Matthew Brown (meb180001), Saman Laleh (sxl173130), Abed Mir (amm161930), John Kenney (jfk150030)

Using a Recurrent Neural Network to Predict the Closing Price of SPY

Abstract

Predicting stock market trends and the prices for particular stocks has been a hot spot for individual investors and investment companies alike. The research on a stock's value and the prediction of stock market trends is not just theoretically significant but has many widespread applications. Obviously, hedge funds and Wall Street firms have become more acquainted with machine learning algorithms to predict future stock prices, but machine learning and recurrent neural networks is not something only the pros can use to their advantage. Several such as Renaissance Technologies, Rebellion Research, etc focus exclusively on the use of machine learning algorithms to generate profits.

The goal of this project is to use a recurrent neural network model to predict the future stock closing price of the ticker symbol SPY. The SPY exchange-traded fund is the equivalent of the S&P 500 index just packaged , a broad measure of the

overall health of the market, consisting of the largest 500 companies in U.S. the stock market. This study was conducted using the last 5 years of daily price data gathered from Yahoo Finance.

Introduction and Background

Predicting the price of a stock based on historical pricing data, and more generally time-series forecasting is not well suited for a traditional neural network due to the sequential nature of the data. Recurrent neural networks were born out of the need to process longer data sequences that are not well suited for a traditional neural network.

A Study of the Main Algorithm

To start off, just like any other successful Machine Learning project, we split our data into two portions; the bigger portion was used as training data for the RNN and the rest was used as test data. With this split, we are able to have the ingredients to make a better prediction of the stock prices at any given time given the independent variables.

Put simply, the main algorithm of our RNN can be split into three rather simple functions, forward propogation, backpropagation, and the weight update.

The forward propagation method was implemented using the following equations for calculating the next hidden state h_t and the next output y_t from the next input x_t :

$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

$$y_t = W_{hy}h_t + b_y + S(x)$$

The S is the sigmoid function and W is the weight used.

(Reference is https://victorzhou.com/blog/intro-to-rnns/)

The second part of our algorithm consisted of the backpropagation algorithm. However, this version of the backpropagation algorithm is slightly different and its called the backpropagation through time (BPTT). It is mainly used for a certain variation of RNN's called RNN with time series or long short term memory's (LSTM's). This is considered to be the ideal version of the backpropagation for RNN dealing with stocks because it deals with making future predictions based on time-based data. The BPTT that we used was implemented using the following functions in mind: the derivative of the loss function, derivative of weights u, w, and v. For the BPTT we would need to find the following three equations along with the appropriate loss function (MSE):

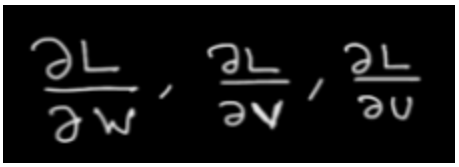
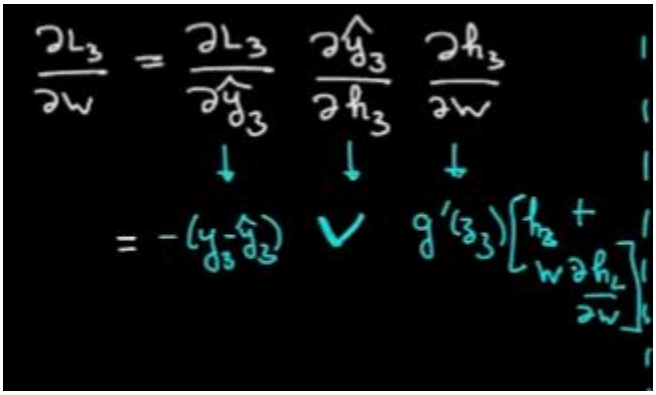
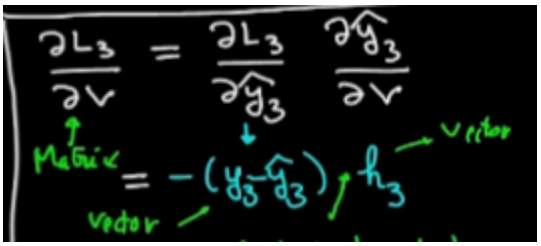
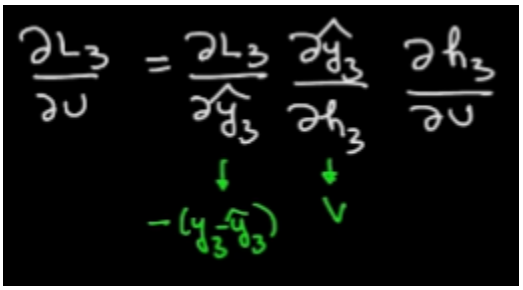


Figure 1: The three main functions we will need to create the backprop algorithm (for reference see https://www.youtube.com/watch?v=RrB605Mbpic&ab_channel=NPTEL-NOCIITM)

To calculate each of the functions given respectively, the equations should be modified as such:







(Reference for pictures above is

https://www.youtube.com/watch?v=RrB605Mbpic&ab_channel=NPTEL-NOCIITM)

The third and final main method we called was the update weight function. Being much less complicated than the previous two functions, this one is only responsible for calculating the new weights based on the learning rate η and previous weights u, v, w, bias output “bo”, and bias hidden “bh”. It also commonly known as stochastic gradient descent. Below is the function used as reference:

$$v_{new} = v_{old} - \eta \frac{dL}{dw}$$

(see reference: https://victorzhou.com/blog/intro-to-rnns/)

The w in this function is the weight being used and L is the Loss function.

