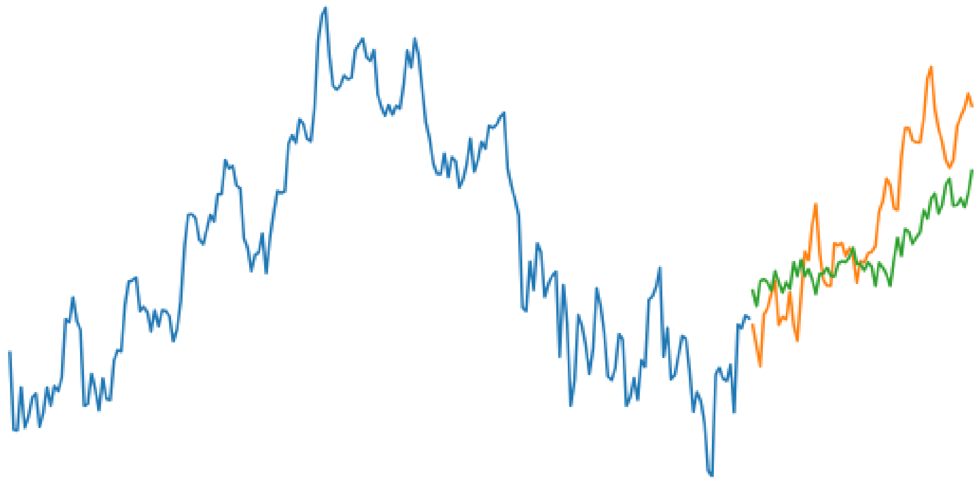
Machine Stock Prediction

Calvin Garrett, Jim Greene, Gustavo Correa



2019/04/09

LSTM

Predicting the stock market has been a popular topic in machine learning for decades. We began our own exploration, starting with the Long Short-Term Memory machine learning model (LSTM).

We used a Yahoo Finance library to pull the stock data. We adjusted the program to take the stock ticker (name) from the user each time the program is run. Using the system date, it retrieves the last 5 years of data from Yahoo.

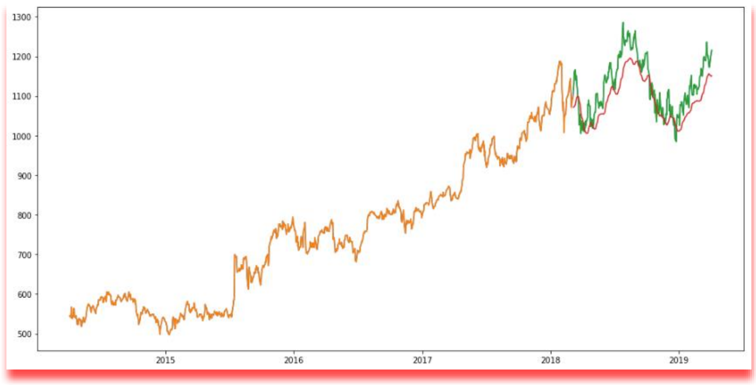
For testing, we used Tata Global Beverages. Our model uses only the close value and makes predictions on that. So, we dropped all the other columns. We also min-max scaled the values. Because stock prices are well maintained, data collection was the easiest part of our project. The data science, however, is a bit more challenging.

Our research showed that using a recurrent neural network called Long Short-Term Memory, or LSTM, was a good foundation in predicting stocks. We also researched a bit into using Support Vector Machines, or SVM, but in the end we decided that LSTM was working well enough for our needs and we also wanted to focus on better understanding the recurrent neural network.

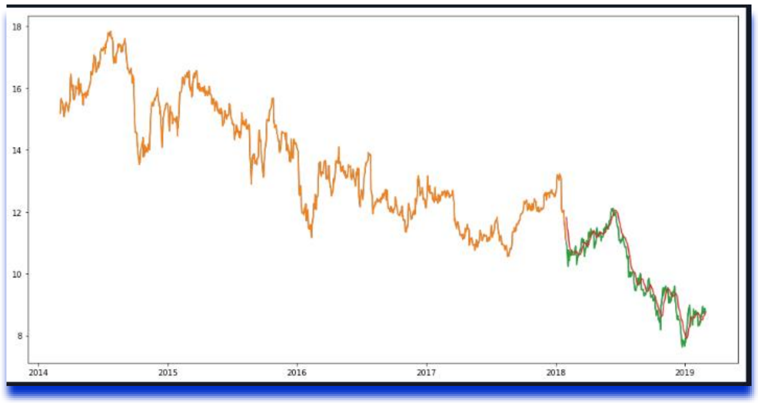
We started with the LSTM example from a blog on Analytics Vidhya. This algorithm uses the previous sixty data inputs to help predict each stock closing price, one day at a time. When the program runs, four neural networks with unique tasks are trained to determine what data to keep or remove from the memory and what to do with the new data, to make each prediction.

Because we did not fully understand the Keras library, testing with different parameters was usually unsuccessful. Using more epochs improved the network.

