usually impossible to buy half of a share of a stock. Brokerages typically put limits on short selling and of course charge transaction fees for using their services to buy and sell options and stocks. Studies have also shown that a fractal description of the market may be a better model than Brownian motion [Peters 1994]. Akgiray [1989] has also shown that stocks typically do not have constant variance.

Because of these discrepancies found in the theoretical Black-Scholes model, we propose the use of Machine Learning techniques in order to develop a non-parametric option pricing model. The reason for this is that Machine Learning techniques do not make any implicit assumptions about the relationships between input variables. Using an Artificial Neural Network, we are able to let the learner discover relationships that may not be included in standard models like Black-Scholes. In the following sections we describe the data used in developing this model as well as additional steps taken to improve accuracy. We test using several different strike prices and different expiration times. We also investigate whether calls or puts are easier to price. We analyze the profit potential of each option and evaluate an option-buying strategy based on machine learning.

2 Procedure

2.1 How Data was Obtained

We obtained our data from the Bloomberg Terminal located in the Harold B. Lee library at BYU. Using this service we were able to obtain historical data for the following attributes:

* Strike Price (X)
* Underlying Stock Price (S)
* 10-day Historical Volatility
* 30-day Historical Volatility
* Days until Expiration (T)
* The Market Price of the Option (P)
* The 10 previous days of stock prices
* Risk-free rate (US 10-year T-Bill)
* Expiration Price of Stock (E)

Because the risk-free interest rate is purely theoretical we estimated this using the returns on a 10-year US Government T-Bill. Using the option price and the price of the stock at expiration we obtained the profit potential of each option using the following formula for a call and put option respectively:

*Call Profit = max(-P, E-X-P)*

*Put Profit = max(-P,X-E-P)*

We also created a Boolean variable *Money* that is a 1 if the profit is nonnegative and is 0 if the profit is negative.

2.2 How Tests Were Run

The tests were run using the Weka open source toolkit. The initial machine learning model used was the Multilayer Perceptron. We also used the ZeroR learner which provides a baseline accuracy by outputting the average value of the output class.

3 Experiments and Results

3.1 Initial Results on a Small Dataset

We began predicting option prices using a very simple dataset containing only these three attributes:

* strike price
* days until expiration of option
* and current stock price

An initial attempt, named the spy dataset, had 212 instances of the data, for 1 stock (SPY), at 4 similar strike prices ($131-$134). This was clearly not a very large sample, but the results of machine learning were still very good.

In order to quantify “good” results, we first ran a baseline learner on the spy dataset (the zeroR in WEKA) on 10-fold cross validation. It had a root mean squared error of 1.0645. (A typical result would be that it would predict the option’s price to be 1.97 when it was actually .86).

Upon trying the multilayered perceptron, the increase in accuracy was substantial. Using merely 1 hidden layer with 2 nodes, it produced a root mean squared error of .17, 15% of the baseline learner’s error. Using 2 hidden layers, each with 3 nodes, it had a root mean squared error of .127, 12% of the baseline learner’s error (A typical result would be to predict a price of $3.14 when the actual price was $3.04.)

So far these results were very promising, and calculating the option’s price appeared almost too easy. The fact that the learner was able to reduce the root mean squared error from 1.065 to .127 was very significant. Not only did this show that option prices are quite easily determined by just these three attributes, it also shows they are quite easily linearly separable.

However, these are results still only for 1 stock and 4 strike prices. In the next section we explore the use of a larger dataset with a wider variety of strike prices. While it is good to predict the option’s price, it gives no indication as to whether one should pay that money for the option or not. We will address this question later in the paper.

3.2 Put vs. Call Option Pricing

The following experiments investigate whether there is a difference between pricing put and call options and whether a neural network can learn to price both. We use a regression neural network where the output classification is a real value representing the price of the option. The following experiments use 5 different data sets to explore the impact of different features on the prediction error. The input data sets are:

1. Strike price, time to expiration, stock price
2. Everything in 1 plus 10-day historical volatility
3. Everything in 1 plus 30-day historical volatility
4. Everything in 1 plus the last 10 days of stock prices
5. Everything in 1 plus the risk-free rate
6. Strike price, time to expiration, stock price, 10-day historical volatility, 30-day historical volatility, last 10 days of stock prices, risk free rate.

