Stock Prediction- IBM

Group 3

Tianjian Cai, xin wen, XInran hao

2017

Contents

[1. Stock Selection 1](#_Toc500539334)

[2. Data augmentation 1](#_Toc500539335)

[3. HMM part 3](#_Toc500539336)

[3.1 HMM Problems 3](#_Toc500539337)

[3.1.1 Output of HMM is not continues 3](#_Toc500539338)

[3.1.2 HMM have only one output at each step 3](#_Toc500539339)

[3.1.3 Output of first-order HMM relates to the last hidden state only 3](#_Toc500539340)

[3.2 Model structure 3](#_Toc500539341)

[3.3 HMM training 4](#_Toc500539342)

[3.4 HMM predicting 4](#_Toc500539343)

[3.5 Result 4](#_Toc500539344)

[4. Neural Network part 6](#_Toc500539345)

[4.1 Basic neural network with different number of neurons 6](#_Toc500539346)

[4.2 Basic Neural Network with different dimensionality of input data 8](#_Toc500539347)

[4.3 Deeper neural network (10 hidden layers) 8](#_Toc500539348)

[4.4 Recursive Neural Network (RNN) 9](#_Toc500539349)

[4.5 Buying and Selling decisions 10](#_Toc500539350)

[Bonus part- Long short term memory(LSTM) and Rolling Window Linear Regression 11](#_Toc500539351)

# Stock Selection

In this project, we select IBM as target stock for analyzing. The trend of IBM has lots of up and downs (figure 1), which could avoid the risk of predicting increasing or decreasing would have satisfied accuracy like Google.

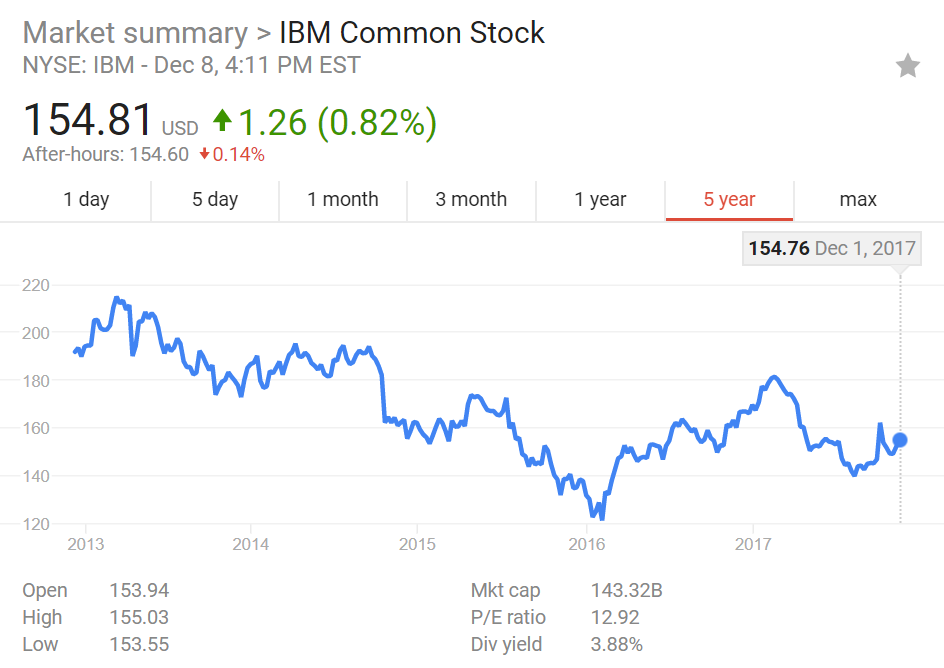


FIG. 1 5 years trend of IBM stock

# Data augmentation

In neural network, the calculation load is relatively abundant for thousands of input data, so more financial indicator is added besides the five original input arguments.

You will need the function dataAugmentation(dataframe) in the .ipynb file for data augmentation, and figure 2 shows the head of input and augmented data.