We trained the model on four years of a specific stock and had it predict the fifth year. In the graphs below, the first four years are represented with an orange line, and the fifth year is in green. The red line shows the prediction.



Google stock prediction



Ford stock prediction

More control is needed so that predictions into the future are not dependent on the actual values of the previous day. We learned it is useful to train on a random walk to compare the quality of predictions.

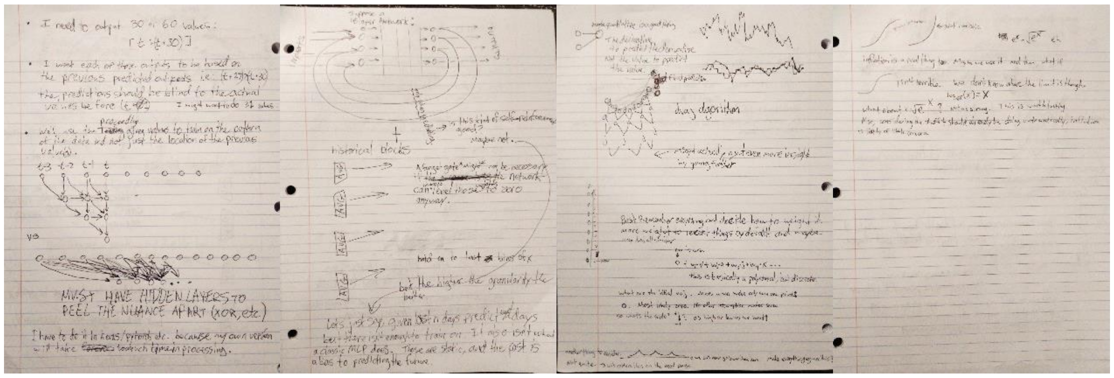
So far, our results have suggested that LSTMs could be used on stocks to help with deciding when to buy or sell stocks. By adjusting parameters and using aggregation we believe we could obtain useful results.

Discrete Derivatives Model

After this introduction, we decided to temporarily step away from using LSTM and think about the nature of the problem.

- The ideal model would predict a certain number of days into the future. So, the network would have to output multiple values.
- Proceeding values should be used to train the model on the pattern of the data and not just the values themselves.

At a high level, we sketched out some possibilities for a method by which this could be done.

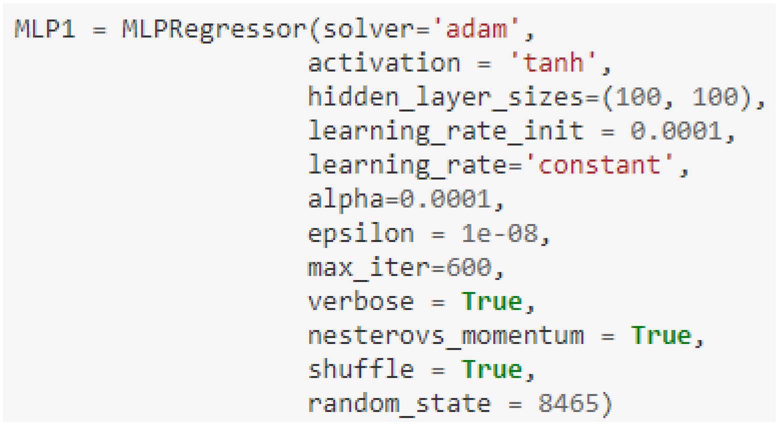

First concept sketches

The concept used involves taking the differences between each day and using that to predict the next change, after the last day.

The network could train on any examples of consecutive days in the previous data.

If, for example, there were 1000 days in the historical data, and we wanted the network to learn on 60-day segments, then it could train on 939 examples; i.e. 1000, minus 60, minus the last day to be predicted.

This can be done by simply using a multi-layer perceptron, and after consulting with another student and AI researcher, Filipe Cancino; we decided to use the Sci-Kit Learn implementation of the MLP regression.

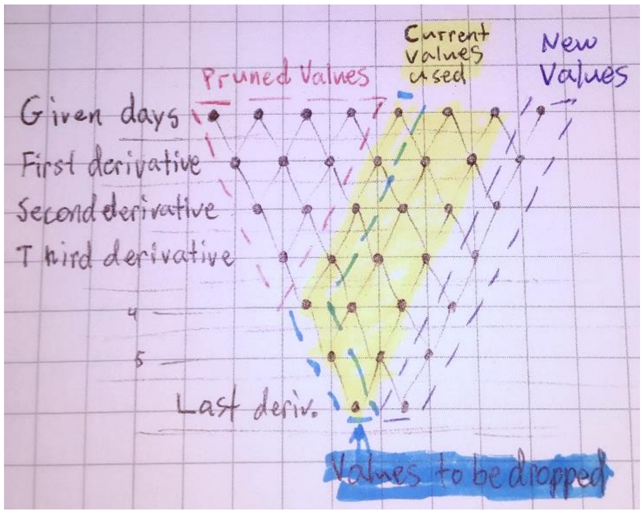


```
MLP1 = MLPRegressor(solver='adam',
                    activation = 'tanh',
                    hidden_layer_sizes=(100, 100),
                    learning_rate_init = 0.0001,
                    learning_rate='constant',
                    alpha=0.0001,
                    epsilon = 1e-08,
                    max_iter=600,
                    verbose = True,
                    nesterovs_momentum = True,
                    shuffle = True,
                    random_state = 8465)
```

