Marist College

School of Computer Science and Mathematics

CMPT_496L: Deep Learning w/ Tensor Flow

Predicting Stock Prices with News Headlines and CNN

Author:
Piradon Liengtiraphan

Supervisor:
Dr. Pablo Rivas

December 10, 2017

Contents

1 Abstract / Executive Summary

Quantitative finance has been quite abuzz with the latest developments in Artificial Intelligence and more specifically Deep Learning. To supplement my future career in the financial industry and due to pure curiosity, I would like to propose that my project be my own attempt to predict the seemingly unpredictable currents of the stock market.

Trading stocks on the open market is one of the staples of an investor's experience. While the concept of trying to predict the motion of the stock market accurately is nothing new, and there have certainly been many attempts to do so, with recent development in Machine Learning I believe it would be an interesting approach towards this age old trial. The ability to model and predict an equity's future price, based on the current financial information and news, could be considered the Holy Grail for investors. To use Machine Learning to calculate and predict the performance of a certain company and have that transcribe to usable data for financial advisers and investors would be a great boon, assuming this process takes into account all the relevant factors. Financial balance sheets and various ratios that describe the health of company are the bases of technical analysis that investors undertake to analyze and predict companys future stock prize. Predicting the direction of stock price is particularly important for value investing and with Machine Learning I believe it is possible to expand this particular usage into main stream investing.

2 Introduction / Overview of Project

This project aims to put an interesting spin on predictive quantitative finance, by using news headlines and the dates to which they are attributed to predict the stock prices. The crux of this project rests on how global and local events affect the actions of investors and the stability of the overall market, specifically the Down Jones Industrial Average.

One of the events that inspired the creation of the project was the aftermath of the 2016 US Presidential Election. With the results being announced the next day for public distribution, through the major news outlets, the Down Jones Industrial Average (DJIA) experienced a surge in value; however, when looking deeper into the the specific companies that raised the the DJIA a trend was noticed. Companies with closer ties with the Republican party experienced a sharp rise in investment, while their Democratic counterparts experiences a small drop. The connecting thing between these rises and falls lies in the fact that a Republican president had won the office, as opposed to a Democratic one. The implication being that in the wake of a Republican victory, investment in Republican associated companies would be more profitable; assuming that the administration adheres to the trend of giving favor towards companies associated with their respective party. The old saying goes "The pen is mightier than the sword", this projects aims to see if this old adage is true, by finding trends associated with particular words as they appear in the headlines, one might be able to predict the direction that a market will take while said headline is in effect.

Should this project yield applicable results, we may be able to improve our predictive quantitative finance tools by directly interfacing with the new outlets to help determine the direction of markets around the world.

3 Background and/or Related Work

This project will be my first entry into the realm of quantitative finance, with my interest in the topic peaked by my previous experience working in the financial industry. In previous semesters, I have taken a machine learning class, which enabled me to have rudimentary experience within this field. However, having a background in finance allows me to understand the factors which could contribute to a shift in market direction. By working through this project I hope to expand my knowledge in both machine learning and finance. While research into the prediction of stock prices is as old as investment banking itself, most algorithms fail to take into account the effect of uncontrollable, external forces, which are independent of the stock market entirely. Having worked previously in cyber security, there is a certain train of thought that external factors influence the security of a system as much as internal factors. A system can be thought of as secure an unassailable, i.e. not worth the effort, due to all the internal factors that are in place to prevent such an event; however, it can at the same time be unprepared to contend with external factors that are outside of its jurisdiction. Ergo leading to something completely unrelated rendering an otherwise secure system vulnerable, even if it is only temporarily. I hope to apply this train of thought to this experiment with the financial industry; quantifying the effect of external factors on stock prices.