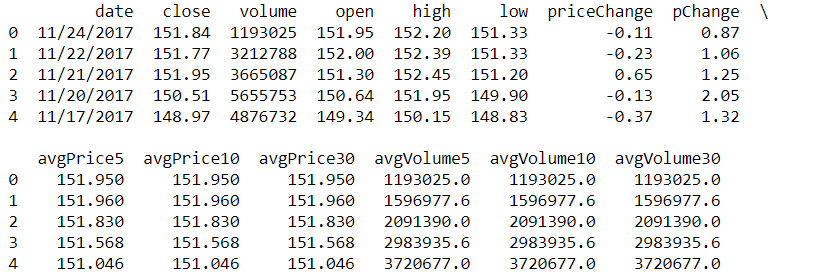


FIG. 2 Head of augmented data

In table 1, the functionality of the above parameters is described. We do not use professional financial indicators, because these indicators could also be derived from input file, and no information is added. Neural network is expected to formulate those functions, and the later conclusion indicated that 5 (column 2 to 6) or 13 (column 2 to 14) input argument do not improve the accuracy of the network.



TABLE 1 Description of input arguments

For fair training and comparison, in neural network part, we divide the 10 years price trend into 3 parts. So the comparison graph would have two break points, which means they belong to different time period.

# HMM part

## 3.1 HMM Problems

The mostly used first-order HMM cannot be directly used to predict the stock price. Here are the problems:

### 3.1.1 Output of HMM is not continues

The stock price, as well as the indicators, are discrete in time but continues in value. The HMM, although discrete in time, have discrete output values.

To solve this problem, the project quantifies (do the A/D transfer) the stock price and indicators to discrete values (e.g. 1,2,3,4…) so that they can be feed into HMM.

### 3.1.2 HMM have only one output at each step

People usually use many indicators to predict the stock price. However, the next step of HMM is only affected by the previous hidden states. What’s more, for each hidden state, it can only emit one observation. This means that the HMM can only be trained by a 1-D sequence.

Although there are many indicators, the indicators must be compressed to 1 value. Using the PCA to analysis the indicators, the rate of change (ROC) have the biggest weight. Therefore, in this project, choose ROC to be the training data of HMM.

### 3.1.3 Output of first-order HMM relates to the last hidden state only

Hidden state of the current state in first-order HMM can only be affected by last hidden state. This is not compatible to the stock. Today’s stock price may be affected by the stock price 10 days before, or even more days further.

As solving N-order HMM is too complex, we can reduce the order of HMM by regarding each hidden state as a vector of previous hidden states. It is kind of like convolutional code encoding.

## 3.2 Model structure

The whole model to predict stock price is shown in figure 3(Assuming each day only have 2 states: 0 and 1):

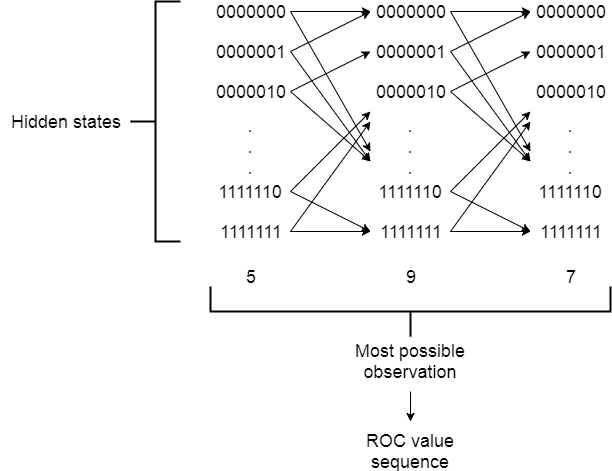


Fig. 3 Structure of HMM model

## 3.3 HMM training

Calculate the ROC value, then use it to train the above HMM structure to get A and B matrix

## 3.4 HMM predicting

Use A, B matrix and ROC sequence to do the decode, and get the hidden state sequence.

Use the last hidden state as the start state, do the HMM forward calculation, and predict the ROC values.

## 3.5 Result

The following figure shows the predicted stock price (yellow line) compares to the real value (red line). The blue line is training data.

