CS410 Final Project–Stock Price Prediction

516030910514 Wang Haoxuan

December 31, 2018

Contents

1 Abstract

We implemented various methods on the stock price prediction problem, including traditional methods and neural network methods. Traditional methods such as random forest regressor and gradient boosting regressor did not perform well, thus we turned to LSTM for better performance. By using a three layer LSTM with dense layers and dropouts, we reached a score of 0.00150 on the public dataset and 0.00144 on the private dataset. Also, a model that combines CNN with LSTM reached a score of 0.00152 on the public dataset. We would demonstrate below about how we constructed each model and tuned them towards our best results.

2 Introduction

In the stock market, billions of dollars are traded every day. The price of the stocks play an important role in the market and making predictions of them is very important. However, the prices depend on various variables and the relationship between them is quite complicated. We want to find this particular relationship, as we think it cannot be random. Many people have worked on this project, but traditional mathematical solutions often do not reach a satisfying result, thus, complicated models need to be constructed and trained.

In this problem, 7 types of data are given(bid-price, bid-volume, ask-price, ask-volume, last-price, mid-price, and volume), they are all from the same type of stocks, and the total time length is not too long. We want to predict the average value of the next 20 time stamps. Availabe data are the values of the past ten time stamps, that is a total of 70 numbers.

In this report, we demonstrate the details of how we examined this problem and how we constructed our models using traditional methods and reinforcement learning methods, as well as their results.

In this project, I implemented the part of traditional methods and tried on the LSTM + supervised learning model. My teammate Tian Changda implmented the pure LSTM and CNN + LSTM part.

3 Problem Examination

After examining the problem thouroughly, we can derive the following conclusions, these ideas are the foundations of our experiments.

- Only one value is needed to be predicted. Even if we can predict the distribution of the next 20 time stamps, sampling from the distribution would be unaccurate; Or we could just predict the next 20 values, but that would even more unaccurate as the errors may accumulate.
- The 7 given features for predictions are not independent from each other, and they are also associated with time. Thus, original data are not so well organized, we can calculate the difference, ignore some of them and even transform them through some function before putting them into the models we have constructed.
- Since we are doing prediction, data from the future cannot be used. Otherwise

- Though NN methods are highly recommended of their performance, we still want to see if traditional methods can perform well. As we assume that only one or two features are essential, traditional methods may do well in predicting the results with these methods.

In the following sections, we would demonstrate the details of how we dealt with the problem using traditional methods and neural network methods.

4 Tools and Data

Keras is a deeplearning network modeling tool package in Python. It is easy to use and can free users from complicated mathematical formulas, letting us directly consider the concrete deeplearning network structure. And Keras need a backend deeplearning network tool plantform.

There are three mainstream deeplearning network tool plantforms, namely, CNTK, Theano and TensorFlow. Though TensorFlow is in fashion now, we choose CNTK as our backend deeplearning network tool plantform, since CNTK has the following strengths:

1. Faster. According to the research results of Hong Kong Baptist University, when running GPU, CNTK outperforms other computing environments in most of the models, and in the cyclic neural network model, CNTK outperforms other computing environments by a large margin.

2. More accurate. CNTK provides many advanced algorithm implementations to help improve prediction accuracy.For example, the Automatic Batching Algorithm allows analysts to merge sequences of different lengths. It improves the operational efficiency and achieves a more optimized randomness, which usually improves the prediction accuracy by one to two percentage points.

3. Good expansibility in GPU. CNTK has its own SGD and Block-Momentum SGD algorithm which can implement high GPU parallel calculation.

Since our group has a GPU computer, we choose CNTK as our backend deeplearning tool plantform.

For the given data, there are 9 columns, which are: Date(in the format YYYY-MM-DD), Time(in the format HH-MM-SS), MidPrice(the mean value of buy in price and sell out price of the stock), LastPrice(the latest deal price of the stock), Volume(the cumulative turnover on that day), BidPrice1(the highest bid price), BidVolume1(the volume corresponding to the BidPrice1), AskPrice1(the highest ask price), and AskVolume1(tHe volume corresponding to the AskPrice1).

5 Traditional Methods

5.1 Random Forest Regressor

Since we intend to extract the features from the known features, using the random forest regressor is a good intention. The random forest method construct trees rooted on different features, and each tree construct logical relationships based on the values of the features. Since we are doing

value prediction, regressor is more recommended than classifier. The Sklearn package includes a well defined class of random forest regressor, thus we adopted it for training.

