Implementation of stock prediction model based on LSTM

Wei Huanzhe William
Data Science
n830026116@mail.uic.edu.cn

Zhang Shichen Harry
Data Science
n830026154@mail.uic.edu.cn

He Qirui Steven
Data Science
n830026032@mail.uic.edu.cn

Chen Xingming Carl
Data Science
n830026012@mail.uic.edu.cn

Date: 2021-06-01 (group11)

***Abstract*--Stock, as a kind of securities, attract the huge demanding of investment. As an investment form, how to make sure to have a high return is a question that bothers investors. In this project, to find a way to solve this question, we try to re-implement a stock prediction model. Stock prediction is a time series problem, thus we will use the Recurrent Neural Network model to deal with it. Firstly, we will go through the theory of Recurrent Neural Network, improved with Long Short-Term Memory. After building and training our model, we find the result of LSTM return a good result.**

***Index Terms* -- Stock Market Prediction, Deep Learning, RNN, LSTM**

Background and motivation

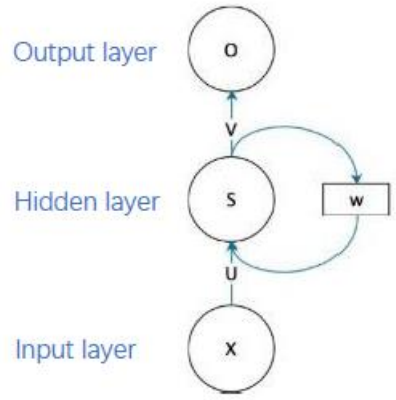
Nowadays, more and more people join in the stock market to pursue great opportunity for chasing fortune. In order to maximize the profit in the stock market, stock forecasting becomes a very popular area for not only the economist and the businessman, but also for the data scientists and people who expertise in artificial intelligence and machine learning. However, because of the volatility of the stock market is very great, and it is possible to have large fluctuations at any time due to some new policies from the government or other reasons. It is difficult for both natural investors and businessman to make a profit from the stock market. In this general background, as data science students, we want to use the existing deep learning model and algorithm to predict the stock price, so that for the investors to have a better choice when buying the stocks.

Theoretically speaking, stock prices are predictable, but there are a large number of factors that affect stock prices, and so far, the factors which affect the stock prices are not clearly defined. This is because the stock forecast is highly nonlinear, which requires the forecast model to be able to deal with non-linear problems. And the stock's data is time series data. So, we use recurrent neural network (RNN) to deal with the stock price prediction. However, RNN models are weak in describing the time series data with long memory. So, we implement the Long Short-Term Memory (LSTM) model which is based on the RNN models to avoid this problem and achieve a better result when predicting stock prices.

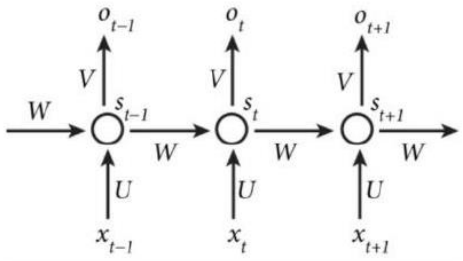
Methodology

A. Recurrent Neural Network(RNN)

For stock prediction is highly nonlinearity, it means that the model we use for prediction must have the ability to dealing with nonlinear problem. Also, for stock has the character of time series, RNN is an applicable model. The RNN model can be drawn like:



With the hidden layer can unfold to:



RNN allow the persistence of information, in the traditional RNN model, the weight w is the same, every time when input through the same cell, the input memory will be kept, also with another input need to be predicted, so in fact the prediction contains all of the previous memories and this time input.

Let's write formulas to show it, for example, we input x_t, the value of the hidden layer is s_t, the output is o_t,

$$o_t = g(Vs_t) \quad (1)$$
$$s_t = f(Ux_t + Ws_{t-1}) \quad (2)$$

(1) is the calculating formula for the output layer, output layer is a fully connected layer. V is the weight matrix of the output layer, g is the activation function. (2) is the calculating formula for the hidden layer, hidden layer is a circulation layer. U is the weight matrix of the input x, W is the is the weight matrix of the last time input s_{t-1} input in this time calculate, the activation function f(x) we use here is ***tanh*** function. From the formulas we can see, the circulation layer has one more weight matrix W.