Visual Results and Analysis

A small visual plot of the data using 100 iterations, where the blue line represents true stock prices, and the red line represents the prediction of the RNN. Predicting the stock price worked best under the following conditions for parameters: learning rate = 0.5, max iterations = 100, hidden layers = 5, timestep = 5. More summary data from trials are located in log.txt.

Table Results of Trial #1

timestep	Learning rate	max iterations	hidden layers	MSE
5	0.5	100	5	0.0146747

Table Results of Trial #2

timestep	Learning rate	max iterations	hidden layers	MSE
5	0.5	200	5	0.0230099

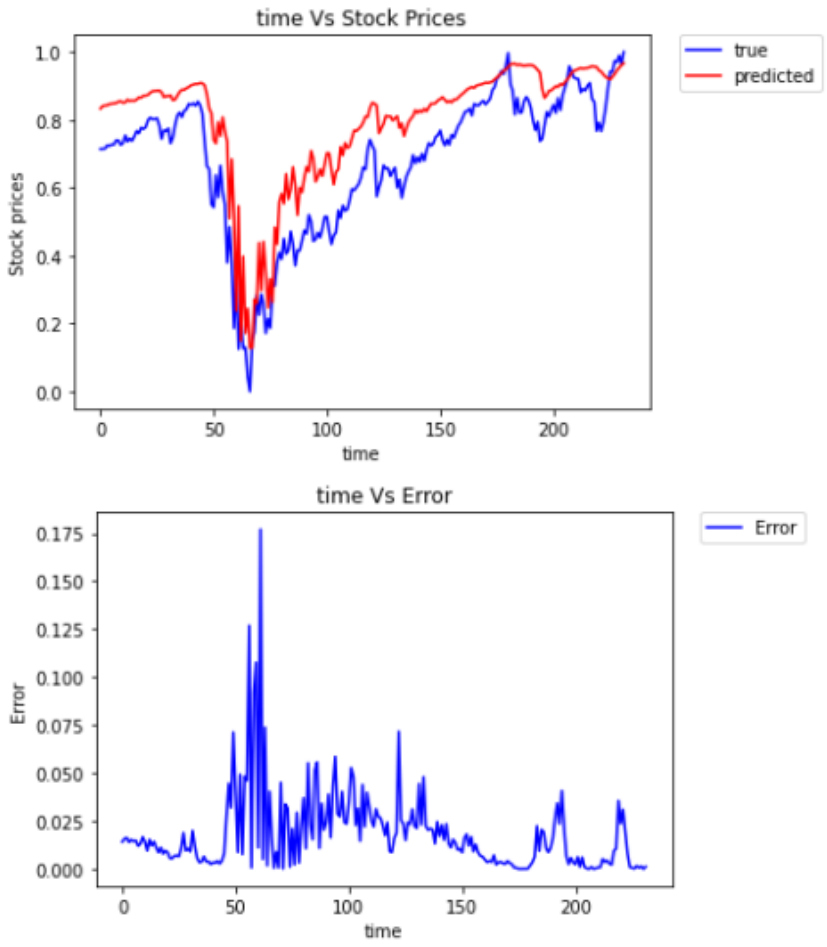
Table Results of Trial #3

timestep	Learning rate	max iterations	hidden layers	MSE
20	0.5	100	5	0.0180546

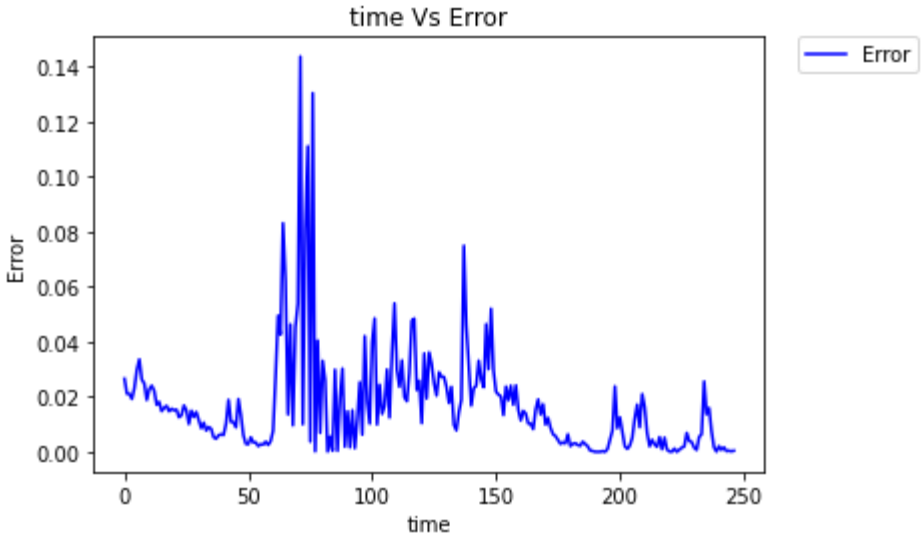
Graph of Trial #1

A second run where we see a slight improvement in our predicted values.