Tests were done using a Multilayer Perceptron learner with one hidden layer. The number of nodes in the hidden layer is equal to the (number of attributes + number of classes)/2 . Testing is done on the call and put data separately and then these data sets are merged and a new nominal value is added denoting whether an option is a call or a put.

The first experiment used option data for the SPY US Equity which expired on January 22, 2011. The data sets included a range of strike prices both above and below the ending value of the stock. There were 1,302 instances in the call data set and there were 973 instances in the put data set. We used 10-fold cross validation and reported the mean absolute error.

*Options Expiring on January 22, 2011*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 |
| Call | **.2259** | **.1867** | **.1937** | **.1094** | **.9104** | **.1026** |
| Put | **.404** | **.448** | **.4178** | **.2492** | **.4085** | **.4616** |
| Both | **.4167** | **.2944** | **.2976** | **.1977** | **.2509** | **.1794** |

The results showed that when the risk free rate was added to the data set of experiment 1, the error actually increased. We tried removing the risk free rate in experiment 6, but it increased the error. This implied that somehow the risk-free rate is valuable if other features are present, but actually decreases the predictive power when included by itself. The best accuracy (highlighted in yellow) was achieved using all of the features for the call options; however, the put options did best when only the last 10 days of stock prices were included.

Another surprising result was that the error for experiments 2 and 3 had higher mean absolute error than when using the past 10 days of data. This is strong evidence that the neural network learned a measure of volatility that is in some way more predictive than just the historical volatility. There were also many missing values for volatility and this may have also contributed to the larger errors. Finally we ran the machine learning model using both call and put data in the same arff file. A nominal attribute was added to denote whether the option was a put or call and the learner was able to do surprisingly well. Its accuracy was actually better in most cases than just the puts alone which shows that there may be value training with both call and put options together.

Using 1659 instances for call options and 1184 instances for put options that expired on March 19, 2011 we get the following results using 10-fold cross validation:

*Options Expiring on March 19, 2011*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | *2* | 3 | 4 | 5 | 6 |
| Call | **.7856** | **.6809** | **.2427** | **.2068** | **.2412** | **.195** |
| Put | **.246** | **.2506** | **.2615** | **.2211** | **.2641** | **.1223** |
| Both | **.4393** | **.278** | **.2667** | **.2012** | **.2686** | **.1809** |

The results were very similar: experiment 6 did the best and experiment 4 was better than either 2 or 3. Interestingly, the run with both puts and calls actually did a little better than just the call run. Once again this shows that pricing call and put options were similar enough for a neural network to learn.

These results show the benefit of adding features to the basic original model. It also shows the learner is able to develop its own measure of volatility simply by using the past stock prices.

3.2 Profit Estimates on Options

**3.2.1 Motivation for Profit Estimates on Options**

So far, we have shown machine learning is very accurate in predicting an option’s price using other attributes, but we have not yet shown whether it can actually help in investing.

Is it possible to estimate profit on an option using purely historical data? This seems to require a fair degree of predicting the future, but we were able to find a good strategy.

The attributes for this experiment were:

* Option\_Price
* 10\_day\_Volatility
* 30\_day\_Volatility
* Strike Price
* Risk-Free\_Rate
* Days\_until\_Expiry
* Stock\_Price
* 1\_day\_lag\_price
* 2\_day\_Lag\_Stock\_Price
* 3\_day\_Lag\_Stock\_Price
* 4\_day\_Lag\_Stock\_Price
* 5\_day\_Lag\_Stock\_Price
* 6\_day\_Lag\_Stock\_Price
* 7\_day\_Lag\_Stock\_Price
* 8\_day\_Lag\_Stock\_Price
* 9\_day\_Lag\_Stock\_Price
* 10\_day\_Lag\_Stock\_Price
* profit
* Option\_Type {"Call" or "Put"}

