CS230

Stock Price Prediction with Deep Learning Framework

Anran Lu
ICME
Stanford University
anranlu@stanford.edu

Zeyu Wang
Department of Economics
Stanford University
zeyuw@stanford.edu

Huanzhong Xu
ICME
Stanford University
xuhuanvc@stanford.edu

Abstract

Stock price prediction is an important topic for portfolio construction. Although according to Efficient Market Hypothesis [3], no information can be used to predict the stock market in such a way as to earn greater profits, we would like to see if there is a way to beat the market in reality. We use both traditional method ARIMA and neural networks including LSTM, GRU and Dual-Stage Attention-Based RNN (DA-RNN), since time-series models are natural options for this task. Our experiments show that DA-RNN generally has better performance when predicting stock prices, but it also suffers from problems like over-fitting due to the data size.

1 Introduction

Stock price prediction has always been a hard and important field for portfolio construction and market efficiency. Eugene Fama's Efficient Market Hypothesis [3] demonstrates it's impossible to "beat the market" consistently on a risk-adjusted basis. Although many studies supported the hypothesis, recent papers have shown otherwise. Classic financial models tend to make intense use of the market structure, thus bringing assumptions contrary to reality from time to time. Hence we are interested in predicting stock prices with a deep learning model, which knows nothing about finance, and see whether it can have a more robust performance. This is not just about making money, but also understanding how the stock market works and whether it is efficient.

The input to our algorithm is historical stock prices of Apple and some other relevant companies. We then use ARIMA and variants of RNN to predict stock prices in the near future. We also compare the traditional statistic methods and deep learning methods to evaluate the precision and robustness.

2 Related work

Stock price prediction has been popular for years in the field of finance. Fama and French [4] proposed the seminal three-factor model to predict stock and bond return by regression. Though it can predict the trend and expectation fairly well, the performance is very limited when looking at daily price prediction.

In recent years, stock price predictions get a lot of interests in the deep learning community. One of the earliest work in this area is by Schöneburg [9]. At that time, there was no RNN or complicated structure, so he decided to tackle a rather simple problem: only predicting the next day up or down, using PERCEPTRON, ADALINE, MADALINE and BACK-PROPAGATION models. It turns out to work fairly well and encourages much follow-up work on stock price predictions.

CS230: Deep Learning, Autumn 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

Recently, Adebiyi, Adewumi, and Ayo [1] compare a three-layer multilayer perceptron model with traditional ARIMA model, and reveal the superiority of neural networks model over ARIMA model. However, it ignores the time series natural of stock price prediction in the design of neural network. Rather, Agarwal and Sastry [8] use RNN models mixed with traditional statistical models to predict stock price. However, the network structure is very simple and may not capture the complicated nature of stock markets.

One state-of-the-art model for stock prediction is by Bao, Yue and Rao [2]. They first decompose the noise from the data by wavelet transforms, then apply stacked autoencoders to generate deep high-level features, and finally feed the features into a LSTM. The novelty in this paper comes from the three-stage process, instead of better structure for time series fitting.

We borrow our network idea heavily from a dual stage RNN by Qin et al [7], another state-of-the-art model for time series prediction. A major flaw of their project is that they predict the present S&P 500 index by taking 81 stocks' past and present prices as input. Not to say that the 81 stocks are highly correlated with S&P 500, using today's information to predict today's price makes the application less meaningful. We want to avoid this flaw by only using past information to predict future stock prices.

3 Dataset

The main dataset we use is daily APPLE price[10]. All of the historical stock prices are fetched from Yahoo Finance. Since we are trying to make our code as end-to-end as possible, we write our own data preprocessing module `data_cleaning.py` to fetch the stock prices from Yahoo Finance and save it into a CSV file, helping each user to define his or her own database. User may specify which stock the model should be trained on. For instance, to fetch the prices of Apple from twenty years ago to today, simply run:

```
$ python data_cleaning.py AAPL
```

The resulting CSV file will be saved to a CSV file, and ready to be trained or tested on later. Note that the file will contain all five prices, including everyday prices such as open, high, and low. The default column we use to develop our model is **adjusted close**. The default train/test split rate is 0.9, so the first 90% time series is used for training and the rest 10% is reserved for testing.

4 Methods

We implement ARIMA, LSTM, GRU and DA-RNN. Our input here is historical adjusted close price and our output is predicted stock price.

ARIMA

ARIMA is a generalized autoregressive moving average model (ARMA). Specifically, it is called ARIMA(p, d, q) if $(1-B)^d x_t$ is a stationary ARMA(p, q) [5], where B is the backshift operator defined by $Bx_t = x_{t-1}$. ARIMA(p, q, d) process can be expressed as

$$\Big(1 - \sum_{i=1}^{p} \phi_i L^i\Big)\Big(1 - L\Big)^d x_t = \Big(1 + \sum_{i=1}^{q} \theta_i L^i\Big)\epsilon_t,$$

where L is the lag operator, θ is the moving average and ϵ is the error term.