4 Methodology

The first step in starting this project was to gather data. The website Kaggle which is a platform for predictive modelling and analytics competitions, contains within it a number of data sets on a plethora of topics. Among these datasets contains ones relating to news headlines and stock prices. The next step was to find one that were relevant to the project (i.e. *all_stocks_5yr.csv* && *Combined_News_DJIA.csv*) and download them. The next step is to massage the data into a format that will be useful for the recurrent neural network that will be used to process the data. Since the datasets were not originally designed to work with one another we must first find within them common points that can be used to coalesce the data sets, in this case "Date".

After all the dates have been formatted to a standardized format and selecting the index that we want to track. The process to parse the data is as follows: run through the date associated with each news headline, and check if there is a matching date within the stock prices data set. If there is a matching date then store the closing price for that stock in a local variable and write a new entry containing: date, month, day of the week, headline, and closing price to the .csv file, separated by the caret symbol. The caret symbol was chosen for its uncommonly within normal written mediums. The next step is to transform the news headline into the a format that is useful to the recurrent neural network. Using the module *gensim* and the function *word2vec* located within we are hoping to transform the headline into a vector which then can be passed onto the RNN for processing. The data is then finally transformed into multiple flattened and unflattened .npy (numpy array) files.

The deep learning model chosen for particular experiment is a recurrent neural network. A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed cycle. This allows the exhibition of dynamic temporal behavior. RNN differs from feedforward neural networks in that RNNs can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition (MNIST) or speech recognition. The recurrent neural network will be optimized with **AdamsOptimizer** and minimized used **mean_squared_error**.

Adam is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal re-scaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters.

Defined in *tensorflow/python/ops/losses/losses_impl.py* **mean_squared_error** adds a Sum-of-Squares loss to the training procedure. The weights acts as coefficients for the loss within training and testing. If a scalar value is provided within the parameters, then the loss is scaled by the value provided. If the weights parameter is a tensor of size [n], then the total loss for each sample of the batch is re-scaled by the corresponding element in the weights vector. If the shape of weights matches the shape of predictions, then the loss of each measurable element of predictions is scaled by the corresponding value of weights.

5 Experiments

The experiments performed in this project were were in pursuit of attaining the best possible result, from a theory that is attempting to grasp at such an elusive concept. With the stock market being inherently unpredictable, the goal of this project operated along the lines of the *"How close can we get to the true value?"* philosophy. Once the methodology was completed the next step was to run the the recurrent neural network on the data set for the first time. The resulting *testing mean* of 365.936 was unfortunately a horrible result. From this particular result we can reach the conclusion that, it is better off to guess at random than attempt to use the neural network to predict the closing price of a certain stock.

To clarify, the *testing mean* is being calculated by allowing the neural network to use the context of the news headlines of the day in its calculation of the predicted closing price for that particular day. That prediction is then given context by being compared to the actual closing price of that day and calculating the *mean_absolute_error*. In the example given above of, extrapolating the meaning from the result in the current context of what it ultimately means, we can see that given the correct value for a certain day in this case being around (326.98) subtracted by the value predicted by the neural network the resulting *mean_absolute_error* is off by a factor of almost double the actual value. Without the need for explanation, this is obviously bad.

The next step in this process is to shift the different parameters of the neural network in an attempt to achieve better results, or at least to get the neural network to understand and process the data better than it currently does in its base state. To do this we must first understand the different parameters in flux here.

- n_input - Dimensional parameter (This remains static)
- n_steps - Dimensional parameter (This remains static)
- n_hidden - The number of hidden layers between the input and whose output is connected to other neurons
- n_classes - The number of target classes for our output (This remains static)

- learning_rate - Training parameter that controls the size of weight and bias changes in learning of the training algorithm
- training_iters - How many iterations of training the RNN goes through (This remains static due to size of data not changing, and maximizing training)
- batch_size - The number of headline vectors the network is looking at at each iteration of train. This number determines training_iters (This remains static, as this was determined to be a good training size)

Next is to alter these particular values in an attempt to optimize the neural network. The challenges we are trying to avoid are underfitting, when a statistical model or machine learning algorithm cannot capture the underlying trend of the data, and overfitting, a modeling error which occurs when a function is too closely fit to a limited set of data points. The two parameters that have the largest effect on the results of the neural network, with this data set gather are n_hidden and learning_rate; these two parameters in alteration will make up a majority of the experiments tested on this recurrent neural network.