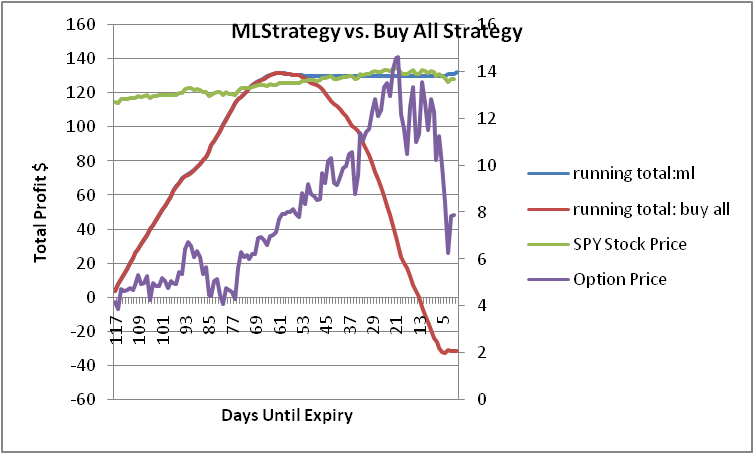
As shown, when predicting profit, option price was used as an attribute. An obvious difficulty was in trying to predict the future based on the past.

**3.2.2 Initial Learning of Profit Estimates**

The first test was run on a dataset of just under 1700 instances of call options, having a range of strike prices from 120-134, all expiring on March 18th 2011. The dataset was split into 66% training and 34% testing. These were the results:

|  |  |  |
| --- | --- | --- |
|  | Baseline Learner | Multilayered Perceptron |
| MSE | 11.39 | .170 |

(The learner found the attributes of option price, strike price, and the ensemble of stock prices with their lagging counterparts to be the most important. Sometimes risk-free investment rate was also considered important.)



Though prediction of profit is very easy in this problem, it requires the learner to be able to “look into the future.” Each instance contained information from different time periods: the profit is only known with certainty upon expiration of the option, but of course we want to know at time of purchase!

In order to provide a more realistic learning environment, all of the training set must be from a time period prior to the training set.

**3.2.3 Learning on Current, Testing on Future**

Two data sets were made from options expiring in January 2011 and March 2011, respectively. The January set was

Table 1 Train on January 2011, Test on March 2011

used for training, and the March set for testing. This provided for a much more realistic learning situation: if it were currently February 2011, we could train the learner on data from January 2011, and then use it to estimate profits on options expiring in March.

|  |  |  |
| --- | --- | --- |
|  | Baseline Learner | Multilayered Perceptron |
| MSE | 3.566 | 0.4604 |

Table 2 Train on November 2008, Test on January 2011

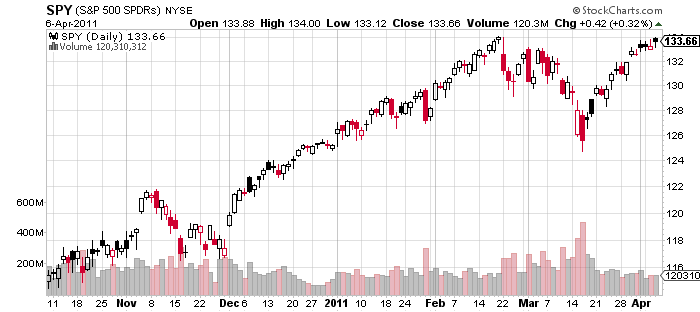


Figure 1 SPY From October 2011 to April 2011

This proved much more difficult for the learner, as shown in table “Train on January 2011, Test on march 2011,” nevertheless, this level of accuracy would still be beneficial to investors, as will be shown in the following section.