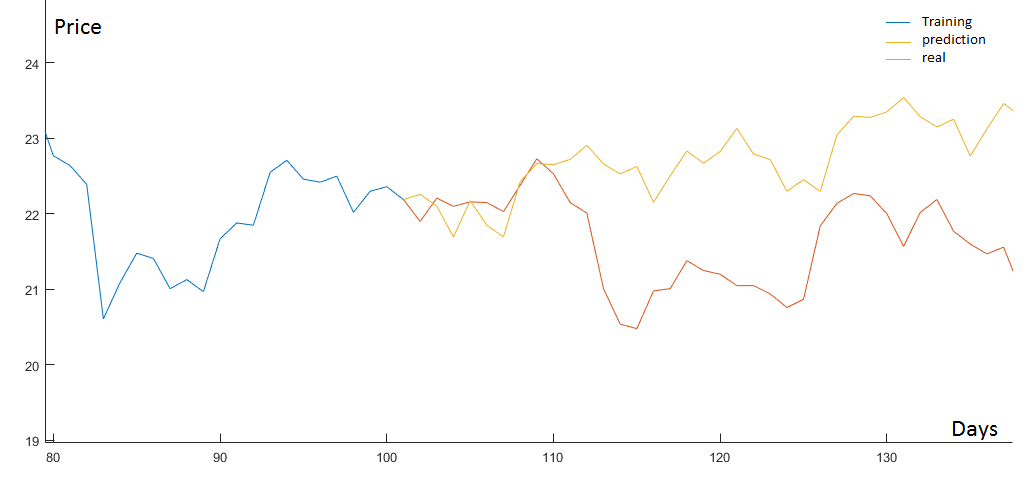


FIG. 4 Prediction and real prices. The blue curve is trained data (100 days), yellow curve is predicted data and red curve is real price of stocks.

The correlation is 0.1 (ranges from -1 to 1, 0 means no correlation).

The figure below shows a simplified version only predicts whether it is good to buy or sell. Correct rate of HMM prediction is 0.43, which confirmed the discussion in part 3.1.

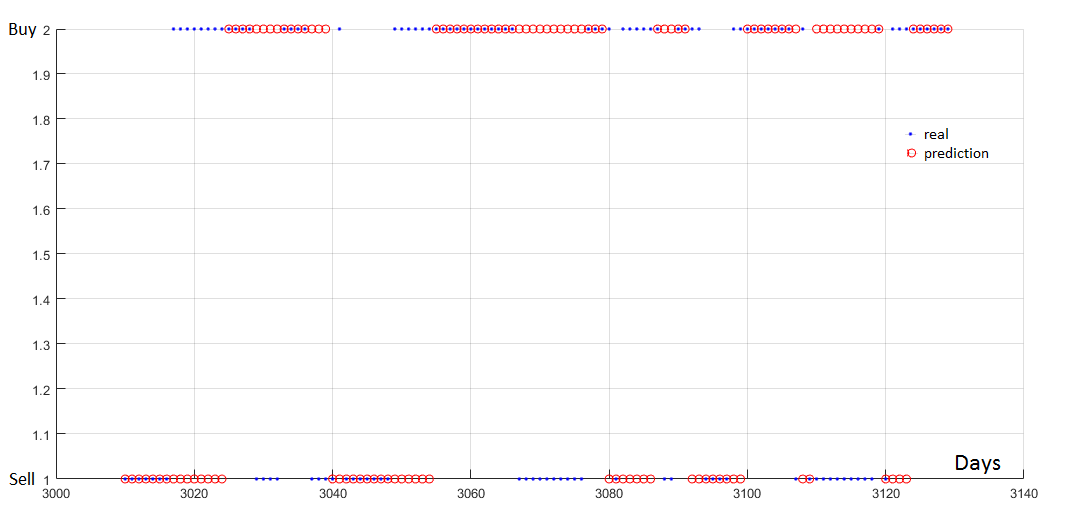


FIG. 5 Buy or Sell Prediction. Accuracy 0.43.

# Neural Network part

The complexity of neural network is much complex than HMM, and it’s expected to simulate any function with enough layers and hidden units. In order to compare the performance of different typical neural networks with different parameters, we chose 3 and 11 layers neural network and recursive neural network for comparison.