So, if we input formula (2) into (1) and make a recursion, we have:

$$o_t = g(Vs_t)$$
$$= Vf(Ux_t + Ws_{t-1})$$
$$= Vf(Ux_t + Wf(Ux_{t-1} + Ws_{t-2}))$$
$$= Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Ws_{t-3})))$$
$$= Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Wf(Ux_{t-3} + \ldots))))$$

For the RNN we will to use actually is the Bidirectional cyclic neural networks, for we need to do the back propagation, we also need to add a formula that:

$$s'_t = f(U'x_t + W's'_{t+1}) \quad (3)$$

and o_t become:

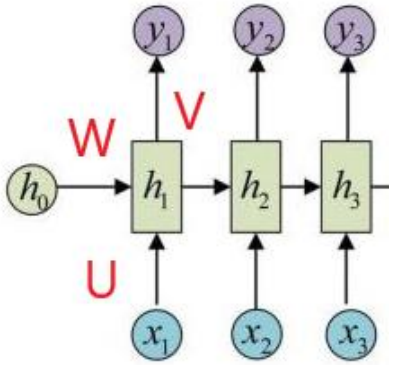
$$o_t = g(Vs_t + V's'_t)$$

For the RNN, when the W>1, the error will become larger and larger. When W<1, the error become smaller and smaller, causes the gradient to 0. It means RNN model is too forgetful. Thus, we need LSTM model to deal with this problem.

B. Long Short-term memory (LSTM) model

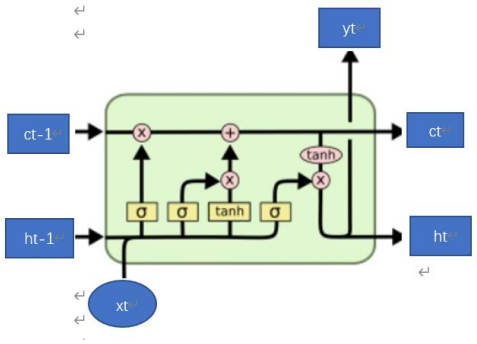
From the above methodology of the RNN model, we can see that there are still some problems when we implement for our stock prices prediction. Since we use a very large dataset which includes 10 years data for training the Recurrent Neural Network, the general RNN model is weak in describing the time series data with long memory. It becomes very difficult to train RNN because of the phenomenon of gradient dissipation and gradient explosion. The Long Short-Term Memory (LSTM) model proposed by Hochireiter and Schmidhuber was modified on the basis of RNN structure, so as to solve the problem that RNN model could not describe the long-term Memory of time series. LSTM is actually

a kind of RNN model. It is not a totally different the model from RNN. The activation function for the LSTM is the same as RNN. We choose to use ***tanh*** activation function. x_1, x_2, x_3 are the input from each of the three time. h_1, h_2, h_3 are the hidden layer. And y_1, y, y_3 are the output. V, U, W is the parameter which control the weight from input x1 to y1.



$$h_t = \tanh\left(x_t U + h_{t-1} W\right)$$
$$y_t = \text{softemax}\left(h_t V\right)$$

We can see that the output of RNN at each time is directly a ***tanh*** nonlinear transformation combining the output of the previous time with the input of the current time and outputs both the memorized and not memorized. However, for LSTM model, it will select the memory, so that it can remember the characteristics for a long time. In LSTM we introduce the concept of "cell state" and the mechanism of "gate".



Specifically, one neuron in the LSTM model contains one cell and three gate mechanisms. Cell is the key of LSTM model, which is similar to memory and is the memory space of the model. Cell states change over time, and the information recorded is determined and updated by the gate mechanism. The gate mechanism is the way to get the information selection through the sigmoid function and the dot product operation. The value of sigmoid is between 0 and 1. When sigmoid is 0, it means discarding information. When sigmoid is 1, it means complete transmission.

The LSTM has three gates to protect and control cell gate:

1. Forget gate
2. Update gate
3. Output gate

The process for LSTM output:

First, the forget gate will forgets some cell state:

$$f_t * c_{t-1} = \sigma(W_t, [\, h_{t-1}, x_t]) * c_{t-1}$$

h_{t-1} is the output of the moment, x_t is t input of this layer, W_t the weight of each variable, b is biased, σ is the sigmoid function, form is:

σ (x) $=(1 + e^{-x})^{-1}$, $f_t * c_{t-1}$ is between 0 ~ 1, said the output for each cell in a state of the values in c_{t-1} is 1 means "keep all", 0 means "completely abandon".

