# Automated Stock Trading through Q Learning

Joe Arriaga

CS 445: Machine Learning
Winter 2019

## Introduction

Those involved in stock trading have been interested in the advantages that computing could provide them ever since computers entered the public sphere. As the world has become more reliant on computers so has the infrastructure of the stock market. Much of the trading is now done programmatically where a single system is capable of executing thousands of trades per second making a profit of only fractions of a penny, but the extreme frequency and volume of trades is able to create a profit. Just as computers were used to trade faster than human traders it is now commonplace for financial firms to use "AI traders" to trade smarter, taking advantage of patterns or trends that humans may miss.

The purpose of this project was to investigate whether a simple machine learning model could be constructed that would be profitable trading stocks. The typical model used for this type of problem is a linear regression model which identifies the trend of a stock's price and predicts its future price. This model works well for a single stock but if one wants to monitor many stocks or the entire market one needs a separate model to track each stock, which was too computationally intensive. Instead, Q learning was chosen with the hope that general patterns could be identified which applied to all stocks. This would allow a successful model to be learned with much lower computational cost, although this simpler Q learning model might not be as profitable as having a linear regression model for each stock.

## Methods

The objective of learning to make a profit trading stocks can be approached in many ways. Depending on how one views the problem different models will appear more appropriate and, conversely, one could choose one of many different models first and find a suitable way to formulate the problem to fit the chosen model. In this case the problem was viewed as a task to be learned that consisted of discrete actions in a relatively well-defined environment so the reinforcement learning model Q Learning was selected.

## Theory

Much work was done to simplify the view of the problem such that it would conform neatly to the Q Learning algorithm. The possible actions suggested themselves easily since there are only three basic actions a stock trader can take: buy, sell, and hold. Determining the states that would make up the environment was somewhat more difficult as the stock market in the real world does not have discrete states and the price of a stock is also continuous rather than discrete. The solution was to simplify the view and declare that from one day to the next the price of a stock can only do three things: increase, decrease, or remain the same. As an aside, it is exceedingly rare for a stock price to remain exactly the same but it is necessary to consider for the sake of completeness.

With this set of three basic states (increase, decrease, no change) one could attempt to find patterns and predict the movement of stock prices but with such little information it would be unlikely to perform well. However, one can capture more information for the algorithm by defining a state to be a sequence or tuple of the basic movements from consecutive days. By looking at the movements of the stock price over a number of days one is more likely to identify trends, and this is what humans tend to do as well. The question then becomes: How many days should be considered? or, How long should the sequences be?

Of course, with a longer sequence the model has more information and is more able to identify patterns and trends. However, there is a trade-off of increasing complexity and decreasing the ability of the model to effectively learn, which is similar to over-fitting. Reinforcement learning relies on encountering the same states over and over, trying different actions and gradually determining which action is best for each possible state. If there are too many states a given state may only be encountered once or twice, providing very little opportunity to learn. The first time a given state is encountered the model must simply make a random guess as to which action to take because it has no prior experience, each subsequent time the model encounters that state it can try a different action and see if the results are better or try the same action and see if a similar outcome is produced, both options contribute to determining the best action and increase the confidence in that determination.

For example, suppose one is attempting to learn to trade stocks and one chooses to use sequences 365 days long with the choice of one of the three basic states (increase, decrease, no change) in each position.

Then there are $3^{365} \approx 1.41 \times 10^{174}$ possible states so even over the course of many years it is unlikely that the same sequence will be encountered more than once, meaning that most of the actions taken are random guesses, and even if a sequence is encountered again a sample size of two is not large enough to confidently say which is the best action.

## Application

So the best sequence length is long enough to provide the information needed to recognize patterns and trends but short enough to be encountered many times and determine the best action to take, in general. Since the number of possible sequences grows exponentially with its length a relatively short sequence is preferred. In this case sequences of length 5 were chosen, yielding $3^5 = 243$ possible states. This is long enough for short-term trends to be identified but short enough that the total number of possible states is not too large and it is likely that states will be encountered many times, enough to obtain a high confidence about the best action to take.

The data set used was a slightly modified version of Dominik Gawlik's data set of S&P 500 companies from the New York Stock Exchange from 2010 through 2016 [1]. The original data set contained records for the stock of each company in the S&P 500 for each day of trading on the New York Stock Exchange. Because the value of companies can change, and because of business actions such as mergers, aquisitions, and the like, over the course of the data set some companies entered the S&P 500 and other left. The records for these companies were culled from the data set in order to leave a consistent set of stocks which are present throughout the entire data set. It is noted in the description that this data set does not adjust prices due to the effects of stock splits. The final data set used had prices for 467 stocks over 1,762 days. The fields for the date and stock symbol were used to organize the data but the only field used for computation was the opening price of each stock for each day.