The input layer of neural network can have 5days or 13days neurons, depend on the number of input parameters and sliding window size. The output of neural network only have one units, indicates the opening price of the following day. And the buy or sell decision will be made based on a human set threshold.

Y label of trained or tested data is the next day opening price, so the neural network is expected to predict the opening price of the next day.

Different neural network could have different model function, but they shared the same training and testing function.

Because we are predicting the price, not the buy or sell decision, so the performance of different neural network is evaluated by mean square error, not accuracy in this part, but the final decision would be converted to the decisions once the best model is founded.

All the code of this part is done by python, inside the file Project-Stock+prediction.ipynb, based on jupyter notebook.

## 4.1 Basic neural network with different number of neurons

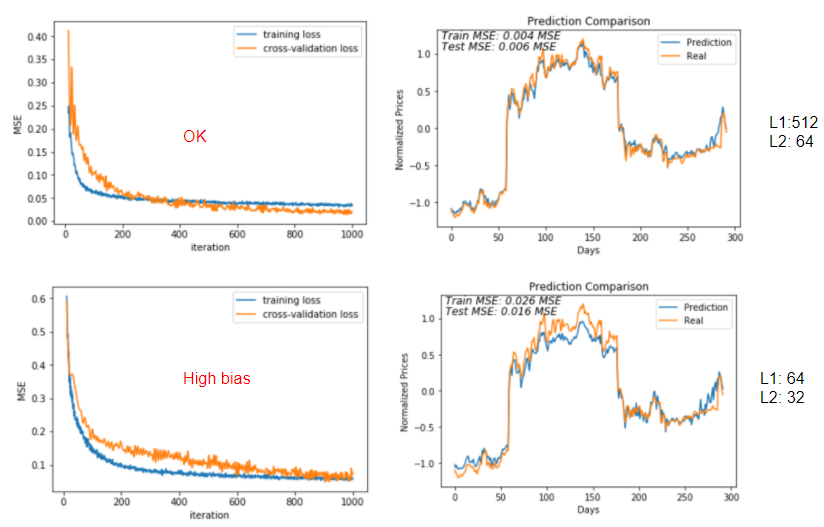
The function build\_model\_SNN(layers) in .ipynb file is for building basic neural networks. We start with using two-layer neural network with different neurons.

In this section, time sliding window is selected as 20.

Dropout is implemented to avoid overfitting. Because the neural network is simple, so the cross-validation data is expected to have lower MSE because some neurons is turned off in the training but all open in the cross validation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Test1 | Test2 | Test3 | Test4 |
| x1 | 512 | 64 | 256 | 256 |
| x2 | 64 | 32 | 256 | 32 |

TABLE 2 Two layers neural network with different neurons



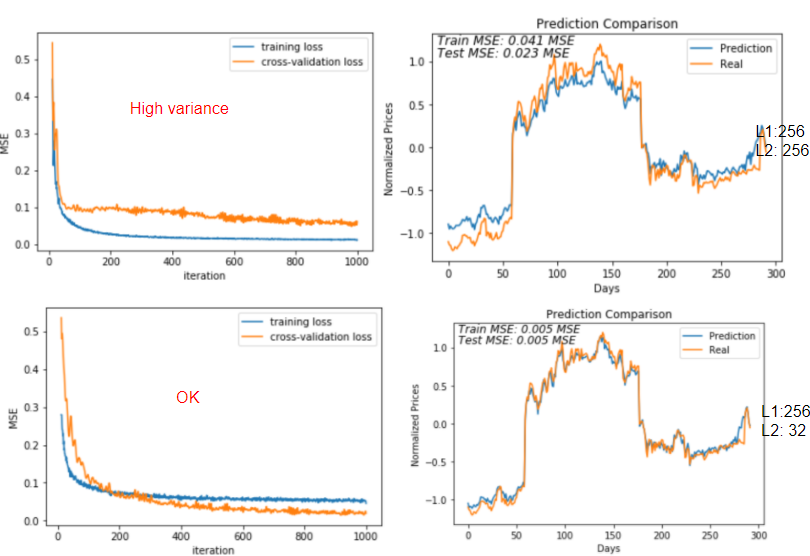


FIG. 6 Prediction of 2 layers neural network with different number of hidden neurons. The last set (layer 1 256 neurons, layer2 32 neurons) gives the best performance with MSE = 0.005 and no underfit or overfit

We selected 4 representative hidden neurons to show the difference, as Table 2 and Fig. 6 shows. The left column is the training performance and right column is mean square error of real and predicted data. The first and last set is good, but the last set has lower MSE because it has more neurons. The second set suffers high bias and third suffers high variance.