In the first experiment we increased the number of hidden layers by a factor of sixteen, taking the base case of 128 hidden layers to 4096 hidden layers. This was mainly to see at which point we would experience overfitting, then work our down from that point to the optimal number of hidden layers. By knowing this number we can make an assumption about the data. If increasing the number of hidden layers does not significantly degrade the quality of the testing mean then the model is complex enough or has enough information to support the increased layer count, indicating a robust data set. The batch size is also a relatively small number, reducing the chance of dilution during training. The resulting testing mean of the first experiment was (insert mean here), most likely due to extreme overfitting, the number of hidden layers was incrementally decremented until we arrived at the "sweet spot". It would seem that for this data set, we achieve the best result when the number of hidden layers rest between 512 and 1024.

The next parameter to modify was the learning rate. As stated above this parameter determines the size of weight and bias changes in the learning of the training algorithm. Meaning that by modifying this parameter we can control the amount of information that neural network is able to see and "learn" from at any given point. Underfitting is the main challenge that we are trying to minimize when altering this particular parameter. Too fast and the neural network will not able about to "learn" the data effectively enough to make an accurate prediction before the next set is passed on. This will also increase the amount that the recurrent neural network will "forget". Too slow and we result in the same conclusion, but in the opposite direction, the network spending too much time on one batch that it forget everything that came before. With that thought in mind given that the base learning rate was 0.0001 we decided to start shifting the decimal point rightwards by one significant digit (i.e. learning rate $\bar{0}.001$). Eventually the ideal learning rate of 0.01 was found and saved, one more significant value to the right resulted in a higher finalized testing mean, which could indicate signs of underfitting due to the learning rate being too high.

An interesting case that was proposed by Professor Rivas was that, the testing mean being returned was not an accurate representation of the data due to the zero padding that was needed to ensure uniformity between the different headline vectors. The proposed problem was that, given the fact that the headlines appear near the beginning of the batch, with the rest being padded with zeros, by the time the neural network has reached the end of the data set to make a prediction, it would have "forgotten" all the headlines already. To rectify the solution, we flipped the vectors so that the headlines appeared last and observed the results. Unfortunately there was no significant difference, for better or for worse, in the testing mean returned.

6 Discussion and/or Analysis

So ultimately we must ask ourselves, what does any of this data mean for anyone? It is best to keep in mind that original premise of this project when evaluating the results stemmed from this project: To predict stock prices using news headlines. Going into this project there was always the suspicion that the final results of this project will yield inconclusive to outright bad results. With that in mind, we can begin the discussion regarding the results of this project.

In this Discussion and Analysis section we will be discussing the parameters that lead to the best possible testing mean generated from the gathered new headlines and stock closing price data set. The conditions are as follows:

- n_input - 200
- n_steps - 491
- n_hidden - 1024
- n_classes - 1
- learning_rate - 0.01
- training_iters - 19975
- batch_size - 25

The testing mean that was returned with these parameters was **91.8063** this is significantly better than the initial result that came about from the base condition, which was valued at **365.559**. So what does this improvement tell us about the data set and the result of this project in achieving its stated goal? From the improvement of the testing mean we know for a fact that factors that contributed to this out come where the changes in two parameters namely: hidden layers and learning rate. From the increase number of hidden layers, from **128** to **1024**, we can know that the model is complex enough or has enough information to support the increased layer count, indicating a robust data set. To the point where there is little to no degradation in resulting testing mean.