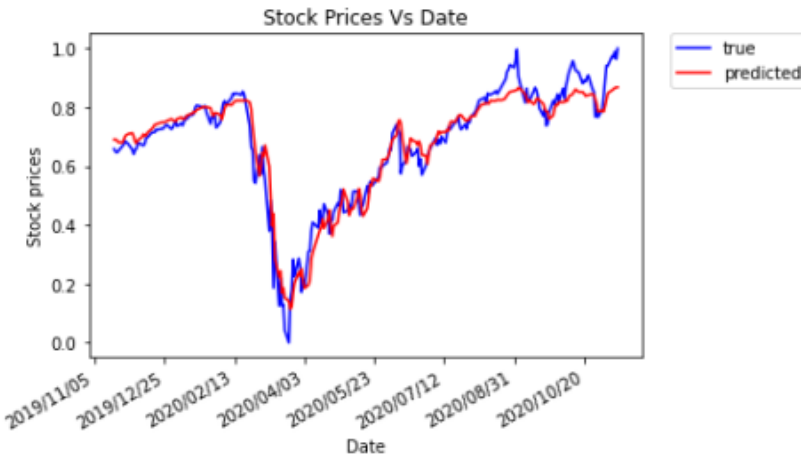
Graph of Trial #2

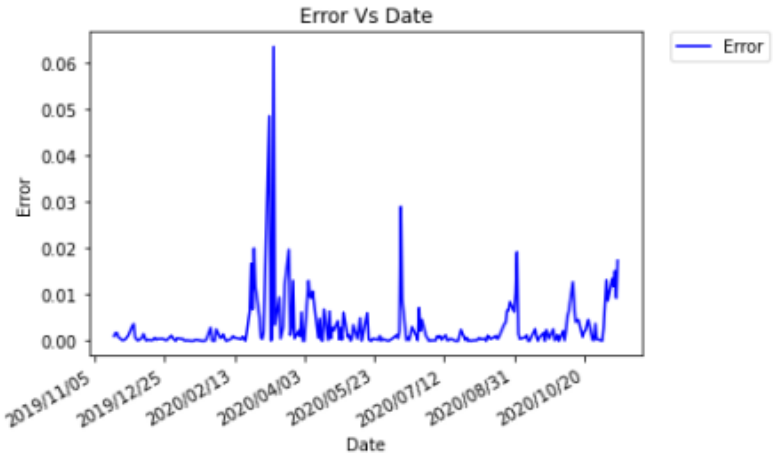


A plot of time vs error


time Vs Error
Error
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
50
100
150
200
250
time
Error

Last Trial





timestep	Learning rate	max iterations	hidden layers	MSE
2	0.001	150	5	0.0030251

Conclusion and Future Work

We found that running each iteration with print statements included made it seem the program was stuck in an infinite loop, although this was not the case. Using 100 iterations seemed to produce output that reduced the margin of error between the true prices and our predicted prices.

We realized that he who controls the stock market can own all the money in the world. So maybe it's not possible to predict all of the ups and downs of the stock market to get every penny of profits, but using a tool like an RNN with time series can certainly help decrease the risks of investing in the stock market for future investors.

References

All references are web-based resources. This includes various GitHub links, blogs, articles, and online textbooks. Below is a list of all resources used to complete this project:

https://www.mygreatlearning.com/blog/backpropagation-through-time/

https://www.deeplearningbook.org/contents/rnn.html

https://medium.com/@SeoJaeDuk/only-numpy-vanilla-recurrent-neural-network-with-activation-deriving-back-propagation-through-time-4110964a9316

https://machinelearningmastery.com/softmax-activation-function-with-python/

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-2-implementing-a-language-model-rnn-with-python-numpy-and-theano/

https://datascienceplus.com/long-short-term-memory-lstm-and-how-to-implement-lstm-using-python/

https://www.freecodecamp.org/news/the-ultimate-guide-to-recurrent-neural-networks-in-python/

https://towardsdatascience.com/time-series-forecasting-with-rnns-ff22683bbbb0

https://victorzhou.com/blog/intro-to-rnns/

https://www.youtube.com/watch?v=RrB605Mbpic

https://towardsdatascience.com/all-you-need-to-know-about-rnns-e514f0b00c7c

https://stackoverflow.com/questions/9627686/plotting-dates-on-the-x-axis-with-pythons-matplotlib