Secondly, update gate includes input the stock prices attribute and update the cell state for each propagation to get the final results.

$$i_t * c_t = \sigma(W_i, [\, h_{t-1}, x_t]) * \tanh\left(Wi, [\, x_t, h_{t-1}]\right)$$

Where ***tanh*** is the activation function when update the information in the hidden layer.

After input the information, we have to update the cell state for each layer:

$$c_t = f_t * c_{t-1} + i_t * c_t$$

This formula actually means adding the input and the information we want to forget through the forget gate is the update cell we want to output.

Finally, we have the output gate to output the variables that we want:

$$h_t = o_t * \tanh\left(c_t\right)$$

The output will be based on the state of our cells, that is, a filtered version. Sigmoid function is used to determine how much output information (the state of the cells which part prints out), using ***tanh*** functions withc_t(get a value between -1 to 1).

According to the mechanism of the three control gates, we finish process one neurons LSTM process. It makes LSTM model can form a long-term memory. And it also makes the model output value and the real value in a selected minimized loss function.

This mechanism successfully avoids the continued product of the gradient in each of the cell update. So, it avoids the RNN model's problems which are the gradient explosion or gradient disoperation.

For our stock prices prediction, we input all the features which is the factors that influence the stock price into the model, and we wish it will output the future range of the stock prices. And for training the LSTM model, as long as we have 10 years' time series data. We will use back propagation to train the network. And reduce the loss of the function. It will be much more efficient than RNN model. The result and the input feature are based on the financial knowledge, and the model is based on the deep learning algorithm. Combining this two knowledge, we hope it will help people who work in business will get a better prediction result.

Data collection and preprocessing

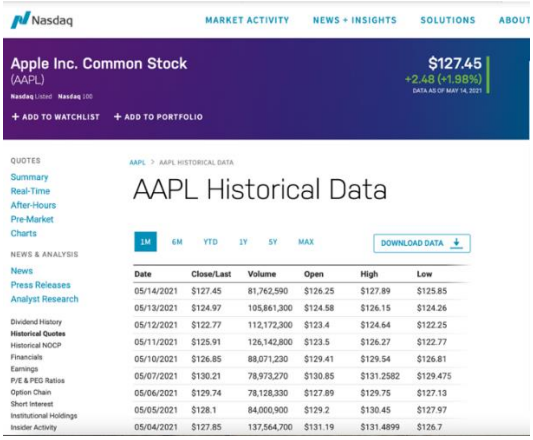
Since we are using the RNN and LSTM Networks to predict the stock data, we should first collect the data of one exact stock. We first explore some of the stock investing websites to decide which stock we want to analyze.

Here we first visit the *Investing.com*: https://cn.investing.com and found an interesting new which indicates that Apple is bearish since the amount of increase of Apple just surpass 1% in the first half year. However the comprehensive index of NASDAQ (https://www.nasdaq.com/) surpass almost 7% at the same period.

(The link of the new: https://cn.investing.com/analysis/article-200463538)

Hence we decide to collect the stock data of Apple and use Python to help us predict the trend of it.

To make sure that we collect the precise stock data of Apple, we choose to collect the data from NASDAQ (https://www.nasdaq.com/), one of the biggest stock market of the world.



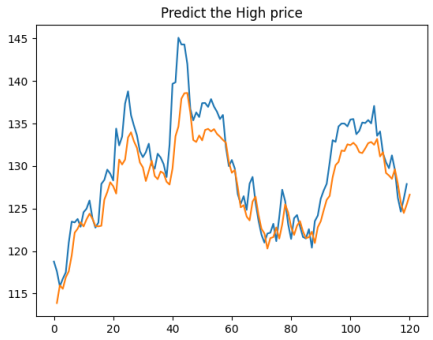
(The link of the page: https://www.nasdaq.com/market-activity/stocks/aapl/historical)

We choose the historical quotes of Apple and download the data from 2011-05-16 to 2021-05-13 as a file called 'data.csv'.