## 4.2 Basic Neural Network with different dimensionality of input data

In this section, we compared the performance of different input features (5 or 13) based on the best 2-layer neural network we just selected.

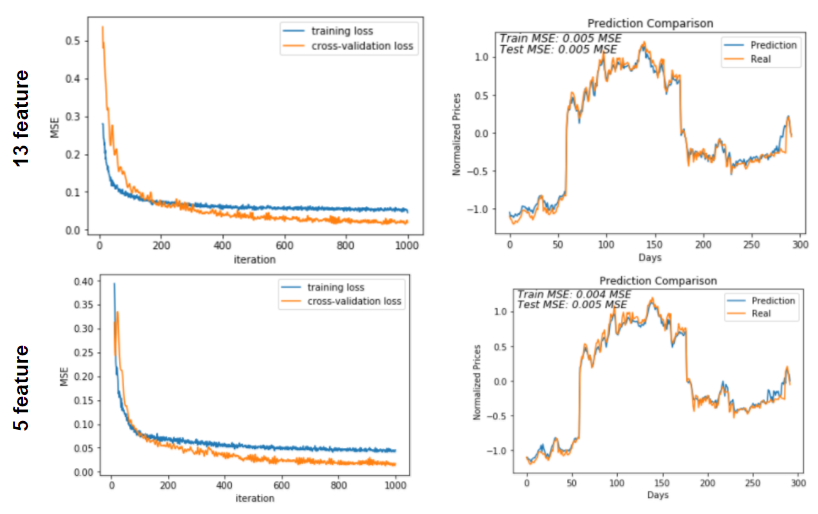


FIG. 7 Comparison of neural network with 5 or 13 input arguments

Based on Fig. 7, we can not see significant difference between different number of input features. It’s not hard to explain because the additional features are calculated by the 5 original features, so there is no additional information added to the model.

## 4.3 Deeper neural network (10 hidden layers)

At this time, we already get good performance on 2-layer neural network. We make an attempt to see what will happen in a deeper neural network. So, a 10-layer neural network is built as Fig. 8 shows. The top part of fig.8 is a schematic structure, and in the middle, it indicates the activation function and dropout rate. At last, the training curve and performance is presented.

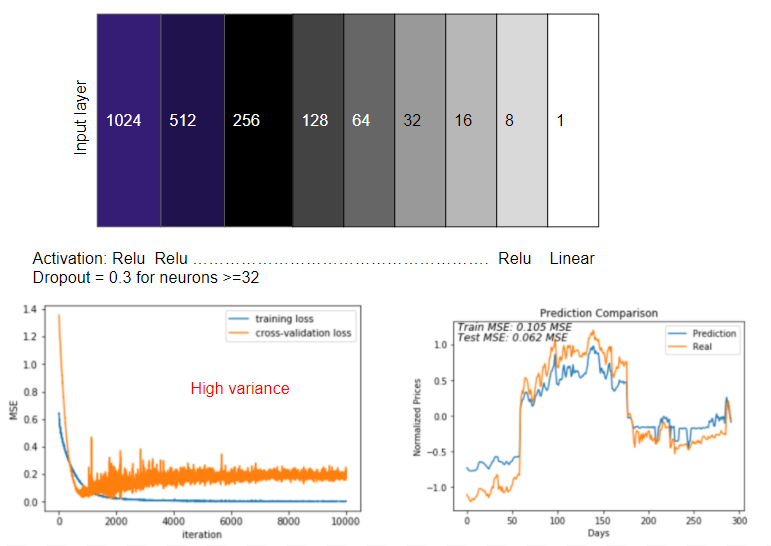


FIG. 8 Structure of performance of a 10-layer neural network. It overfitted

Because we only have around 2500 data for training and testing, the limited number of data is hard to handle different number of neurons, so high variance is observed.