Common RNNs

To compare with ARIMA, we use LSTM, adapted from [6] to perform the prediction as our deep learning baseline. LSTM unit has a cell, an input gate, an output gate as well as a forget gate. We use it because it allows lags of unknown period among different events and prevents the problem of vanishing gradient. The architecture of LSTM we use is shown in Figure 1.

We also use GRU to predict stock price. GRU requires fewer parameters where it does not have an output gate. Since we only use 20 years of historical prices in this project, we expect GRU performs better than LSTM. Its architecture is shown in Figure 1.

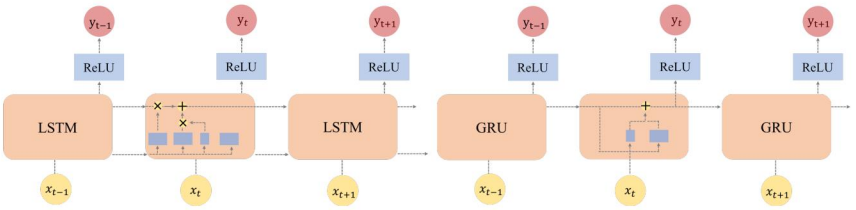


Figure 1: Architecture of LSTM & GRU

DA-RNN

DA-RNN [7] is adapted from an encoder-decoder RNN. It not only takes the historical price as input, but can also take other time series data as input for driving forces. This feature can incorporate as much information as possible in making predictions. In the first stage, the model uses an attention mechanism to select the relevant driving force at each time step based on previous encoder hidden state. In the second stage, another attention mechanism is used to extract relevant encoder hidden states across all time steps. The two attention mechanisms are integrated with an LSTM-based RNN and can be jointly trained using standard back propagation. The advantage of this model is that it can capture both "useful" input features and the long-term temporal dependencies of the stock prices. Its architecture is shown in the following figure.

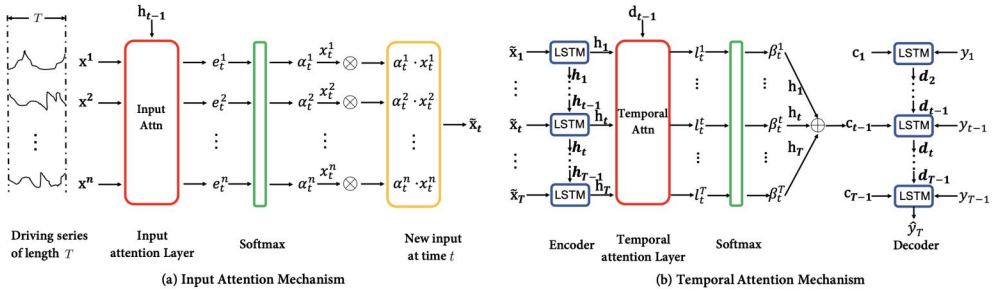


Figure 2: Architecture of DA-RNN

5 Results and Discussion

To make different methods comparable, in the result section we always apply our methods into prediction of the stock price of Apple. For the hyper paramter, we choose 3 as the window size, 30 as the sequence length for LSTM and GRU. We have two layers, each with 128 hidden unites in our network. Batch size for gradient descend is 8. We tuned all these hyper parameters by ourselves, taking into account both the computing resource and over-fitting problem.

We use MSE as our loss function. Note that in addition to stock prices, we also train and test our NN models on daily log returns, which is defined by

$$R_t = \log \frac{P_t}{P_{t-1}}$$

Note that both models measure how accurate the prediction was, since one can easily derive daily log return from the daily stock price, and vice versa. However, the error of daily stock price can be huge as the prediction progresses into later time period, as the error in the earlier periods accumulates. The huge error in the later period not only makes the error in the earlier period negligible, but also make the metrics irrelevant. Daily log return, on the other hand, has a much smaller range (usually between 0.9 to 1.1), and stay in the same scale no matter in earlier or later periods.

For ARIMA the traditional statistical model, we choose the parameter $p = 4, q = 1, d = 0$ by trying different combinations of parameter and choose for the best. The results are in Figure 3. As we can see, the traditional statistical model successfully captures the upward trending of stock price. However, it fails to capture daily volatility of stock price, which is not built into the ARIMA model.