After downloading the stock data of Apple, we

preprocess the data by using Excel for the convenience of data analyzing. Here we remove all the '$' symbol in front of the numbers and then sort the data in ascending order according to the 'Date' attribute. Also after checking the result of our prediction we find it not ideal thus we imply feature mapping by adding one more attribute 'Change' to the data which represent the changing rate of the closing price in one day in order to achieve a higher degree of fitting.

$$Change = \frac{Closing\ price_{previous} - Closing\ price_{following}}{Closing\ price_{following}}$$



Before adding attribute 'Change'

After adding attribute 'Change'

Obviously we can see from the figures that after adding the attribute 'Change', the degree of fitting increase a lot.

Hence the integrated attribute of our data set is: Date, Close(closing price), Volume(volume of business), Open(opening price), High(the highest price in one day), Low(the lowest price in one day) and Change. There are totally 2516 rows of data, each one provides the above attribute of Apple's stock of an exact day.

Furthermore, we also use some mathematical methods to preprocess the data for the revivification of data in the later process, including calculating the mean and the standard deviation of the data and also do normalization to the data so that the data is unified to the specified interval.

$$mean = \frac{\sum_{i=0}^{n} X_i}{n}$$

$$standard\ deviation = \sqrt{\frac{(X_i - mean)^2}{n}}$$

$$normalization = \frac{X_i - X_{min}}{X_{max} - X_{min}}$$

In our program, 95% of the data are used as training set and 5% of the data are used as testing set. Also, we use 15% of the data for validation.

Model Evaluation

We choose to use the Mean Square Error Loss to evaluate our model. The Mean Square Error Loss evaluates the difference between the predicted value and the real value, the smaller the loss, the better the model. The formula of MSE loss is shown below.

$$Loss = \sum_{i=1}^{n} (Y - f(x))$$

$$MSE = \frac{1}{N} \sum_{i=1}^{n} (Y - f(x))$$

Determination of some hyper-parameters

Among countless parameters of the model, we think there are three hyper-parameters that will affect the accuracy of the model, which are the learning rate, epoch, and the time steps. The Time Steps parameter means that Time Steps of previous days' data will be used to predict the next day. So, we just did a

comparison between different values of the hyper-parameters on the prediction of the highest value of test set.

For epoch=10:

Mean Square Error		Time Steps			
		1	10	20	30
Learning Rate	0.0001	3.13795874	0.20286099	0.35923273	0.41193145
	0.001	4.11310577	0.00520201	0.01321076	0.02509 08
	0.01	5.00923957	0.00624515	0.00315741	0.00358249

For epoch=20:

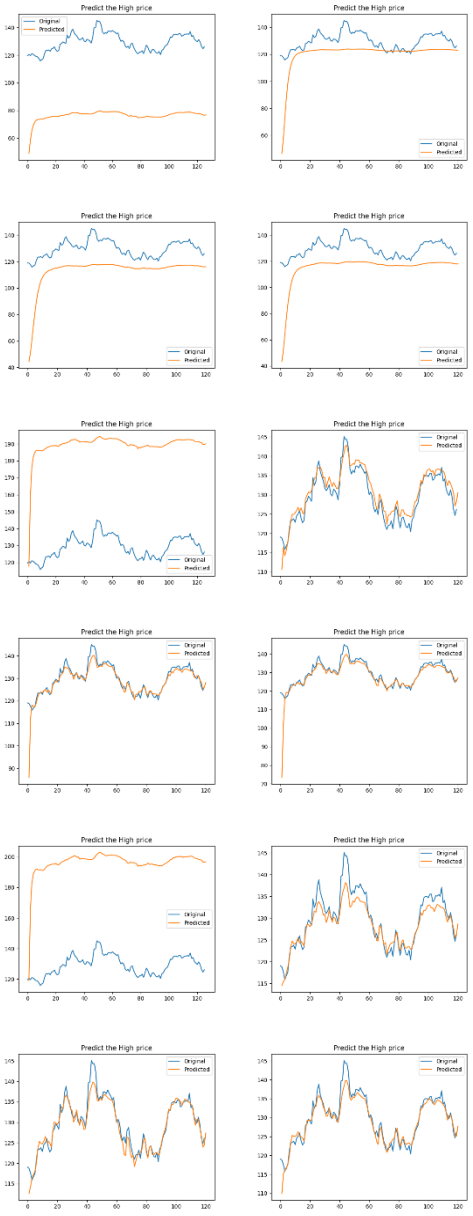
Mean Square Error		Time Steps			
		1	10	20	30
Learning Rate	0.0001	3.07704324	0.2633796	0.09486823	0.13692171
	0.001	7.80486323	0.00214969	0.00273071	0.00378769
	0.01	3.12264988	0.0155339	0.00271655	0.00595005