## 4.4 Recursive Neural Network (RNN)

It’s known that stock price is highly dependent on previous trend, and RNN is a good model to handle history. So, we attempt RNN in this part. We use two-layer RNN structure with one hidden layer with 256 neurons, and one output layer with 1 neuron.

RNN is inside build\_model\_RNN(layers) function, and the performance is amazing- the MSE is less than 0.0001 (Fig. 9).

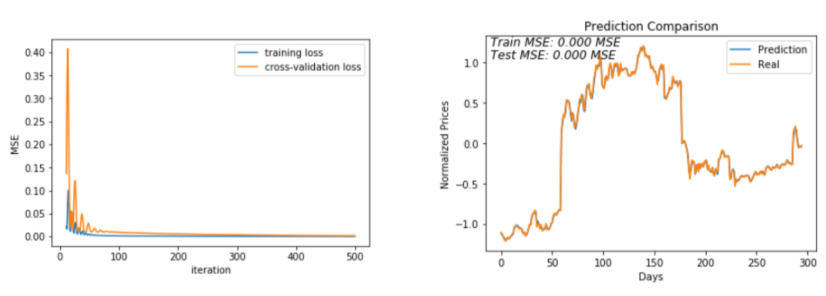


FIG. 9. Performance of RNN, which gives the best performance

## 4.5 Buying and Selling decisions

In this part, a two-layer RNN gives the best performance. Thus, we converted the predicted next day price into buying and selling decisions. The function is named decision(X\_test, pred, y\_test) in the bottom of ipynb file.

In this section, we set 2% as the threshold, price change larger than 2% would be considered as buy, less than 2% as sell, and between 2% as do nothing. The accuracy is 96.8%, which is satisfying.

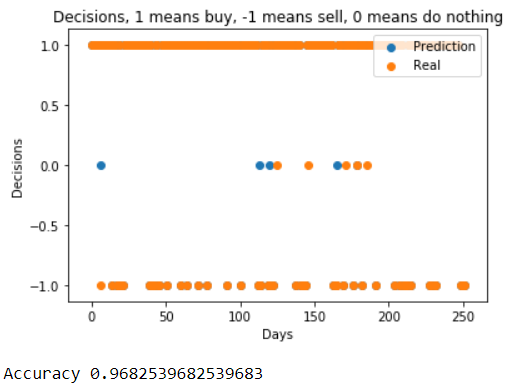


FIG. 10 Accuracy of decisions based on RNN model, 96.8 accuracy reached

# Bonus part- Long short term memory(LSTM) and Rolling Window Linear Regression

The last part is several attempts in advanced model (LSTM and RWLR). This additional part is done by Xiran thus may not follow the previous trend.

We used LSTM and Rolling Windows Linear Regression as two complementary methods to support the research of NN and HMM.

However, the more our group digging into these two models, the more we believe that LSTM and Regression models can have a better theoretical result.

NN and especially CNN is good at feature extraction. This means that if we give CNN a picture, CNN can extract the contour of the picture. However, we need to use RNN and LSTM for prediction because they are for sequence prediction and we all know stock is strongly related with sequence.

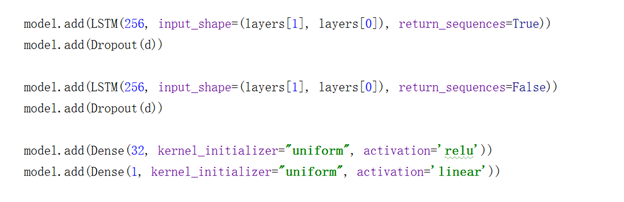
The reason that we choose LSTM but RNN is that RNN has simple structure. RNN has one tanh layer and two interactive layers, but LSTM has four tanh layers and four interactive layers. By using the memory cell, we can make sure that we don’t have vanishing gradient problem, which means we can have a good prediction.