Settings used for the regression

Because this is not a recurrent network, rather than hoping the model would magically learn about the patterns, we explicitly gave some essence of the pattern as inputs: After computing the differences between each input day, we compute the differences between those differences, and the differences between those, giving the network a collection discrete derivatives to learn off of.

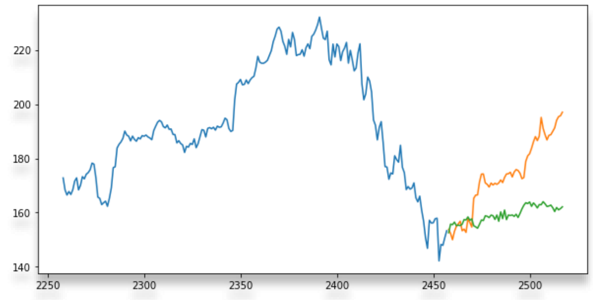
By doing this, the number of input nodes that can be used becomes N(N+1)/2, with N being the number of days in the example. Because the model only predicts one day at a time, when we need to run a prediction a number of days into the 'horizon' or future; we run the prediction in a loop, creating new inputs by adding the previously predicted day to the input vector and dropping values and dropping the oldest values.

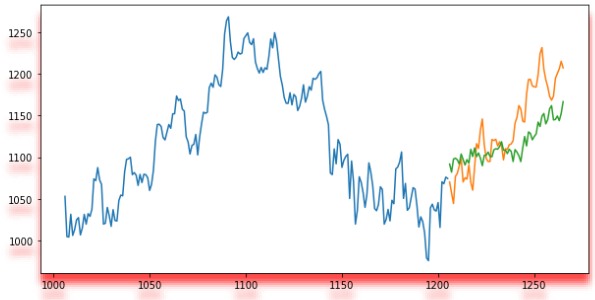


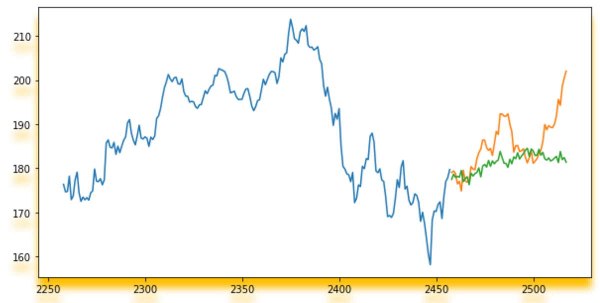
Prune and update diagram

Using all N(N+1)/2 nodes is both computationally expensive and generally ineffective for prediction (The insight gained from older nodes is smaller, and we also have a problem with vanishing gradients). So, the code includes a pruning process to only keep the last X values from each derivative. All variables were z-score standardized.

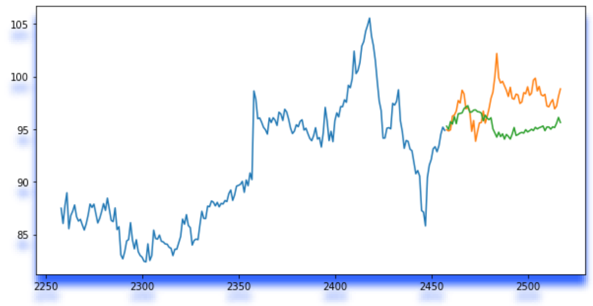
After spending several hours exploring possible parameters and input dimensions, we made a single prediction on four different stocks. This method trains and predicts 61 days in about 39 seconds with Jupyter Notebook.


APPL (Apple)


GOOG (Alphabet C)



HD (Home Depot)



WMT (Walmart)

Although the model often predicts the general trend, we want to do more before using this to make trading decisions. There are several variables that could be added to provide a richer context to the predictor. These include the daily transaction volume and opening prices. Many others have had success including knowledge about the news. Further, it is impossible for this model to predict a sudden boom or crash. It would be interesting to attempt a model solely for that purpose.

End Thoughts

It has been suggested that machine learning should not be applied to the stock market. One problem is that it makes activities like option trading more unfair to those who aren't applying technology, than it already is. Another problem is that predicting the stock market influences its outcome. To reduce this, the models need to be kept fairly private. Whether or not these are serious concerns, it is possible that if excessive hype develops over AI stock prediction it will cause a crash.

We've learned more about recurrent neural networks, particularly LSTMs, and understand how to design a working stock market predictor. Because of this project, we are better prepared to research and apply machine learning techniques with other topics.