Using LSTM Network to Predict Stock Prices

Franklin Jia
{frankyj3}@stanford.edu

Abstract

The goal of this project was to build an LSTM network to predict stock prices using publicly available closing prices from the S&P500. Three different predictions were measured: Day-by-Day prediction, Whole Sequence prediction, and Tendency prediction. Of the three, the Day-by-Day prediction saw the most success, while the Tendency prediction struggled greatly. Hyperparameter tuning was implemented to optimize the LSTM model after preliminary testing.

1 Introduction

The financial industry has been slow to integrate machine learning and AI into their system. The problem I hope to tackle is the effectiveness of an RNN (our more specifically a LSTM) for predicting future stock prices. This is an interesting problem because if deep learning can accurately predict stock movements given historical data, this would be an enormous threat to money managers in the financial industry, especially if the process can also be automated.

Effectively predicting stock prices has been a problem that many have tried to solve, achieving varying degrees of success. I wanted to attempt this difficult problem because I too wish to be able to better understand the stock market. I hoped that working on a deep learning project involving stock price prediction would help me take one step forward.

The inputs to my LSTM model are windows of S&P500 closing prices. The window is comprised of the prices for a specified number of days that the user wishes to use to predict the closing price of the next day, and the actual price of the next day at the end of the window. That makes up one input. The output of the model is a number that represents the next day's predicted closing price.

2 Related Work

There are a few existing projects I referred to while building my model. I studied an existing implementation from github made by user lilianweng. She tried to do both next day stock price prediction with closing prices, as well as using embeddings. I didn't pay as much attention to the embeddings because that was not where I wanted to go with this project at this particular time. In addition, it seemed that her LSTM network was not particularly complex (only one layer) and she did mention in a blog post that she didn't optimize any hyperparameters. However, her sliding windows approach was similar to the approach that I ultimately took. I also read the paper "Comparative Study of Stock Trend Prediction Using Time Delay, Recurrent and Probabilistic Neural Networks", which proved to be a really good introduction for me to get started.

3 Dataset Preparation

3.1 Platform

I used Keras with Tensorflow backend to implement my LSTM model. Everything was built from scratch.

3.2 Data Generation

The S&P500 dataset is downloaded from Yahoo! Finance. It came with the Date, Open Price, High Price, Low Price, Close Price and Volume. The data is then preprocessed so that I only take the Closing Prices from Jan 3, 1950 until February 23, 2018.

During preprocessing, I split the dataset into windows. Each window contained the prices for a specified number of days (after hyperparameter tuning, I found 30 days was a good number) and the ground truth price for the next day. These windows made up the inputs to the model. I then proceeded to normalize the dataset by dividing prices within a window by the first price in that window and subtracting 1. There were around 17000 closing prices in the dataset, and I used 10 percent of the dataset for the test set and 90 percent for the training set. During training, there was also a validation split of 5 percent from the training set. I also kept a copy of the unnormalized ground truth values in the test set so I could see the stock prices after denormalizing the predictions.

4 Methods

4.1 Model

The model that I decided to build is an LSTM or Long Short-Term Memory network. Most of the RNN successes in a variety of fields have happened as a result of LSTMs. The reason why I chose to use a LSTM is because even though a RNN (the basic structure with only a tanh activation) in theory should be able to learn to use past information, but in practice, it doesn't work out so well. RNNs work well in cases when the gap between the relevant information and the place that it's needed is small, but it fails to learn this connection with the gap gets bigger. This is where LSTMs have managed to excel where standard RNNs fail. I wish to see the connections between long-term trends so I chose to use a LSTM over a standard RNN.

For my implementation, I executed tests for hyperparameter tuning to try and optimize my model further. The discussion on hyperparameter tuning can be found in the next section.

For my model, I implemented a sequential LSTM network with 3 LSTM layers and a dense layer. Each LSTM layer had 30 units. The loss function I used was RMSE (Root Mean Squared Error), which I picked because RMSE severely punishes large errors. The optimizer I chose was RMSProp because from the online implementations and papers that I read, this seemed to be the prevalent choice. The LSTM model also has a timestep of 30, which is necessary for backpropagation through time (BPTT). The batch size was chosen to be 512. The results from using this model can be seen in the Results section.

4.2 Hyperparameter Tuning

As aforementioned, I performed hyperparameter tuning to optimize my model. The hyperparameters that I tuned were the batch size, number of units per LSTM layer, timestep for backpropagation, number of LSTM layers, and number of training epochs. I put together some candidate values from the implementations online, blog posts and papers that I read, as well as a few candidate values of my own. To test, I held all other hyperparameters constant while testing the candidate values for a particular hyperparameter. After the tests finished, I looked at the outputted values and determined which value I thought was the best based on RMSE, complexity and running time. After a value was chosen, that would become the new constant for the hyperparameter in the following tests for other hyperparameters. The values from the experiments can be seen in the Results section.