### Algorithm Implementation

The algorithm implemented considered the changes of each stock price over the course of six days (eg. days 1 through 6), yielding a sequence of five changes. This sequence comprised the current state and would be looked up in the reward potential table. Each state had an associated reward potential which is the predicted movement of the stock price for the day following that particular state; either increasing, decreasing, or remaining constant. This predicted movement determines the action to take: if the predicted movement is positive the price is expected to increase so one should buy, if the predicted movement is negative one should sell, if the predicted movement is to remain the same one should simply hold the stock. In an attempt to maximize profits and minimize losses the "buy" action bought 10 shares of the stock while the "sell" action sold all shares currently owned.

The price change for the following day is then calculated and the result is incorporated into the price's predicted movement for that state. To simplify the algorithm the reward potential was simply increased or decreased by 0.5 in the same direction as the movement of the stock price. So, if the stock price increased one should have bought the stock and the reward potential value was increased to strengthen the suggestion towards this action. Over time the most common action determined the sign of the value while the magnitude indicated the confidence of that action. For example, a value of -1,230 is a strong indicator to sell, while a value of 200 is a fairly weak indicator to buy.

This process was carried out for each of the 467 stocks and all learning was combined in the reward potential table in an attempt to learn general trends that would be able to generalize.

The program then proceeded to the next day, calculated the newest change in price, updated the sequence (eg. now considering days 2 through 7), and repeated the process considering the new sequence.

## Evaluating Performance

It is not entirely obvious how to measure performance of trading stocks. There are many factors that influence the movement of a stock's price and the market as a whole and one must consider if each of these factors is important in evaluating the final outcome and how much significance to attribute to each. In many ways, the entire pursuit and occupation of trading stocks is an attempt to answer these questions.

Perhaps the most obvious answer is to simply calculate how much money the model made. However, this ignores the fact that stocks have different prices and very poor performance in general may be masked by the luck of making a substantial profit on one stock. One might then consider measuring the profit or loss as a percentage of the original buying price but this has its own pitfalls in that if a \$2 stock rises to \$10 that is a 500% increase while a \$100 stock rising to \$150 is only a 50% increase but yields a much greater profit for the same number of shares. A popular measure is return on investment but that is a significantly more complex problem of trying to determine an optimal use of funds rather than just a profitable use. One might also be aware that the market as a whole rises and falls and might want to discount those effects to determine the stocks which perform better than the market. A complex model would likely benefit from the vast corpus of domain knowledge to identify the subtle differences that might mark a stock as optimal rather than just good or bad but the model used for this project had neither the need nor the mechanisms to incorporate such detailed domain knowledge.

A more objective metric that is not hindered by the peculiarities of real-world money and the stock market could be simply to measure how many predictions were correct. The assumption being that as long as the predictions are correct if the model is not making enough money it is simple to tweak it to be more profitable. One could also be more detailed and measure predictions of positive price change and negative price change. This might provide more insight into the operation of the model and the types of errors it might be committing. So if the model is too pessimistic and predicting too many negative price changes one could add a bias to artificially push it to predict more positive price changes and correct the errors.

It was concluded that a simple measure of performance was best suited for a simple objective and the objective was simply to turn a profit. That is, over the course of all the buying and selling, was the model able to identify enough patterns in order to make more than it spent. If this turned out to be the case then it would appear that it was worth engaging in the endeavor.

The practical mechanism for this was tracking the net worth of the model. The net worth was simply the sum of the model's balance and the value of owned stocks. Net worth provided a more representative measure of the model's performance since the balance alone does not show the value gained in a purchase, only the value lost. Because there is no ending state for this problem tracking the net worth allows one to simulate the result if one were to sell all owned stocks while also continuing the experiment. This is particularly important when the experiment reaches the end of the data since it is an artificial stopping point

The model started with a balance of 0 but was able to make as many purchases as it liked. These purchases would cause the balance to become negative, similar to a no-interest loan. Any time the model sold stock the corresponding value would be added to the balance and if it had made a profit the final balance would then be positive. The program did not behave this way in practice, of course, since it was often buying many of the 467 stocks each day and rarely sold enough to offset its purchases plus any pre-existing debt.

## Experiments

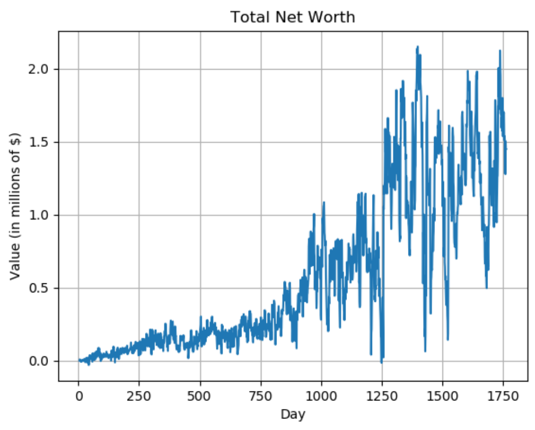
One experiment was performed as described above simply to determine whether the relatively simple Q Learning model was capable of learning enough to make a profit, the performance of the model would demonstrate its ability or inability.