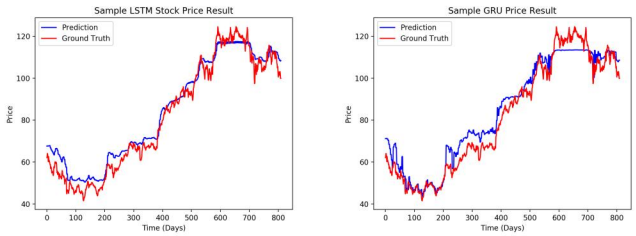


Figure 3: Prediction of Apple Stock Price by LSTM, GRU (Train)

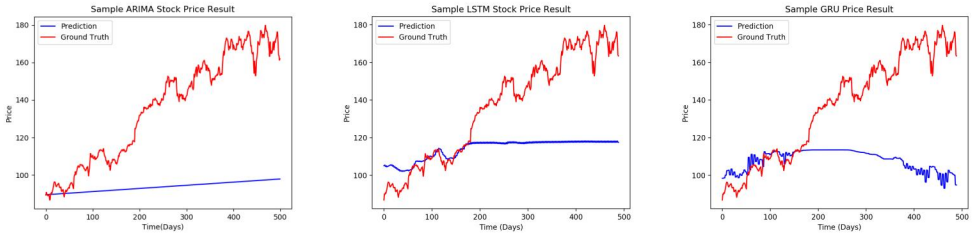
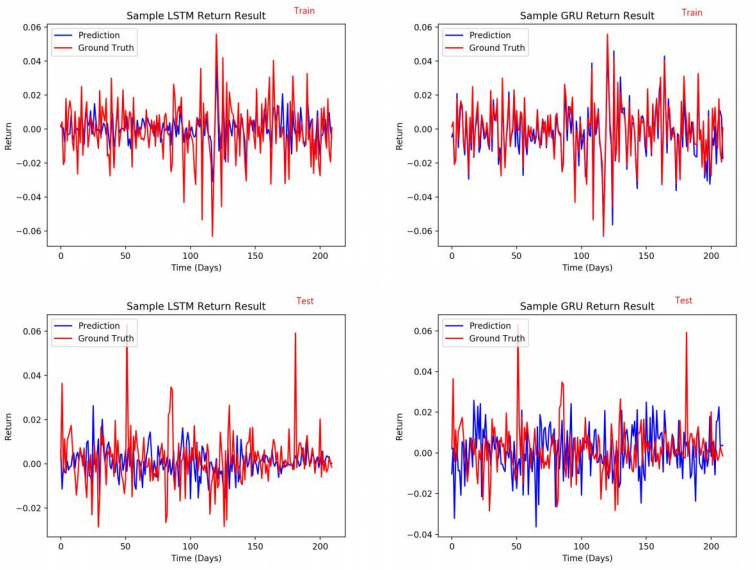


Figure 4: Prediction of Apple Stock Price by ARIMA, LSTM, GRU (Test)

For LSTM and GRU, they do a fairly good job at the earlier periods in the test set, successfully predicting at least the up and down of the stock price. However, the performance drops dramatically as we move further down the time. It becomes a very flat curve, and has very limited predicting power, if at all. On the other hand, they did very well in the training sets (Figure 4 & 5), implying a very serious over-fitting problem. We will talk about this main concern at the end of this section.



For DA-RNN, it follows similar patterns. It does a good job in the training set (Figure 6), but the over-fitting problem still exists. At the very beginning of the test set, DA-RNN can predict fairly good. However, as the error accumulates over time, the difference between actual and predicted increases dramatically. After about 50 days, the prediction stops working. The complicated structure does not provide much advantage, compared to the traditional LSTM and GRU models.

To compare the four approaches, we put their performance in one table, both training and testing. From the numbers, DA-RNN performs the best in terms of test error, and the over-fitting problem

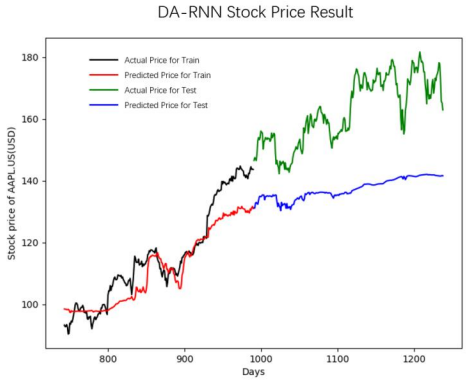


Figure 5: Prediction of Price by DA-RNN

Model	Train Error	Test Error
ARIMA	0.013329	0.014268
LSTM	0.005251	0.022363
GRU	0.009311	0.012347
DA-RNN	0.0098431	0.011279

Table 1: MSE Loss of Different Methods

seems not that serious excpet for LSTM. However, from the plot of predicted stock price and log return, we can see that the over-fitting problem is huge for all the neural network approaches. We tried all the methods that we can think of to reduce over-fitting, including regularization, drop-out, simplifying the architecture. None of them works very well.