For epoch=50:

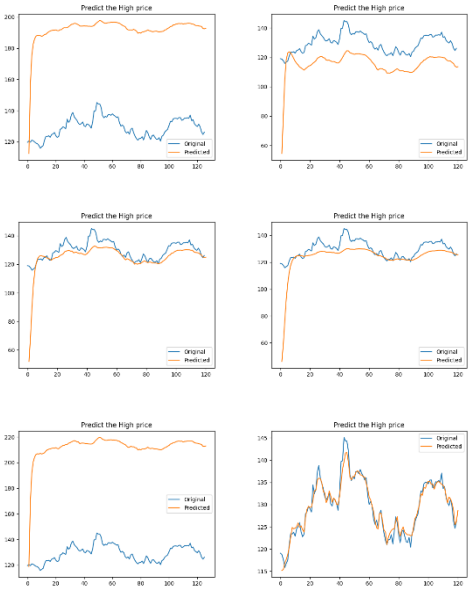
Mean Square Error		Time Steps			
		1	10	20	30
Learning Rate	0.0001	3.07704324	0.2633796	0.09486823	0.03826791
	0.001	6.75292623	0.00333377	0.00199777	0.00223447
	0.01	4.48748078	0.01675641	0.00313391	0.00522151

We can also take a look at the fitting degree of different parameters.

For epoch=10:



For epoch=20:



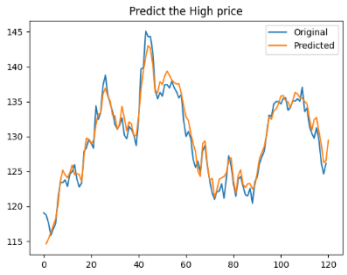
For epoch=50:

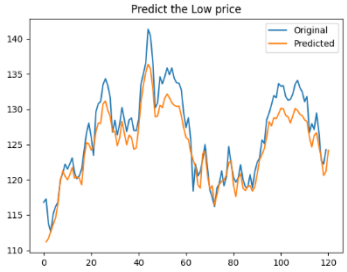
Therefore, we choose epoch=50, learning rate=0.001 and Time Steps=20 as appropriate values of the two parameters.

Result Analysis

Now we have determined some appropriate hyper-parameters, we can use the model to make prediction on the next day's stock. The result is shown below.

```
Epoch 50/50
The train loss is 0.000705. The valid loss is 0.000320.
The mean squared error of stock ['High', 'Low'] is [0.00202335 0.006975  ]
The predicted  High price for the next 1 day(s) is: 129.44873954591893
The predicted  Low price for the next 1 day(s) is: 124.17486553267028
```





We can see that the loss of training set and validation set is very low, the mean squared error of test set is also very low, and the degree of fitting on the test set is very high. Therefore, we can make a conclusion that our model is usable, and we can use it to predict the stock price.

Drawbacks:

The number of features of our data is too small, thus the prediction result is not suitable for real situations, our prediction is just for reference, other market-influencing factors have to be considered while making an investment.

Future improvement:

Due to the limitation of our data collection, the number of features of our data is too small, thus in the future we may try to increase the amount of the features of our data to make our model more practical.

References

[1]魏宇.(2010).沪深 300 股指期货的波动率预测模型研究. *管理科学学报*(02),66-76. doi:CNKI:SUN:JCYJ.0.2010-02-009.

[2]徐正国,张世英.(2004).调整"已实现"波动率与 GARCH 及 SV 模型对波动的预测能力的比较研究. *系统工程*(08),60-63. doi:CNKI:SUN:GCXT.0.2004-08-015.

[3]Sepp Hochreiter,,Jü & rgen Schmidhuber.(1997).Long Short-Term Memory. *Neural Computation*(8),. doi:10.1162/neco.1997.9.8.1735.

[4] 陈卫华.(2018).基于深度学习的上证综指波动率预测效果比较研究. *统计与信息论坛*(05),99-106. doi:CNKI:SUN:TJLT.0.2018-05-015.

Appendix:

Code of our model:
https://github.com/williamuic/FinancialComputingGroup11