The data to be put into the model was not carefully dealt with. We only took $7 * 10$ values(all the values for the past ten time stamps) as the input and 1 value(the average midprice for the next 20 time stamps) as the output. For all the values, scaling is highly recommended, as their absolute values vary greatly. MinMax scaling is used, so that we can get values from 0 to 1. From printing out some of the training data, we can find that most of the nearest ten data are very similar to each other, at first we thought this was a good phenomenon that shows our data preprocessing was correct, but latter we found that this was a quite troublesome question as this makes the training difficult.

The result was not satisfying, as we can see in the table below. Setting 300 trees is for covering all the features as the roots of the trees are selected randomly and there might be overlaps; And a max depth of 10 is for convergence, as too deep a tree would cause the training time to be extremely long and the model would be hard to converge.

Reaching a result like above is definitely not acceptable, we then examined the data and found that the data are actually divided by time: There are huge faults between the data, such that the previous data is in the morning and the next data is at afternoon. Thus, we need to ignore this part of training data as they might mislead our model. After doing this operation, we trained the model again, but this time with only 50 *trees* and still depth 10, and reached the same accuracy. We decreased the number of trees to shorten the training time.

	Tree Number	Max Depth	Accuracy
Without Time Clip	300	10	0.00340
With Time Clip	50	10	0.00340

Table 1: Result for the Random Forest Regressor

From the result above, we reach the following conclusions:

- Time clipping is not so important, as the noisy data is very less compared with all the data, clipping them may lead to some improvement, but not so significant.
- Random forest method did not perform well, this should be due to that the features extracted are not independent from each other, they are either related by some mathematical transformation, or related by time. Thus the worse performance is expected.

5.2 Gradient Boosting Tree

After achieving the result using random forest regressor, we were actually not so convinced that this kind of method is unsuitable. Thus we tried using gradient boosting tree, as we think using gradients to deal with values is more reasonable. The result we have reached is shown below:

	Loss	Learning Rate	Tree Number	Max Depth	Accuracy
Without Time Clip	ls	0.1	300	10	0.00355
With Time Clip	ls	0.1	300	10	0.00348

Table 2: Result for the Gradient Boosting Tree

Our model won't stop training until it converges, and by the result, we can find that its performance is even worse than the random forest regressor. This made us to consider that using features as the evidence for predicting might be a wrong direction. Also, more tuning on the parameters shouldn't lead to much better resultsto reach the benchmark. Thus, we consider other kinds of models below.

5.3 Hidden Markov Model

Hidden markov model is a classfical way of predicting stocks. We can construct 3 hidden states: Price go up, price go down, and price stay the same. The evidences are significant, thus the whol markov chain is quite long and complicated. We are able to construct and train the whole model, but only defining the three hidden states is not enough for prediction, we cannot just determine one hidden state and let the true value to be predicted be generated by random. Thus, we were interested in the range of the value change, if it very small and have a certain distribution, then the hidden markov model could be used and we can use its result to predict. However, this did not work as we found that the value varies from -1 to 0.8, and they do not distribute with a certain pattern. Thus, HMM was out of consideration of our model.

5.4 Traditional Methods Conclusion

From the discussion above, we can find that traditional methods are quite clear in its theory, we know exactly what the model is doing, why it is performing in some certain way and have a clear direction of how to improve its perfomance. However, through our experiments, we find that their performance cannot help us to reach the benchmark. We think it may be that traditonal methods are better at doing classification than regression, using them to construct functions with high dimensions is very difficult. In the past years, since neural networks have proved to outperform traditional methods significantly, we choose to construct NN structures to do prediction.

6 Neural Network Models

6.1 Basics

Many neural networks model were found and put into real life applications, thus choosing which one is of great importance. Of course we can construct our own neural network structure, but that is of no necessity, many experts have already constrcuted efficient models experimented on them and achieved great results.

To make our model construction easier so that we can spend less time on model constrcuting and more time on thinking about how to imporve our model, we use Keras as our general structure for training. Since the data given are sequential, RNN is the first choice for us. Also, the data has

a time feature, thus LSTM model is our first thought. Besides LSTM, we have also tried other models for prediction, which would be demonstrated below.