We believe that the problem of over-fitting is intrinsic in stock price prediction. Stock market exists for a relatively short period of time, and many stocks are highly correlated, making it very difficult to get large numbers of independent training sets. For example, if considering daily level stock price, a stock has at most a hundred thousand data points. However, the problem of stock price prediction is very hard. You can image that how stock markets worked 20 years ago is very different from how it works today. The difference between stocks and stocks is also huge. A simple neural network with just hundreds of parameters is not capable of making predictions. However, if we switch to complicated neural network structures like DA-RNN model which works well for some other time series prediction problems, we face the problem of over-fitting, because now we need to estimates thousands or maybe millions of parameters, but only having 100 thousand data points. One way to handle this problem is to look at minute-level, or even second-level stock price data. However, we do not have that data nor the computing resource to perform such tasks in this project.

6 Conclusion and Future Work

In this project, we would like to predict future stock price using historical stock prices. We compare the traditional statistical method ARIMA, classical RNN architectures LSTM and GRU, as well as innovated RNN architecture DA-RNN. In general, RNN performs better than ARIMA as RNN allows more flexibility for the time series. However, RNN-based networks suffer high variance/overfitting when we try the models on test sets, because the training dataset is not big enough. DA-RNN has the best performance when predicting stock prices out of the training sets, although the results do not stand out significantly and are far from satisfying. All of the approaches we tried stop producing meaningful predictions a few days after the last observed data points.

For future work, we would like to denoise the dataset to make it less 'uncorrelated'. We will also incorporate the idea of pairs trading where we predict stock price from the same sector as a group. Furthermore, in addition to stock price, we want to add other inputs which also impact stock market significantly such as economic crisis and earning report, and even unstructured data like news titles. Another approach of having more data is to apply our model into minute-level, even second-level stock prices. We hope that with much more data, the problem of over-fitting can be mitigated. This may reduce our prediction power from several days ahead of the market, to several seconds ahead of the market. As long as the prediction works well enough, it is still meaningful and profitable.

7 Contributions

Anran, Zeyu and Huanzhong separate the work evenly and collaborate well. Anran comes up with the initial idea of stock price prediction and helps with baseline implementation. Also, she works on data

visualization and writes up most of the poster. Zeyu processes the data and creates the training and testing module. He works on ARIMA implementation and evaluate the results from predict precision and loss function. Huanzhong works on literature review and figures out related methods, such as GRU and DA-RNN. He implements GRU and DA-RNN as well as result evaluation. Furthermore, he tunes the hyperparameter in order to have a better accuracy. Overall, it is a pleasant time to work in this group and we are looking forward to future collaboration again.

8 Code

`https://github.com/xuhuanvc/stock-rnn`

References

[1] Ayodele Adebiyi, Oluyinka Adewumi, and Charles Ayo. "Comparison of ARIMA and Artificial Neural Networks Models for Stock Price Prediction". In: *Journal of Applied Mathematics* (2014).

[2] Wei Bao, Jun Yue, and Yulei Rao. "A deep learning framework for financial time series using stacked autoencoders and long-short term memory". In: *PLoS ONE* (2017). URL: `https://doi.org/10.1371/journal.pone.0180944`.

[3] Eugene Fama. "Efficient Capital Markets: A Review of Theory and Empirical Work". In: *Journal of Finance* (1970).

[4] Eugene Fama and Kenneth French. "Common risk factors in the returns on stocks and bonds". In: *Journal of Financial Economics* (1993).

[5] Tze Leung Lai and Haipeng Xing. *Statistical Models and Methods for Financial Markets*. Springer, 2008.

[6] *Predict Stock Prices Using RNN: Part 1*. `https://lilianweng.github.io/lil-log/2017/07/08/predict-stock-prices-using-RNN-part-1.html`.

[7] Yao Qin et al. "A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction". In: (2017).

[8] Akhter Rathera, Arun Agarwal, and V.N. Sastryb. "Recurrent neural network and a hybrid model for prediction of stock returns". In: *Expert Systems with Applications* (2015).

[9] Eberhard Schöneburg. "Stock price prediction using neural networks: A project report". In: *Neurocomputing* (1990).

[10] *S&P500 stocks*. `https://finance.yahoo.com/`. Accessed: 2018-11-08.

[11] Lipo Wang and Shekhar Gupta. "Neural Networks and Wavelet De-Noising for Stock Trading and Prediction". In: *Time Series Analysis, Model. & Applications* (2013), pp. 229–247. URL: `https://pdfs.semanticscholar.org/cea1/cd0af8d06aeb69fb524876cc6e3c3f2cce9c.pdf`.