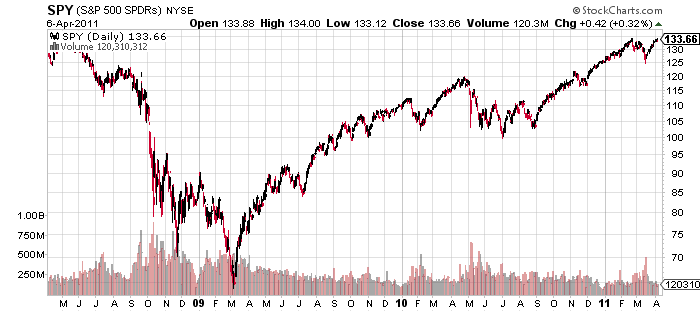
As shown in the chart of stock prices of SPY from November 2010 to April 2011, the stock prices were fairly similar between January and March. Thus the learner was training on data which was actually fairly similar to data it would be test on. If the situations are different, the learner has a much more difficult time.

Figure 2 Total Profits of ML Strategy in March 2011

Another test was ran, where the learner was trained on options expiring in November 2008, a period often considered “bearish,” meaning stock prices were generally decreasing significantly. It was then tested on data from January 2011. The table “Train on November 2008, Test on January 2011” shows the results for this experiment.

As shown, the perceptron actually performed significantly worse than the baseline learner in this case. This is because it had overfit to a bear market, and was thus very poor at predicting option profits.

|  |  |  |
| --- | --- | --- |
|  | Baseline Learner | Multilayered Perceptron |
| MSE | 18.8909 | 44.0632 |

This proves the point that clearly if the learner is shown only negative (or conversely, only very positive data) it will learn a very inaccurate view of the world.

3.3 Using the Machine Learning Options Strategy

Now that we have a learner that is able to train on historical option data and then make predictions about current options data, it would be nice to apply it to improve investment decisions.

An investment strategy was developed, here called the “machine learner strategy” (or ML strategy for short). The strategy is simply this: when the machine learner predicts that an option will produce more profit than the cost of the option, buy one of them. If it is not, do not buy it. In comparison, a strategy of “buy all” is compared.

**3.3.1 Applying the ML Strategy**

Figure 3 SPY from May 2008 to April 2011

The ML Strategy was used in March 2011, having been trained on the data from January 2011. In this case, accuracy is not the objective: profit is. The figure showing “Total Profits of ML Strategy in March 2011” shows how the strategy fared in comparison to the “buy all” strategy. Also shown are the stock’s price and option prices.

As seen, the two strategies performed identically for the first half of the date-range. This indicated that the ML strategy was also buying all the options it could.

However, at about halfway through the date-range, it stopped buying options, while the other strategy continued to buy. And it was a good thing the ML Strategy stopped buying: nearly all the options from that point on were losers. Thus, the “buy all” strategy lost nearly all of the gains it had made, while the ML Strategy successfully retained them.\

The stock price and option price are shown on the same graph (the option price corresponding to the right axis, also in dollars). These show that there was no clear indication that the options were too expensive, but the machine learner was still able to determine these facts.

**3.3.2 The ML Strategy Trained Poorly, but Still Performs Adequately**

What happens to the strategy now when it is trained on skewed data? For example, what were the results when it was trained during the bearish month of November 2008, then applied in January 2011? While November 2008 was terrible for call options, January 2011 was actually quite good.

The “buy-all” strategy performed very well, while the ML Strategy actually bought absolutely no options. In this worst-case scenario where it was very poorly trained, the ML strategy would have lost out on the profits gained from the simple “buy-all” strategy, but at least it doesn’t lose anything.

**3.2.3 ML Strategy Applied to Put Options w/ Larger Data Set**

We wanted to see how accurate we could get in predicting

profit on put options. In order to setup the tests, we obtained data for two sets: a training set and a test set. The training set consisted of 973 instances. These instances were composed of all put options from 8/26/10 to 01/22/11 with a strike price from 120 to 134 for ticker symbol SPY. Similarly, the testing set consisted of 522 instances. These represented all put options from 1/23/11 to 3/19/11 with a strike price from 120 to 134 for ticker symbol SPY. The same 19 attributes from the call options tests.

Our initial test for the put options model used a Multilayer Perceptron with one hidden layer, no validation set, 19 hidden nodes, learning rate of 0.3, and a momentum of 0.2.

