Utilizing a Long Short-Term Memory Algorithm to Predict Closing Price of Stocks  
Ross Bravo

The motivation for producing this algorithm was for personal growth and practice. After completing my undergraduate degree, I wish to pursue a PhD in mathematics and then work in the field of Quantitative Analysis for a hedge fund. Quants work heavily with stocks and prediction/analysis algorithms. So, producing this project gave me an exposure to the terminology of the field while also allowing me to practice my programming skills. The practicality of this algorithm will provide stability in a chaotic stock market because if brokers have increased knowledge and stability then they will have more reason for trading which will increase financial productivity. Which is great for the economy. I approached the algorithm from a technical analysis of the stock market and utilized a method known as Long Short-Term Memory (LSTM), which essentially uses gates to store important information and forget information that is not relevant to the analysis of the stocks closing price. This yielded significantly positive results when the Root Mean Square Error was computed close to $11. This is a significantly better result compared to my first iteration that utilized k-Nearest Neighbors (KNN) and resulted in the Root Mean Square Error being computed close to $115.

The problem in the current atmosphere of stock trading is the extrodinarlly high risk that is run to those buying and selling stocks. It is an extremely lucrative exchange but also one of the highest risk ventures that any investor can partake in. When the risk out weighs the reward is when investors tend to become stingy with their money and the stock market tends to take a hit in productivity. My goal is to develop an algorithm that can predict the closing price of a stock. This information will help give a higher degree of certainty to investors which will help to negate some of the risk. In turn this will keep the productivity of the stock market at a higher level because the risk to reward ratio is smaller and easier to manage. This is important because in a capitalistic economy such as America’s, when the capital market is thriving, then the economy is thriving. According to the U.S. Department of the Treasury, the capital markets fund 65% of economic activity in the U.S. (Impose A Tax on Financial Transactions). This money goes towards taxes and 401k’s. So, when the capital market slows down on productivity the entire country takes a hit. This algorithm will take the Open, High, and Low prices of the stock in a given day, and the Volume that it has been traded at, while also taking the closing price of the past 60 days into account as well. Then using this input, it will output the expected closing price of the day.

There are plenty of other attempts at predicting closing prices for stocks. The first one I want to discuss is Vivek Palaniappan’s approach in the article “Neural Networks to Predict the Market.” Vivek also uses an LSTM for analyzing the stock price of Apple. His worked pretty well but a major weakness in his program was his LSTM only utilized 20 units whereas my program used 50. Units are analogous to a hidden layer in standard feed-forward neural networks. So, his algorithm had significantly less weights and biases to work with which let more data that should be forgotten stick around longer. He may have done this for technical limitations or to decrease the time it took for computation, but nonetheless it hindered the accuracy of his algorithm. The strengths lied in the trends of the predictions. His algorithm may not have guessed the price as close as mine had but his predicted the trend more consistently than mine. For example, when the price increased his prediction increased and when the price fell his prediction decreased. Whereas my predicted price would just hover around the range of the actual price. Nonetheless Vivek’s approach is particularly brilliant. Reading his article is what inspired me to try and utilize an LSTM in my algorithm. Whereas before I was using a KNN with significantly less accurate results.

The next approach I wanted to touch base on was that of Joshua Wyatt Smith in his article “Stock prediction using recurrent neural networks”. Smith uses a recurrent neural network (RNN) which is a neural network where nodes are connected to form a directed graph based around a temporal sequence. LSTM is a form of RNN but smith recognized his lack of an understanding of LSTMs is the reason he went with a broader RNN. His algorithm resulted in a 65% accuracy when predicting the gradient of the stocks which without the use of an LSTM is quite impressive. From what I analyzed his main flaw came from overtraining. This is because he concatenated multiple different stocks to make a general model for a sector of the financial market instead of specific model for one companies’ stock. This caused it to be more difficult to train because the data is not from one source and companies’ stock prices in theory are totally independent from each other. Emphasis on ‘in theory’ because in reality predicting the behavior and rational of a stock broker is impossible and it may be based on the prices of another companies’ stock, so maybe Smith was on to something by generalizing into sectors and that is what yielded a high accuracy using an RNN.

