A Simple Trading System Based on Deep Neural Network

Jared Qiu
Statistics
University of Illinis at Urbana Champaign
Champaign, US

***Abstract*—This paper will discuss how to use deep learning technic to build a simple trading system. Neural network like CNN and DNN are pretty good at fitting and prediction model for lots of data. Financial market is a good place to test if the model is useful because the data update every single second globally.**

Keywords ---- Deep Neural Network; Convolution Neural Network; Trading system; Financial data

Introduction

Asset management is a significant component in society. It is the catalyst of the global economy which could provide lots of liquidity for financial market. In this report, we will discuss the possibility of fitting model with Deep learning which is a brunch of machine learning that has great ability in prediction. We'll try to combine these two to use deep neutral network to fit financial data. The motivation we choose this topic is due to our passion for quantitative finance. Financial market is always full of new data and we can test our model whenever we want. The input data would open, low, high, close prize data in the past 10 days. We combine them to get input data with size 40 for DNN and size (40,1) for CNN. The output layer contains a single output which is the closed prize for the day we want to predict.

I. Related Work

A. Idea Scource

The idea came from the paper published by a PhD student in Cambridge. He tried to use DNN to test if financial market is predictable.

B. Related work.

Deep learning is a fast growing area. Many researchers who have experience in deep learning have applied it to different areas. Finance is the most promising one of them and we choose it as our focus area. Our project's idea is based on the paper written by Yong and his colleagues. Our method are almost the same except for different optimizer and activation function. We collect data from a similar source New York Stock Exchange. We also build three layers based forward network. The main difference is that Yong's paper emphasizes on how to predict future stock price while we are not satisfied with the DNN model and continue with a CNN model.

To build a CNN model, we learn from Ashwin and his partners' paper. Ashwin's paper is rather comprehensive, including topics like choosing a time window of 30 minutes, choices of loss function, ordinary least–squares regression etc. Because our goal is to get a CNN model to predict stock price, we learn from this paper how to train a CNN model and ignore the other topics of the model. We also learn from Ugur'a paper to learn how to build A CNN model that will be illustrated in detail. He used CNN as main structure for his model to predict ETF price. The theory is to think financial data as a 2-D convolutional neural network because every data point has at least four attribute(open, low, high, close) This method did pretty well at classify and predict daily asset state(up/down).

LSTM is one of the most popular method I have seen so far that is applied in financial market. LSTM belongs to one of recurrent method in deep learning. Basically, LSTM is similar to autoregression if you are familiar with time series. They all have one thing in common, current result depends on the previous result. Jialin, (2019), Xue(2020) all use LSTM as main architecture in the papers. The reason behind this is financial

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

data is one kind of time series data that there can be lags of unknown duration between important. Thus LSTM would be a good choice. We studied both Jialin's and Xue's paper and tried to implement LSTM model in a similar way. But we failed due to some technical difficulties and would only include DNN and CNN for our model in this paper.

II. DATA

The data we choose is NYSE (New York Stock Exchange) market data. The data are standard financial data which contain 6 attributes – company, open price, closed price, high, low and volume. The attribute we use for prediction is open, closed, high, low and Volume. The source of the data is from the GitHub. Many people build model with financial data in GitHub, thus it's easy to find classic data in there. The shape of raw data is (851264, 6). We did data cleaning and exclude all unnecessary part. The company we choose is Willis Towers Watson PLC. This is what our model will focus on. Of course, other company could be used in this model too with enough training. The shape of data after cleaning is (1762, 5). This is about six years stock price for Willis Towers Watson PLC. The data contains daily information for every trading day during the period.

A. *Training / Validation /Split/Transformation*

The split ratio is 9:1. We use 90% data as training data and the rest as testing data. We did data normalization to the data which is [0,1] normalization. This could restrict all data in [0,1]. No other special step required.

B. *Sample Data*

Below is the first 5 entry for cleaned and adjusted data. It contains data from 2010-01-04 to 2010-01-08 and all the attribute we would use for training the model.

date	open	low	high	adj close
2010-01-04	0.157047	0.161167	0.156390	0.159399
2010-01-05	0.157238	0.158884	0.154995	0.157092
2010-01-06	0.156140	0.146049	0.153341	0.143942
2010-01-07	0.142436	0.134457	0.140094	0.132105
2010-01-08	0.127950	0.131464	0.134455	0.138726

Data Source:
https://raw.githubusercontent.com/ashishpatel26/NYSE-STOCK_MARKET-ANALYSIS-USING-LSTM/master/nyse/prices-split-adjusted.csv

III. METHODS

We have two models, DNN and CNN.

For DNN. The network structure has a simple structure. It is actually not easy for DNN to capture pattern of time points for two-dimensional data. But here every data point is a five-dimensional data. (open, high, low, close, volume) Thus, we think it worth a try.

A. *Description*

We tried two different models here which is DNN and CNN. The first model we used here is DNN. The model has four hidden layers and for the first three layers it contains 10 neurons and uses ReLU activation for all neurons. For the last hidden layer, it uses sigmoid for the only one neuron.
The second model we used is CNN which is a much better model than CNN for our project because it can better capture the variation of financial data. I'll do detailed explanation for both of them in the following part.

B. *Why it Work*

We use open, close, low, price, volume in the past ten days as our predictor to predict our response variable which is the closed price in that day. The model fits the data pretty good. We use a learning rate = 0.01 with loss function of mean squared error. The normalized loss drops from normalized loss 0.0671 to 0.0557 after training We use gradient descent which is an optimization method to make the loss as low as possible and it does work.

```
model.summary()

Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 10)                410
_________________________________________________________________
dense_5 (Dense)              (None, 10)                110
_________________________________________________________________
dense_6 (Dense)              (None, 10)                110
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 11
=================================================================
Total params: 641
Trainable params: 641
Non-trainable params: 0
_________________________________________________________________
```

DNN Architecture

The reason DNN works is the cost function we use is a convex function whose shape looks like a U. The optimization method we use can make cost stay around the bottom of U which is the minimum value.