6.2 LSTM + Supervised Learning

LSTM(Long Short-Term Memory) is an RNN structure for predicting sequential data. It was widely used in natural language processing, where a line of words have relationships with each other in a time order. LSTM uses 2 gates to control the state of its unit. One is forget gate, it decides how much of the cell state of the previous moment is reserved for the current moment.The other is input gate, it determines how much network input is saved to the cell state at the current time. And LSTM uses output gate to control how much the unit state will contribute to the unit output of the current moment. Before using it, we first have to prove that the data has correlations with time, thus we plotted the following graph:

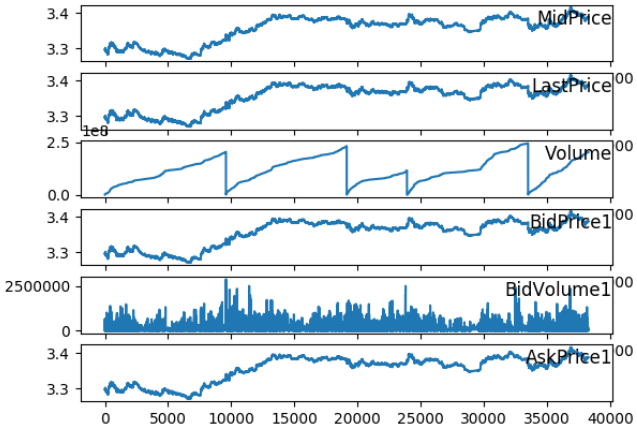


Figure 1: Data relationships with time

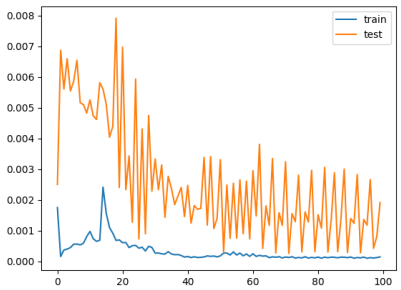
From the graph, we can find that the midprice, lastprice, bidprice and askprice are highly relevant with each other, volume has a periodic relationship with time, but bidvolume's relationship with time is not so clear. However, this does mean we want to ignore the bidvolume, there might be in fact some inner patterns that is very important.

Since the most common LSTM is used for only on kind of features(a bag of words), it should be used only to predict one column of data(which means one feature). Thus we tried it first, and reached the benchmark. But this is not what we want, otherwise the other features has no meanings. Making full use of them should lead to a better result. Thus, we intend to use an LSTM + Supervised Learning methods to do prediction.

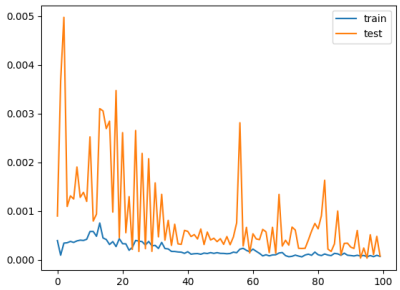
The data should be preprocessed first, we do the scaling first and delete the data that are not continuous in time. All the data are put into a pandas table, shift operation is used, and data that occur with NaN is ignored from the dataset. The neural network is simple, only one layer of LSTM and one layer of dense constructs the whole model. Three models are constrcuted based on different operations on data:

- We predict the next 20 values based on the past 10 time stamps.
- We predict the next value based on the past 10 time stamps.
- We predict the average value of the next 20 values based on the past 10 time stamps.

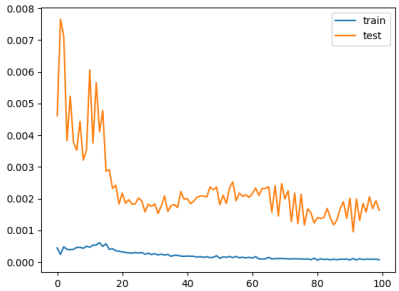
These three models' convergency is shown below:



(a) Model 1



(b) Model 2



(c) Model 3

Figure 2: Training Convergence

From the graphs, we can see that the training set converges quickly, the validation set also converges but not in a manner we like(the first one isbecause we did not ignore the data that does not satisfy the time requirement), their accuracy was low too:

	Accuracy
Model 1	0.03863
Model 2	0.018
Model 3	0.018

Table 3: Accuracy for LSTM + Supervised Learning

These models got very bad results, we think the reason is that:

- First, they are not well tuned, parameters are all set arbitrarily.
- Second, the whole strcuture of the model might be too simple so that it cannot extract the features from the dataset.
- Data preprocessing is too simple, not much is done to deal with the data and there is too much noise in the data for the model to be trained efficiently.
- LSTM with supervised learning may not be a good idea, or to say, it makes the original problem more difficult, but there is actually no difference with pure LSTM.