I obtained my dataset from Quandl.com which is a public website where anyone can access a plethora of financial data on hundreds of publicly traded companies. I used Boeing’s stock price data from September 2013 to December 2017. I had 80% of my data going towards training which wound up being 857 data points. Then I had the remaining 20% going to a testing category. Since my algorithm was utilizing the previous 60 days of data the training set was the first 80% of dates in chronological order and the test set was the backend of the dataset. I gave the Open, High, and Low prices of the stock in a given day, and the Volume that it has been traded at, while also taking the closing price of the past 60 days as the input values. Then using this input, it produced the expected closing price of the day as the output. I used sklearn.preprocessing’s MinMaxScaler to normalize the data.

I used a lot of functionality and features in this program. I used pandas and NumPy to obtain, organize, and manipulate the data. Also, pyplot was used to display the graph of closing prices and predicted prices in the notebook. I used pylab to set the figure size and Keras to build and design my LSTM model. Finally, I also used NumPy to calculated the Root Mean Squared Error.

The learning algorithm that I used was Long Short-Term Memory. My model was a sequential model which means it is a linear stack of layers. The first layer used has 50 memory units and it returns sequences so that the next layer receives sequences and not just randomly scattered data. Then the next layer is another 50 memory units. The final layer is a fully connected layer with only one neuron for the predicted closing price. I use an Adam optimizer which is an optimizer that computes individual learning rates for each different parameter. It does this by using estimations of the first and second moments of the gradient to adapt the learning rate for each weight of the neural network.

As for how an LSTM algorithm works is a bit more complicated but here is a short description of it. An LSTM network is made up of different memory blocks called cells. Each ell is responsible for remembering the cell state and the hidden state. Manipulations to these memories are done through the forget gate, input gate, and output gate. The forget gate is what removes information from the cell state. This information is the useless or less important bits of data that do not help understand or compute the end task. The forget gate takes as input, the output of the previous cell and the input of that particular step in the algorithm. These inputs are multiplied by the appropriate weight matrix and then adds the bias. Sigmoid is then applied to determine if the cell state should forget (‘0’) or keep (‘1’) this information. Next the input gate is what adds information to the cell state. First it takes in input from the output of the previous state and the input from the current step and applies a sigmoid filter similar to the forget gate. Then it creates a vector of all possible values that can be added to the cell state using a tanh function. Finally, it multiplies the results from the first two steps then adding this information to the cell state. Finally, the output gate scales the values of the cell state to the range -1 to +1 using tanh. Then multiplies it with the values of the sigmoid function applied to the input data which is the current states input and the output of the previous cell. Then sends this new information to the hidden state of this cell and to the next cell.

The first attempt I made at this algorithm was utilizing a KNN process instead of a LSTM. KNN was significantly less efficient. It was able to identify major drops in the data but the Root Mean Square error was close to $115. The reason for the failure of KNN is that it is a regression algorithm applied to a time-series problem. It was after reading Vivek’s article “Neural Networks to Predict the Market” that I decided to transition to a LSTM.

More experimentation was required when deciding how many previous closing price data points I should use as input. I settled on the last 60 days for a few reasons. When I went anything less than 60, for example 40 previous days, I would receive greater error because there wasn’t enough input to help make a justified prediction. The other reason for 60 was overfitting. If I went higher, for example 80 previous days, I would run into overfitting. Where the analysis of the data would become too closely related to the closing price of the previous 80 days and this wound up not fitting future data if there were extreme spikes. The problem with this is that the stock market tends to roll with extreme spikes. So, if there is a massive drop or increase then the price tends to stay around that area for the next few weeks. So, the algorithm would be fit to the last 80 days and ignore the spike and would now be off for the next few weeks till enough data from the new statuesque made from the spike would affect the prediction. After a few trial runs of incrementing by 5 days from 35 days to 100 days, 60 days resulted in the smallest average Root Mean Square Error. It provided a good balance between enough data but not too much data.

