Predict Stock Price by LSTM Model

National Taiwan Normal University 110-2 Neuron Network,

Assignment 1

Chia-Hao, Chiang
NTNU CSIE Data Visualization Lab
Taipei, R.O.C
61047061s@gapps.ntnu.edu.tw

***Abstract*—Predictions are expected to be powerful and useful in many aspects, like medical, finance, or environment changes. Our goal is to increase the precision and lengthen the time interval, but there are still restrictions or obstacles like data collecting and the dropping performance as time scale rockets.**

In this paper, we are given a historical dataset in order to get the prediction in stock price, as a practice. Here, I train a model, using Long Short-Term Memory, estimating future price in weeks. Eventually, I observe the influences of epochs and batch sizes in performance.

Keywords—machine learning, long short term memory, training.

I. Introduction

As being in the global village era, the connections among countries become closer and more interactive. In an economics speaking, or the financial stream, the investment overseas or the supply chain plays a significant part in one's industry. Therefore, we take actions analyzing such funds flow or build a model to either simulate the impacts on one's economy. On the other hand, the individual is not likely able to access enormous financial data. Then how do we evaluate the investment, for example, stocks in market? One of the most convenient is to use a predicting model to help one making decisions in a favor of reliability of machine learning.

II. Methodology

A. Dataset

We take the history price of the stock, Lotus, from 2012/3/27 to 2022/3/25, including four attributes: open price, high price, low price, close price, and volume. In this paper, I considered only the close price for the model training, and expect to perform the predictions in 2022/3/26 to 2022/4/1.

B. LSTM Architecture

Long Short-Term Memory, LSTM[1] is proven to be useful in sequence prediction problems. It stores important historical data and drops the one is not significant. Here, it's designed to have 3 hidden layers, and it takes 60-day-history close price data and outputs one predicted stock price sequentially. Eventually, separate dataset into training and test data by the ratio 8:2. The activation function uses default one, hyperbolic tangent.

C. Cost Function

We take root mean square error, RMSE, for short, in this case since the problem is more like a regression problem.

$$RMSE = \frac{1}{N}\sqrt{\sum_{i=1}^{N}(\hat{y}_i - y_t)^2} \quad (1)$$

III. Experiment and Analysis

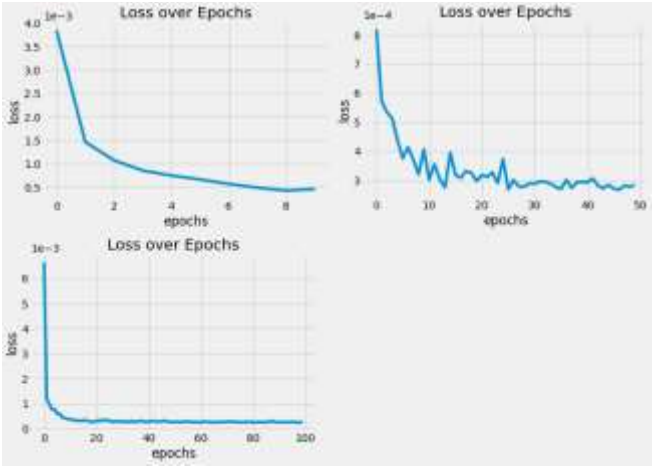
Experiments on different batch size and epochs are performed to evaluate the prediction. Based on the same model structure, I'd like to compare the results in cost function, using root mean square error (RMSE), and observe the loss during the training process.

A. Analysis in Different Epochs

The epoch is set as 10, 50, and 100 with the same batch size equals to 10.

- Observation of losses

  We can see in epoch = 100 at bottom-left that the loss drops significantly in around epoch=10 and then become relatively stable by the value equals to $5 * 10^{-4}$, but the lowest loss occurs in epoch=50 set, which is around $3 * 10^{-4}$. (B: Batch size; E: Epochs)

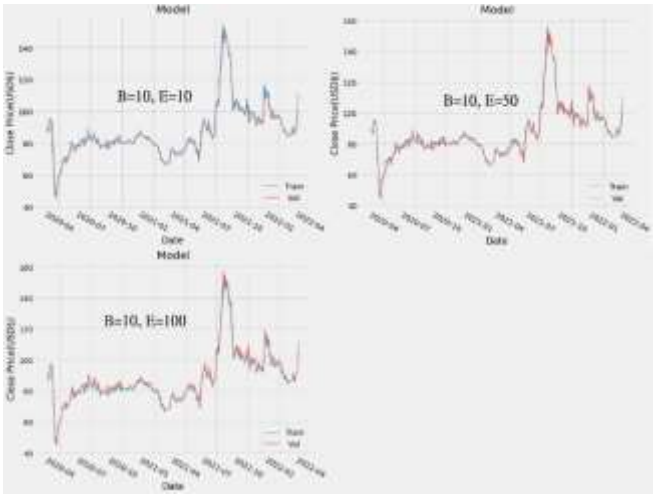


- Comparison of different epochs

  We calculate the root mean square error to see the precision towards the test set, as shown in the table below. It shows that in epoch=10, the predictions are closer to the test data.