The mean absolute error after running the data on this setup was 0.1647. This is a decrease of more than 2031% (3.346) off the mean absolute error on the same tests from the ZeroR learning algorithm. After tweaking some of the options for the Multilayer Perceptron, we found that by decreasing the learning rate to 0.1 and increasing the momentum to 0.4, we could further decrease the mean absolute error down to 0.1026. Changing the number of hidden layers, hidden nodes, or validation set size caused this number to increase.

After we found the best performing Multilayer Perceptron, we calculated the actual profits gained assuming we were purchasing put options contracts every time our machine learner predicted a profit. How much did we make using this model? Absolutely nothing! Our model predicted that every put option contract for SPY issued from 1/23/11 to 3/19/11 would yield in a loss on our investment. After comparing these results to the actual amount earned, or lost, we found that this prediction is correct.

Figure 4 Total Profits of ML Strategy in March 2011

This prediction can be explained using the first chart on page 4. This chart shows the stock price for SPY from 10/08/10 to 3/19/11. We are only interested in the price from 1/23/11 to 3/19/11. The trend from 1/23/11 to 2/10/11 is bullish—not what we want for put options. Remember that put option contracts are purchased with the hope that the price of the stock will go down. If it goes up, you lose money. That’s what happened during this period. So, we definitely would not have wanted to purchase a put option during this time.

Then, from 2/10/11 to 3/19/11, the price of the option takes a fall. However, the important thing to note here is that although the price of the stock decreased, it did not decrease enough to cover the costs of the option contract. Put a different way, the money earned on the decrease of the stock did not exceed the price at which we purchased the option. In many cases this difference was very small, in fact within just a few pennies, but our machine learning algorithm was able to detect that.

Testing and training on these put options also seemed to perform very well. Although we made no money, our model was very useful as a warning system to prevent us from making investment decisions that would cause us to lose money, thus allowing us to put our money elsewhere so that we can cause it to grow.

Conclusions

Initial results showed that using Machine Learning to price options results in models with significant gains over the baseline learner. We show that accuracy can be further improved by adding measures of volatility, previous stock prices. Adding a risk-free rate by itself did not improve accuracy, but it did provide predictive power when combined with the other attributes. We showed that using the last 10 days of stock prices achieves better accuracy than just using 10-day or 30-day historical volatility estimates. Also results show that it is possible to train a neural network to price both call and put options.

Options profits estimates were accurate given training data which was representative of the test data (i.e., training on data from a bull market and testing on a bull market). However, if the model was trained on data non-representative of the test set data, the model would predict expectedly unrealistic results.

Using the options profit estimates led to an effective option investment strategy, depending on the accuracy of the learner.

Future areas of research include doing a more thorough analysis of the impact of different attributes on error. Additional attributes could also improve accuracy. Another area that would benefit from future research is to generalize the Machine Learner to price options from multiple stocks and to use data spanning several years. Rather than using the zeroR baseline learner used in Weka, using the Black-Scholes model as a base learner would allow us to more accurately compare the predictive power of our models.

References

[Akgiray 1989] Akiray V. Conditional heteroskedasticity in time series of stock returns: evidence and forecasts. *Journal of Business*. (61):55-80, 1989.

[Black *et al.*, 1973] Black, Fischer, Myron Scholes. The Pricing of Options and Corporate Liabilites. *Journal of Political Economy* 81 (3):637-654, 1973.

[Peters 1994] Peters EE. Fractal market analysis: applying chaos theory to investment and economics. John Wiley & Sons, 1994.

[Samur and Temur, 2009] Zeynep Iltuzer Samur, Gul Tekin Temur. The Use of Artficial Neural Network in Option Pricing: The Case of S&P 100 Index Options. *World Academy of Science, Engineering and Technology* 54, 2009.

[Yao *et al.,* 2000] Jingtao Yao, Yili Li, Chew Lim Tan. Option price forecasting using neural networks. *The International Journal of Management Science*. Omega 28 455-466, 2000.