After all was finished my algorithm provided consistent and efficient results. With a Root Mean Square Error on 10 runs averaging to $11. With the lowest being $9.2 and the highest being $15.7. Graph 1 shows the closing price for the Boeing’s stock.    
*Graph 1 shows the results of one trial run of my algorithm. The Root Mean Square Error resulted in $9.24.*  
The blue line is the train data, orange line is the expected value, and green line is the predicted value. This was significantly better results than what I was expecting. That said, I do imagine there are programs out there that can beat mine and may be more accurate.

Perfectly predicting the stock market is essentially impossible. My data does not cover a major recession or depression. The uncertainty of the economy comes from the free will that stock brokers and humans possess. We define the value of companies and that definition can change on a whim. Either from advancement of technology, a natural disaster that wipes out their manufacturing plants, a scandal or death of the leadership within the company, or an all-out market collapse similar to that which we experienced in 2008. So many unknowable and uncontrollable variables can affect the stock price. Algorithms like these will only be good for day to day business for stock analysis or trade but for it to be productive in the stock market it would need to be paired with fail safes that could take over whenever a massive unordinary occurrence rocks the stock market.

In conclusion, this project granted me an introduction into the world of stocks while also letting me practice my programming skills. It also helped to begin to think of solutions to the problem that a high-risk instable capital market can have. Comparing the use of other algorithms from other authors or by experimenting myself, I have come to the conclusion that a LSTM is the most efficient approach we have to solving such a complex problem. Though there is a lot of work left to do in making a better algorithm, I am proud of the results that were produced with only being off by approximately $11.

If given more time and resources I feel I would take this project further in a few directions. First, my implementation of using the past 60 closing prices as part of the calculation needs to be more generalized. With such a quick deadline I just brute forced a solution by testing a bunch of different options and seeing what works best. But the past 60 might only work best for this specific data set of Boeing stock prices from 2013-2017. So, finding a way to quickly determine the best amount of previous closing prices to use would be beneficial. Which leads to my next development, to make the algorithm more general. I have not tested it on other stocks, or trained it to deal with huge spikes such as a recession or depression. I believe Smith’s approach to generalizing an algorithm like this is worth while exploring. In his article “Stock Prediction Using Recurrent Neural Networks” he explains how he made it more general by grouping certain company stocks into sectors and predicting how that sector would perform. Utilizing sectors could give a more accurate prediction of the stocks price because some companies, although not physically linked, may affect one another’s stock prices. So, keeping in mind how a company’s direct competitors are performing might provide insight on how the company will perform. So, merging an analysis of sectors with the analysis of a specific company may provide a more wholistic and accurate analysis of stock prices.

Bibliography

“Financial Transaction Taxes: A Fact Sheet.” *Securities Industry and Financial Markets Association*, Securities Industry and Financial Markets Association, 2019, www.sifma.org/resources/general/financial-transaction-taxes-a-fact-sheet/.

“Impose a Tax on Financial Transactions.” *Congressional Budget Office*, 13 Dec. 2018, www.cbo.gov/budget-options/2018/54823.

Palaniappan, Vivek. “Neural Networks to Predict the Market.” *Medium*, Towards Data Science, 21 Nov. 2018, towardsdatascience.com/neural-networks-to-predict-the-market-c4861b649371.

Smith, Joshua Wyatt. “Stock Prediction Using Recurrent Neural Networks.” *Medium*, Towards Data Science, 21 Aug. 2019, towardsdatascience.com/stock-prediction-using-recurrent-neural-networks-c03637437578.

Srivastava, Pranjal. “Essentials of Deep Learning : Introduction to Long Short Term Memory.” *Analytics Vidhya*, 23 Dec. 2017, www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/.

References:  
Pandas, NumPy, Scikit-learn, Tensorflow, Keras, Pyplot, Pylab