Batch size	RMSE		
	Epoch =10	***Epoch = 50***	***Epoch = 100***
10	0.819346	0.824618	1.309416

In addition, the line plots of each epoch are shown below. It shows the trends of three lines fit quite well to the test data.

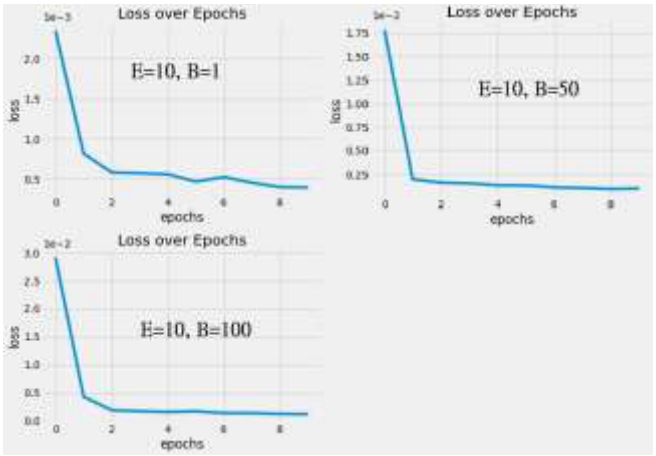


B. *Analysis in Different Batch Size*

The batch size is set as 1, 50, and 100 with the same epoch equals to 10.

- Comparison of losses

  Below, we can see in batch size = 50 at top-right, the lowest loss occurs, which is around $1.0 * 10^{-3}$. (B: Batch size; E: Epochs)

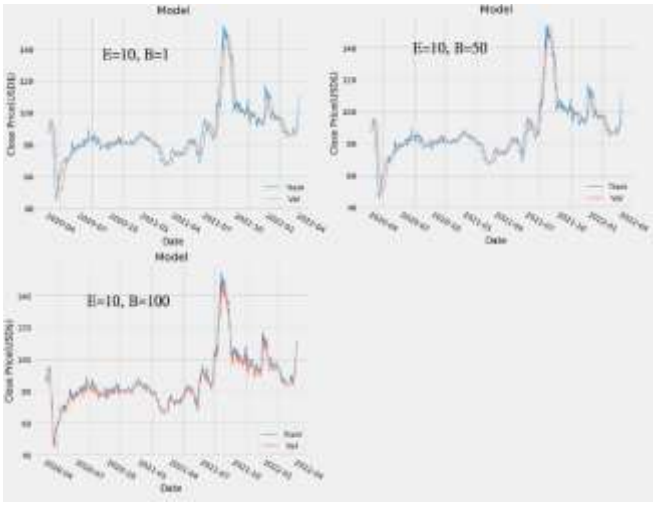


- Comparison of Different Batch Sizes

  Similarly, we calculate the root mean square error to see the precision towards the test set, as shown in the table below. It shows that in batch size = 100, the predictions are closer to the test data.

Epochs	**RMSE**		
	Batch size =1	***Batch Size = 50***	***Batch Size = 100***
10	2.0405152	0.5576838	0.3613455

In addition, the line plots of each epoch are shown below. It shows the trends of the line, batch size = 100 fits quite well to the test data.



IV. CONCLUSION

After training with different values of batch size and epoch, we could conclude that:

1) When fixing the batch size, as Epoch grows, the performance is not guaranteed to be better, and it could be worse and cost much of time training.

2) When fixing epochs, batch size has highly correlation with the performance

V. FUTURE WORK

In this paper, it only used close price as the training data, that is, the model is not multi-variate; in addition, the number of layers may be not the most effective one. Therefore, I think the most significant way to increase the performance is to establish the model in a form of multi-variate one. Second, the structure also needs to be properly modified in a sense of efficiency, rather than keeping adjusting the batch size of the epoch only.

Moreover, there are plenty of pre-trained model nowadays, and it would be more adequate to convince other people how the model performs by comparing with other ones.

VI. ACKNOWLEDGMENT

Thanks to Professor, Yeh and TAs' tutorial for the neuron network. It's a profound start for me to dive into machine learning domain since I've never gave it a try, but decided to put myself into it for my career few years ago. Also, it's a chance practicing writing a paper in English.

VII. REFERENCES

1. Sepp Hochreiter; Jürgen Schmidhuber (1997). *"Long short-term memory"*. Neural Computation. **9** (8): 1735–1780.