Because LSTM has really a similar structure as CNN, we are not going to mention everything about the code. However, the procedures are below:

We used open, close max and min price for training our model.

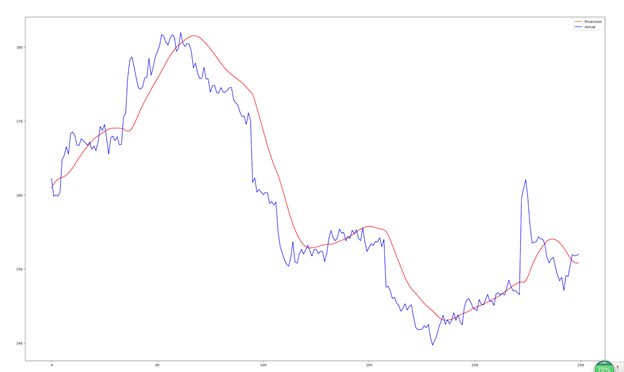
We have windows=22, which means we can satisfy the professor’s requirement. We use 22 days for training and 22 days for testing, and another 22 days for training and another 22 days for testing and so on so forth.

We load the data. Normalized the data to (-1, 1). Build the model with two LSTM layers and one fully-activation layer. Last, denormalized the data.



The score and the result are below:





From here, we can see that the predicted value (red) and actual value (blue) are similar. The predicted value can reflect the actual value’s trend. However, there are some kind of delay in time, which is a minor in this prediction. However, we use only epoch=30, if the epoch changed to 100, the problem can be compromised. I am sorry I only have a cpu but gpu.

Rolling Windows Linear Regression can be built by an Office Excel. Therefore, the code is not hard. Like the definition of the rolling windows, we use windows=22 days for training, and use another windows=22 days for training, and so on so forth.

The procedures are below:

We used open, close, max and min price for training.

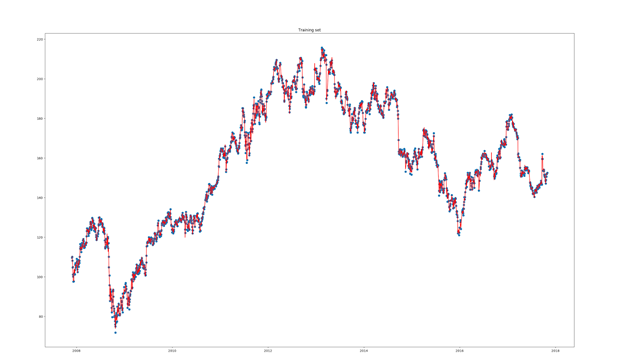
We used open, max and min prices for matrix X, which is a matrix of feature and bias.

We used close prices for matrix Y, actual output value.

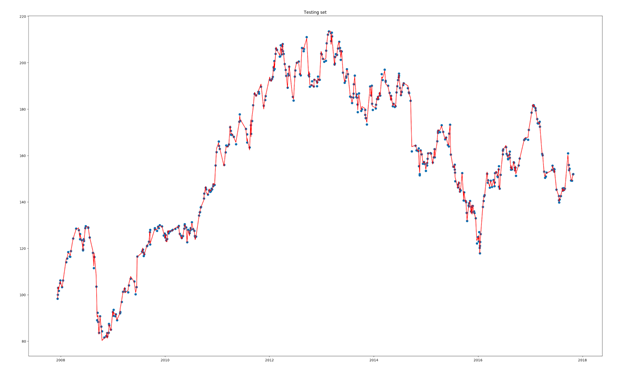
After using X and Y for building up training and testing data, we calculated the weights from training set and used that for testing set.

The result is shown below:

R\_squared is nearly 1, so you can see that the actual value and the predicted value are closely related.



In training set, the randomly chosen actual values are crossed by predicted value. Nearly every points is crossed.



In addition, in testing set, the randomly chosen actual values are crossed by predicted value.

Overall, theoretically, LSTM can predict the trend of the actual stock price in a descent way. So is the linear regression. However, we don’t know for sure why Linear Regression can converged in so short a time.