Unfortunately, The DNN model has a pretty terrible result which can barely predict the future data. Professor James told us it's because DNN is too simple to capture the financial data variation.

As a result, we decided to use another method which is CNN.

CNNs are usually used for visual images and categorizations, however it could be used in the regression cases like this. In order to use the CNN, we need to change the shape of our input and output values, and then redo the train-test split.

Previously our feature data is at shape (1750, 40), which is a two dimensional. It represents 1750 observations and 40 variables. To make regression predictions in CNN, we need to use the Conv 1d model in Keras. Thus, we need to make the input one dimensional. To do that, we can add a dimension and treat the whole data as a single input row. Then, we can test-train split the data again, using the 90% cut-off.

For creating the CNN model, first we add a Conv1D layer, the input shape is (40, 1) because we have reshaped the data to be a single input row. It uses ReLU as activation function. Then we use a flatten layer to flatten the data for further calculation. Then we add a dense layer with ReLU. Finally, we compile the model and calculate the loss with ADAM optimizer.

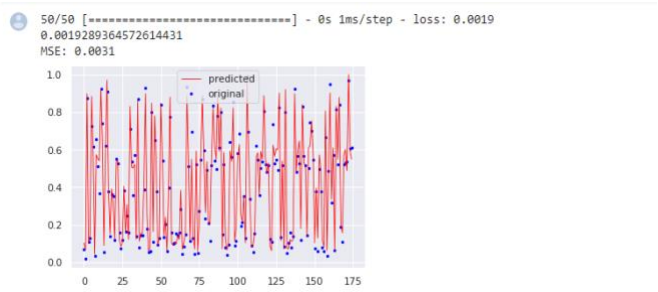
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 39, 32)            96
_________________________________________________________________
flatten (Flatten)            (None, 1248)              0
_________________________________________________________________
dense (Dense)                (None, 64)                79936
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 80,097
Trainable params: 80,097
Non-trainable params: 0
_________________________________________________________________
```

CNN Summary Table

It works because we use the Conv 1d model in Keras and we make the input one dimension. After creating the model, we can fit and predict it with our training data.

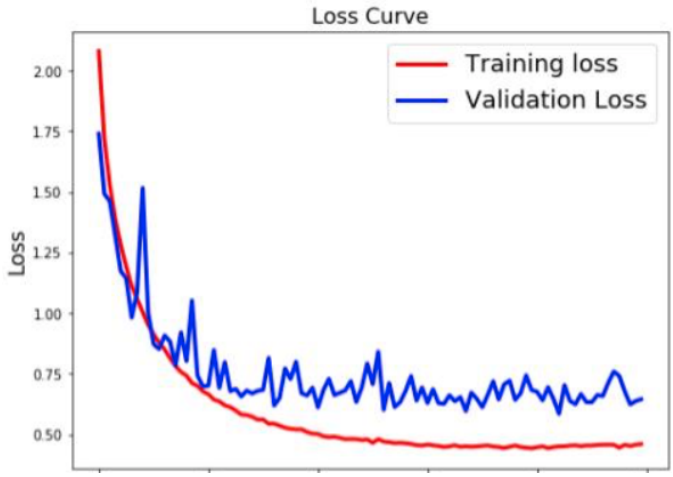
IV. Results

The plot of original VS predicted is at below. We use a batch size of 12 with 200 epochs, with loss function mean squared error.



Graphics of Original VS Predicted

The result from CNN model is very good, with an MSE of 0.3%. We can see the training loss keeps decreasing because our model is trained to do well in this data. Test data has an obvious decreasing trend even though it's not as good as training data but totally good enough. The reason behind it is that test data are actually unseen by the model. And in the first plot MSE can see the predicted data point is not far away from the true data which means out model is good. It can do well on unseen data and not overfitting.



Graphs on Training/Validation (Dev)/Test

V. Conclusion

Predicting financial stock price is one of the influential practices of deep learning. Many funds started using AI technics these days like two sigma / tower research or some other quant fund. What we are doing here definitely is just

building the simplest model dealing with financial data, which means we have great area to improve our model if we want to use it in practice. Although our DNN and CNN model cannot be completely accurate, it is still useful. With the neural network model we built, we can predict the prize data and have a rough idea of what kind of result should we be expecting.

We also considered to build a LSTM model, but as said earlier, we failed due to some technical problems. We might try it in future.

We definitely have learned a lot from this practice. This model is our first trying applying AI in financial market. We notice It is definitely not easy at all to use our knowledge in practice. However, we'll keep going in this road no matter how hard it will be because this is where our passions are.

VI. References

[1] Yong, Bang Xiang, Mohd Rozaini Abdul Rahim, and Ahmad Shahidan Abdullah. "A stock market trading system using deep neural network." Asian Simulation Conference. Springer, Singapore, 2017.

[2] Gudelek, M. Ugur, S. Arda Boluk, and A. Murat Ozbayoglu. "A deep learning based stock trading model with 2-D CNN trend detection." 2017 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2017.

[3] Liu, Jialin, et al. "Stock prices prediction using deep learning models." *arXiv preprint arXiv:1909.12227* (2)

[4] Siripurapu, Ashwin. "Convolutional networks for stock trading." Stanford Univ Dep Comput Sci (2014): 1-6.

[5] Yan, Xue, Wang Weihan, and Miao Chang. "Research on financial assets transaction prediction model based on LSTM neural network." NEURAL COMPUTING & APPLICATIONS (2020)