5 Results

5.1 Hyperparameter Results

5.1.1 Timestep

During the testing, I found that as timestep increased, the RMSE also increased. This was particularly odd because one would imagine that having more days to see before making a prediction on the next day would provide more information to better predict. Because this went against common sense and because the RMSE was still rather low in the higher timestep candidates, I chose to go with 50 since it was in the middle of the pack and had a lower RMSE than 30.

5.1.2 Number of LSTM Layers

The general trend showed that as the layers increased, the RMSE also increased. However, at the end, it looked as if it might begin to reverse the trend. This is something to look into for the future. For the purposes of this project, I chose to go with 3 LSTM layers because even though 1 layer had a lower RMSE overall, the overall complexity of 1 layer is much less than for 3 layers and from some research, it seemed that it was possible that the 1 layer LSTM model may have just been echoing the data, rather than learning any hidden patterns in the data. Therefore, because the RMSE difference wasn't that big, I chose to go with a more complex model that still managed to have a reasonably low RMSE.

Constant Hyperparameters: batch_size=512, num_epochs=10, cell_state_size=30, num_layers=1										
Time Step	1	5	10	20	30	50	75	100	150	200
RMSE	15.6342	16.1263	17.8642	17.2667	19.3238	18.7191	20.1120	23.5684	22.0322	23.1279

Constant Hyperparameters: batch_size=512, num_epochs=10, timestep=100, cell_state_size=30, num_layers=1					
Number of Layers	[1, 30, 1]	[1, 30, 30, 1]	[1, 30, 30, 30, 1]	[1, 30, 30, 30, 30, 1]	[1, 30, 30, 30, 30, 30, 1]
RMSE	23.0648	24.5531	28.0774	40.5112	35.5278

Constant Hyperparameters: batch_size=512, timestep=100, cell_state_size=30, layers=3									
Number of Epochs	1	5	10	20	50	70	100	150	200
RMSE	38.9027	33.8085	29.1995	30.3862	16.9379	18.1954	22.4314	15.9138	17.6349

Constant Hyperparameters: batch_size=512, num_epochs=10, timestep=50, num_layers=1							
Cell State Size	1	5	10	20	30	50	100
RMSE	82.7455	22.5114	23.6805	21.2036	19.2802	19.0261	17.0686

Constant Hyperparameters: num_epochs=70, timestep=50, cell_state_size=20, num_layers=1				
Batch Size	64	128	256	512
RMSE	15.7528	18.3372	17.4127	19.8010

(a) Candidate values for hyperparameters

5.1.3 Number of Epochs

This was the hyperparameter that I tuned last. I simply ran the program with the chosen values for the other hyperparameters and picked chose the number of epochs by evaluating lowest RMSE as well as running time. Because the difference of RMSE between 50 and 150 was small, I chose to pick 50 due to significantly faster training.

5.1.4 Number of Units

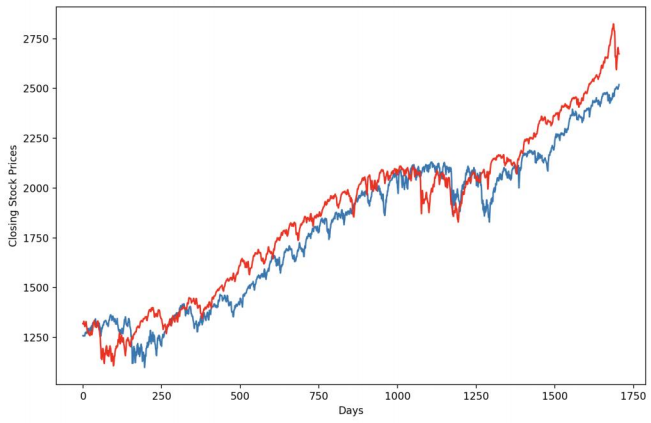
It was pretty obvious that by increasing the number of units, the RMSE would decrease. It seemed to become stable from 20 units onwards, so I chose to use 30 units in consideration of the running time as well as complexity.

5.1.5 Batch Size

From the experiments, the RMSE increased slightly as the batch size increased. However, since increasing the batch size affects the training time quite a bit, I chose to run with a batch size of 512.