This model was just a try on LSTM and Keras, details are simialr with pure LSTM, which we would demonstrate below.

6.3 Pure LSTM

6.3.1 The first model

In the neural network regression models, a very important work is data preprocessing, it decides the upper bound of the model.

In our first model, we just read the columns from the training data file, and do z-score normalization to all kinds of the training data. For X_train data, we added the LastPrice, Volume, BidVolume1, AskVolume1,BidPrice1,and AskPrice1 in the current 10 time steps. We didn't add MidPrice because MidPrice is the mean value of AskPrice and BidPrice. For y_train data, we used the mean value of the next 20 MidPrice data. For all train data, we set the step equals to 4 to speed up training.

In this model, we used 2 layers of LSTM with 64 units each. After each LSTM layer, we added dropout layer to prevent the over fitting problem. Then we used 2 layers of fully connected layers with 64 units each, with Relu to be the activation function. Then, we used adam optimizer ,using mean squared error as or loss function and setting the batch size 32 and trained 10 epochs.

We treat the test data the same way like the training data, and predict the y_test result. Then, we use reverse z-transform to get the result of our first model. However, sadly, after submitting our prediction to kaggle, our accuracy is only 0.054.

We think that may be our data preprocessing is not very good, then comes our second model.

6.3.2 The second model

After our first model, we think we should examine the training data carefully. Then, from the data, we find that the time of the stock price is not continuous. On the one hand, the trade time of the stock is 9:30-11:30 and 13:00-15:00 on workdays. On the other hand, there are some errors exist in the give training data. Since the test data given is continuous, we have to clean the training data and get rid of the discontinuous data.

We used the time package in Python to help us get rid of the discontinuous data. First we get the time data from Date and Time columns of the training data file and join the date and time together. Then, we uses time.strptime() function and time.mktime() function to get the time of each row in the training data file. The time of each row is a large integer whose unit is second. So, when we put data in the X_train and y_train list, we only need to see that whether the current data's time is exactly 60 less than the 20th data's time after the current one. If the answer is false, we reject the current data and do not put it in the train list.

And we also find that the Volume data is very large and varies little, so we think maybe its difference is a better factor for predicting the stock price. So, we made new data for all Volume data by the fomula.

$$newVolume[i] = Volume[i] - Volume[i-1]$$

.
Then we use the same method as the first model to preprocess the training data. This time, considering that deeper network has better learning ability, we added another LSTM layer to the network also with dropout layer. And we make the number of units in each LSTM layer to 128. The other parameters are the same as the first model.

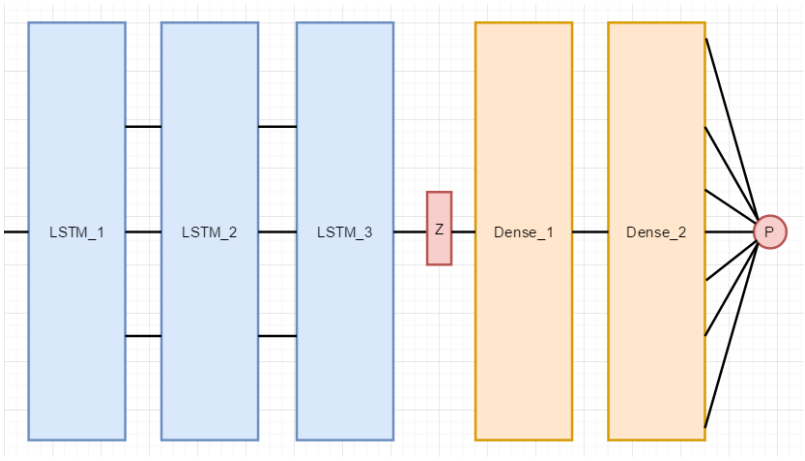
After calculating the predict result, we submitted it to kaggle. We get better result but it is still not satisfying, which is 0.022 accuracy. Then, we think we should listen to our classmates about how they construct their models.

6.3.3 The third model

After listening to other teams' method, we find a fault in our data preprocessing method. That is, the training data and the test data is not with the same distribution. In the training data, the MidPrice is from 3.6 to 3.9. While in the test data, the MidPrice is from 3.2 to 3.5. So if we just use the mean value of the next 20 MidPrice to be our y_train, it will be very inaccuracy.