The next factor to discuss in the learning rate, **0.01** is a relatively fast learning rate when compared to the original learning rate of the base case **0.0001**. This implies that as we increase learning rate we allow the the recurrent neural network to minimize the functions more, leading the more efficient "learning" by the neural network. By finding the ideal number to maximize the efficiency of the minimization of the function, all-the-while avoiding the problem of minimizing too much, we can assist in the containment of the neural network in generating inaccurate data. There is also no decay on the learning rate, as the news headline vectors being passed into the recurrent neural network, are all unique to each day, the probability of repetitive headlines along concurrent days are quite low, thanks to the original data gathering team. This implies that the neural network is learning throughout the entire process of training, with little chance of overfitting. The benefit of this is that for each Epoch in training, we can ascertain that the neural network is constantly familiarizing itself with the data input and improve upon its previous understanding; by the time that it reaches the testing data, it would have been trained for the maximum amount of time.

The implications of the testing mean return is slightly harder to infer. Given that testing mean is measurement of how inaccurate the predicted value is from the actual value, we can see that regardless of the improvements made along the way by optimizing the neural network to the data set, the predicted value is still incorrect. As stated at the beginning of this section we acknowledge the fact that this project exists more as a though experiment than a actual practicality. The testing mean produced in the best case scenario was **91.8063**, meaning that the predicted value is off from the actual value by about the same value. While inaccurate the number generate is still better than simply guesswork. Leaving reasonable doubt that with enough data and optimal testing this value could be improved further. If project fails in its premise to predict stock prices; however, it seems to have stumbled upon a connection that was most likely ignored before. With this information in hand we can use this possible correlation to improve upon current stock prediction algorithms to take advantage of the inclusion of external, and unrelated factors.

However that does not mean that this experiment was done in vain. By performing this experiment we know that there seems to be enough of a correlation between news headlines and stock prices that we can improve our prediction number, consistently by modifying the neural network to more optimally take in input and generate out put. The implications of this experiment still stand; that stock prices are somewhat affected by "external factors" such as news headlines. To further this project we would probably modify the initial premise. Instead of attempting to predict the entire stock price, we would use the previous day's closing price and use the daily news to predict percent change. Should this project be given more time, and enough interest was garner, the parameters of the experiment would most likely be changed along these lines to allow for more practical usage of the resulting recurrent neural network.

7 Conclusion

What I had hoped to achieve with this experiment was to be able to find discernible patterns and correlations between the inherently random and unpredictable stock market, and the daily events that occur during its valuation. Ideally a correlation between these two seemingly unrelated element could have been found. This resulting data, if it had been more significant, would have been used to help companies gain a significant advantage, on their competitors who have chosen to ignore the effect of externally unrelated events on the intrinsically internal nature of stock prices. At the current time, near the conclusion of the semester, values generated taking into account only news headlines of the day to predict stock prices, are still off by a large margin. It has been acknowledge in previous sections that the premise of this experiment could be altered to achieve better and more applicable results. However if an assumption based on the data that we have collected and the result generated were to be made it would most likely be as follows.

From the generated testing mean as a result of training a recurrent neural network to predict stock prices based off of nothing but news headlines, we can conclude that there is no significant correlation to be discerned that would justify stock prediction based solely off of this singular input. Given that the best testing mean generated was **91.8063**, we can assume that there is a correlation, as the number is still better than random guessing. With this correlation in mind, the next logical step would be to take advantage of it as effectively as possible, by modifying the goal of the the project from predicting the entire stock price to using the previous day's stock price and the daily headlines of the day to predict percent change. I believe that once we start operating with this premise, the data returned from the recurrent neural network, will experience a large jump in statistical significance.

In conclusion, while the results of the project were not what was initially hoped for, the knowledge gained through its completion still has the potential for positive impact, assuming it it applied properly. I hope to continue this research, if at all possible, modifying it to its new goal of predicting percent change within stock. This would allow for better use of the data gathered, and the possibility of aiding future stock prediction algorithms or the improvement of those currently in existence.

I would like to thank Professor Rivas for his assistance during this entire en devour. I can confidently say that none of this would be possible without him. As an novice in the area of machine learning, I was prone to asking for help quite often. I apologies for any inconvenience caused, and once again thank him for his patience and tutelage.