A second experiment was performed as a benchmark using the same mechanisms and code except that the decisions to buy, sell, or hold stocks were random rather than relying on the past experience of the model stored in the reward table. This experiment would allow comparison to determine the effectiveness of the original model.

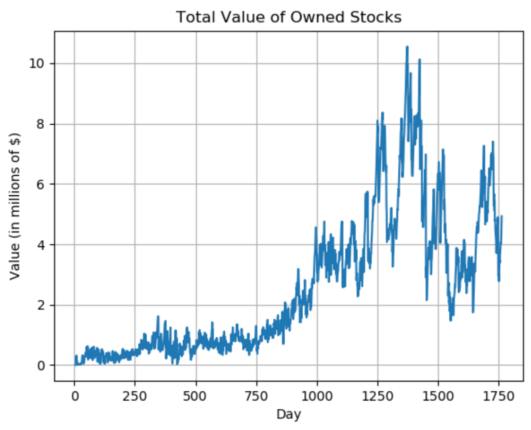
## Results

The Q Learning model performed extremely well. It was able to make a significant profit and performed much better than the random decision model. The net worth of the model trends upward with a monotonically increasing line of best fit. The net worth varies significantly from the line of best fit throughout the experiment. The variation in the graph does not appear large in the first half of the experiment only because of the scale, it is still significant; however the variation in the second half of the experiment is massive, almost \$1M at times.

The value of owned stocks in the Q Learning model increases for most of the experiment. The rate of growth is slow in the first half but increases from the halfway point to the $\frac{3}{4}$ point. Around day 1500 an event in the data seems to cause the price of all stocks to fall and the model is unable to avoid this loss of value, but it recovers afterwards and ends with significant positive value.



(a)



(b)

Figure 1: (a) Total net worth and (b) Value of owned stock of the Q Learning model over the course of the experiment.

The model that made random decisions was also profitable but nowhere near the scale of the Q Learning model. It exhibits similar erratic behavior day to day although the extremes are not as large despite the smaller scale of the graph making them look significant. The net worth grows at a slow, steady rate overall but is not monotonic and dips at times.

The random decision model was run several times with different seeds and all executions behaved similarly. This run shows a large gain in the value of the owned stocks at the very beginning. This gain was present in some executions but somewhat unusual; however, all executions exhibited a similar rate of growth, which is the main factor in the model's ability to make a profit.

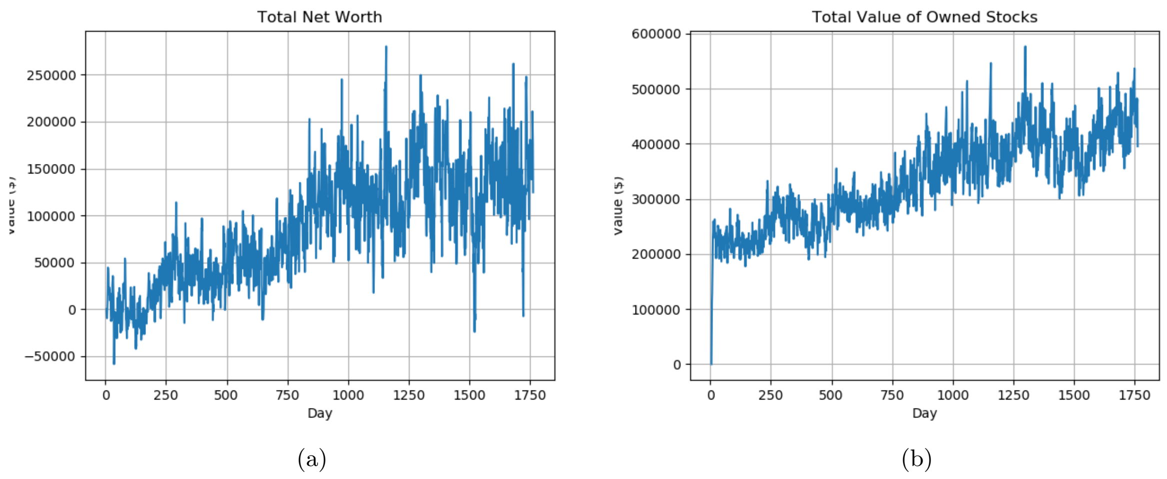


Figure 2: (a) Total net worth and (b) Value of owned stock of the random decision model over the course of the experiment.