So, we changed our y_train. We used the difference of the mean value of the next 20 MidPrice and the MidPrice of the last item in the 10 training data. And also we used z-score normalization to treat the new y_train data.And for other factors, keep the same with the second model.

This time, we also used 3 LSTM layers with dropout layers. And we changed the fully connected layers' unit number to 128 since we want it to have better performance.



We also set the batch size = 32 and epochs = 10. But after the training process, we find that the loss is still decreasing, which means that the model is not converge. So, we retrained the model and set epochs = 50. This time, we get the converged model.

This time, we get a good model which received 0.0015 accuracy rate in kaggle's public leadboard, better than the bench mark.

However, we found that the team in first place used a CNN layer before the LSTM layers, and this approach is acknowledged by our TA, so we want to use CNN to extract some features of the data, and then put them to LSTM layers.

6.4 LSTM + CNN

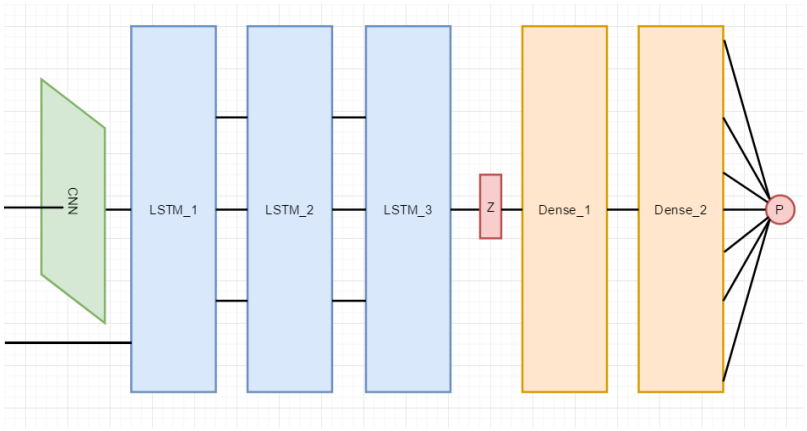
6.4.1 Data preprocessing

We used the same method for data preprocessing as the third model.

6.4.2 Network model sturcture

We set a new sequential network model before the original LSTM model. We used a 2D CNN model. Considering the 6*10 training data as a figure and make it through a CNN layer with 32 filters , 3*3 kernel_size and 1*1 strides. After the CNN layer, we also added a Max Pooling layer with 2*2 pool size.

After our data go through the CNN and pooling layer, we concatenate its result to the original X_train.



Then, using the same network structure as the third model.

6.4.3 Result

Adding an CNN layer before the original LSTM model seems not improve the model. The accuracy of the forth model in kaggle is 0.00152 in public leadboard. While this approach is still better than the bench mark.

6.5 GAN-Based Prediction

We have also considered using GAN as a way of doing prediction. Since GAN is used to generate distributions, we can use this generated distribution to reach our result. However, this was rejected. The reason is that we were not sure about what to generate. If we generate a distribution, how to sample from the distribution so that we can reach accurate result is a vital problem; If we generate one value, then the model would be similar to the previous ones but it is harder to train; If we generate 20 values, there is much doubt on whether the model would be able to converge or not, since GANs are already hard to train and converge. Due to the time limit, this GAN idea was not approved of, and we are looking forward to other new thinkings to improve this idea.

7 Conclusion

Through this project, we have learned a lot about the task of prediction. Traditional methods seem to be out of date, but they consist some important thinkings that are essential for dealing with these kind of tasks. Also, the traditional models are easier to train and tune, as their inner structures are shown to us clearly. In the contrast, neural networks are similar to a black box, where we can only observe the input and output, but the way they can represent complex functions and their fantastic performance made it the most powerful tool in prediction nowadays. In this project, we have experimented on both traditional methods and neural network methods, achieving different performance. Our top 2 models are the pure LSTM model and the CNN+LSTM model, getting an score rate of 0.00144 at last. By tuning the hyperparameters and modifying our models, we get a better insight into the field of deep learning, and formed basic senses about how to choose a model

and how to modify it to make it perform better. Finally, best thanks to the effort made by our professor Zhang LiQing and our TA Zhang Yiyi in this project.

References

[1] Artificial Intelligence: A Modern Approach, 3rd Edition. Peter Norvig, Stuart Russell, 2009.

[2] Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, Wang-chun Woo, CVPR 2015.

[3] Generative Adversarial Networks. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, 2014.