5.2 Model Prediction Results

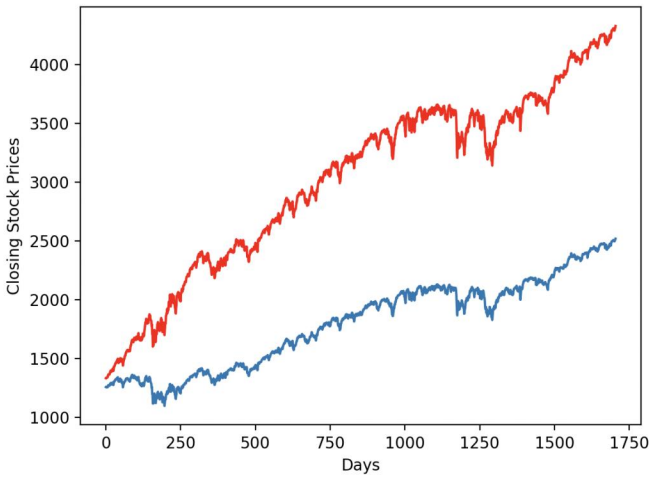
5.2.1 Day-by-Day Prediction



(a) Day-by-Day prediction (red=prediction, blue=orig)

The first of three predictions that I tried was the day-by-day prediction. This prediction takes a window of ground truth prices and predicts the next day's closing price. As can be seen in the graph, this prediction seemed to achieve quite a bit of success. The model seems to have learned something about the structure of the S&P500 graph.

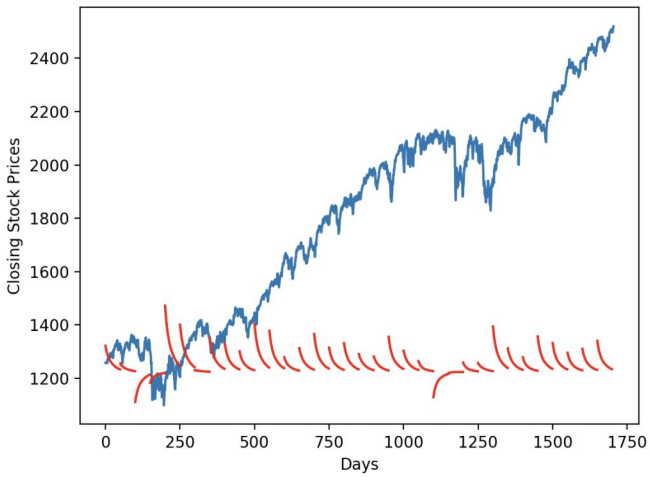
5.2.2 Whole Sequence Prediction



(a) Whole sequence prediction (red=prediction, blue=orig)

The second of three predictions that I tried was the whole sequence prediction. This prediction starts with a window of ground truth prices and predicts the next day's closing price. Then for the next prediction, the oldest price in the window is removed and that prediction is then appended to the end of the window to form the new window for the next day's prediction. The results from this prediction were mixed. On one hand, the prediction is rather far off from the ground truth values; however, on the other hand, the prediction seems to have learned some inherent information about the structure of the graph and predicted the trends well. I believe that perhaps errors that may have risen in the predictions were getting compounded since predictions were being used to make new predictions.

5.2.3 Tendency Prediction



(a) Tendency prediction (red=prediction, blue=orig)

This tendency prediction was trying to predict the trends across 50 days of closing prices. It used the same method as the whole sequence prediction, but was limited to a 50 day time span. At the end of the time span, it would start again with a ground truth window at that stopping point and continue forward for another 50 days. This achieved very poor results, as can be seen in the graph. It is predicting downward trends inn areas where it should be predicting upward trends. Once again, I believe that perhaps the issue here is the same as with the whole sequence prediction. Using predictions to make new predictions might be compounding small errors, leading to bad results.

6 Conclusion

From the results, the Day-by-Day prediction looked to be the best prediction model. It was decently accurate when coming to predicting the closing price for the next day. The Whole Sequence Prediction and Tendency Prediction models both may have suffered due to the same problem of compounding errors. If I had had more time, I would have liked to see if I could try a different approach to achieve better results, especially for the tendency prediction.

For future work, I would like to experiment with building a neural network (most likely another LSTM) to predict stock price movement based on news articles and integrate it with this model to see how much it would help performance for the day-by-day prediction. I would also like to make a framework to utilize my model in real-time.

References

[1] lilianweng. *stock-rnn*. https://github.com/lilianweng/stock-rnn

[2] Donald C. Wunsch, II, Danil V. Prokhorov, Emad W. Saad. *Comparative Study of Stock Trend Prediction Using Time Delay, Recurrent and Probabilistic Neural Networks*

[3] Akhter Mohiuddin Rather, Arun Agarwal, V.N. Sastry. *Recurrent neural network and a hybrid model for prediction of stock returns*

[4] Monica Adya, Fred Collopy. *How Effective are Neural Networks at Forecasting and Prediction? A Review and Evaluation*

[5] lucko515. *tesla-stocks-prediction* https://github.com/lucko515/tesla-stocks-prediction

Github Repository Link

https://github.com/FrankyJ578/LSTM-stock-prediction/

Contributions

I worked on this project alone.

Thanks for a great quarter! I learned a lot and had a lot of fun!