After the experiment was run the reward potential table held the learned experience of the model; the patterns found were surprising. Very few of the reward potential values were negative, indicating that the model almost always found it worthwhile to buy. This may be because stock prices tend to rise by default, although an analysis of the data would be necessary to confirm this. Sequences containing any elements of no price change had values close to zero. This is most likely simply due to few of these sequences existing in the data so the reward potential values were changed very seldomly, as a result the confidence in the suggested actions was low. It is difficult to identify any other, more useful patterns without a detailed analysis.

## Discussion

### Performance

The Q Learning model learned to trade stocks successfully and was able to make a substantial profit. It almost never went into deficit, although it did often lose large amounts of money.

This was a simplified environment and many concessions were made to reduce the complexity of the problem which may have benefitted the model. The model had an unlimited budget, although that also meant it could lose as much as possible and do as poorly as possible. It was able to buy or sell every stock every day which, although it may be possible in the real world, is often impractical. There were also no associated costs with trading such as processing fees that would have created an overhead cost. Because of these and other differences it is difficult to say how much validity a model such as this has on actual stock trading.

The model performed quite well despite having a somewhat unusual problem formulation. Q Learning relies on having well-defined sets of states, actions, and interactions between them. That is, given a state and an action the outcome should be deterministic and always result in the same outcome state. In this case the states and actions were well-defined but their interactions were not. A given state and action pair could theoretically result in any of three subsequent states. This could have interfered with the "agency" of the model to control the cause-and-effect of the environment and interrupted its learning. Learning is based on recognizing patterns but if the same state and action pair sometimes results in one state and sometimes in another, how is one supposed to learn the correct action? Similarly, given a state, if sometimes one action leads to the reward and sometimes a different action leads to the reward, how does one know which to choose? Which is better? Apparently, with enough examples the Q Learning model can surmount these uncertain situations and learn based on the probabilities of outcomes rather than having to know exactly what the result will be.

One wonders if providing the model with more information would have made it more successful. If the sequences the model used had been 10, 20, 50, or 100 items long, would the performance have improved? It seems unlikely due to the issues mentioned when discussing the theory behind the model in the Methods section. Longer sequences reduce the model's opportunities to learn and increase the computational cost of the implementation, both of which are strikes against longer sequences. Conversely, if the sequences had only been 3 items long would it have been able to achieve the same success? It is unclear how much of the sequence was necessary. An analysis of the reward potential table may provide some answers but one could also simply run another experiment with the shorter sequence and compare the results and the reward potential tables. It was initially unclear whether 5-item sequences would contain enough information to make accurate predictions so it is a pleasant surprise that even shorter sequences can be considered.

## Random Decisions vs. Q Learning

The Q Learning model performed about ten times better than the random decision model. This indicates that it did indeed learn some patterns that allowed it to perform well rather than simply getting lucky. It is interesting to note that the random decision model did make a profit. This suggests that any combination of buying and selling stocks, as long as one buys and sells with similar frequency, will create a profit. This may be due to the mean stock price of all stocks increasing over time but an analysis of the data would determine if this was the case.

## Conclusions

Although linear regression is the standard approach to this type of problem this experiment showed that other models, such as Q Learning, can be successful and may be less complex and easier to compute. A direct comparison between linear regression and Q Learning would be necessary to determine which is better.

This experiment used a simplified environment and, consequently, there are myriad ways one could make a more realistic environment that could offer new insights into the problem. One could limit the available funds, resulting in more of an optimization problem where the model must decide which stocks will yield the best return. One could limit the number of stocks the model can own or the number or frequency of trades it can make, either by creating a limit or by making these actions incur a fee or cost which would drain the funds available in the balance.

When designing the experiment it was considered whether a lookup table would be sufficient since it was quite limited in size. The other option was to use a neural network to learn the reward function since it would be much more powerful and not artificially limited. The amount of success achieved with the simple lookup table was surprising and shows that the neural network was far more than was needed.

The data set repository from Kaggle [1] included another file which had much of the same data except that the stock prices had been adjusted for stock splits whereas the data set used in this experiment did not account for stock splits. After researching what a stock split is, the adjusted data set seems more correct and that should have been the data set used. However, this mistake was minor and does not seem to have greatly affected the results. I do wonder if this might be the reason for the sudden loss of stock value that occurred in both models shortly after day 1500.

Q Learning is an interesting model which appears able to perform relatively complex tasks despite its apparent simplicity. I was unsure if modifying the problem formulation to fit into the Q Learning framework and applying Q Learning to a slightly different type of problem would work but I was pleasantly surprised with its flexibility. I was worried that using a sequence of 5 price changes would not be enough information to detect patterns or that combining the behaviors of all the stocks into a single predictor system would generalize like I hoped but thinking through the problem, identifying the important characteristics, and following the principles of the model and algorithm created a model that was successful and provided great results.

## References

[1] Dominik Gawlik. New york stock exchange. https://www.kaggle.com/dgawlik/nyse#prices.csv.