CS230

Generating Probability Distributions for Future Stock Prices

Jeffrey Sun
jeffsun1@stanford.edu

Mohammad Shaib Lari
slari1@stanford.edu

Julius Zhang
juliuszh@stanford.edu

Abstract

This project explores the efficacy of using softmax to generate detailed probability distributions for future stock prices. Our findings show that such a model can generate realistic price probability distributions, with a MSE loss of 0.0815 when predicting among a set of 10 price change ranges. These results build on top of our other experimental findings with various deep learning models.

1 Introduction

The field of financial forecasting represents an area with large amounts of historical data that we can use to test and validate deep learning approaches. We start by building various models that predict future stock price, and we ultimately build a model that predicts future stock price probability distributions. The input to our algorithm is of the shape (n, f, d), where n is the number of stocks, f is the number of features per stock per day, and d is the number of days. We train using n=900, f=8, d=258. We use 2 convolutional layers, 1 fully-connected layer, and a softmax output. The resulting model predicts the future percent change in price on the next day, and assign probabilities to each range of percentage change (the boundaries between each range being -4%, -3%,-2%,-1%,0%,+1%,+2%,+3%, and +4%).

2 Related work

There have been many attempts to use deep learning models to predict stock prices. For example [1] have found existence of nonlinear factors which explain predictability of returns. Very sophisticated models have been built using Deep Learning techniques combining both macroeconomic data and firm-specific information [2].

Using deep learning techniques, researchers have found that inclusion of price and order flow history over many past observations improves forecast dependency indicating that there is path-dependence in price dynamics [3]. Our model similarly uses the trends in previous price movements to learn what price patterns are the best predictors of future price movements.

While we used fully connected and CNNs, we believe that using an LSTM or other recurrent network with attention would likely achieve high predictability. Both [4] and [5] have found that to be the case.

We can also apply our model to intraday price fluctuations to predict prices for day trading following techniques such as those developed in [6]. Finally, a sample trading application can be built similar to [7] to show how a portfolio manager will perform using Deep Learning techniques.

CS230: Deep Learning, Winter 2020, Stanford University, CA.

3 Dataset and Features

The Quandl daily stock price dataset from the Hong Kong exchange was gathered from https://www.quandl.com/data/HKEX-Hong-Kong-Exchange. This time-series data is discretized per day, and preprocessing included cleaning the data to fill in missing values with the previous day's value. Below is an example of the values for one stock indicator:

Date	Price	Bid	Ask	High	Low	Previous Close	Share Volume	Turnover
2018-01-02	1.840	1.830	1.850	1.850	1.830	1.830	2246.0	4155.0
2018-01-03	1.830	1.830	1.850	1.840	1.810	1.840	26.0	48.0
2018-01-04	1.900	1.900	1.910	1.920	1.820	1.830	5395.0	10350.0

4 Methods

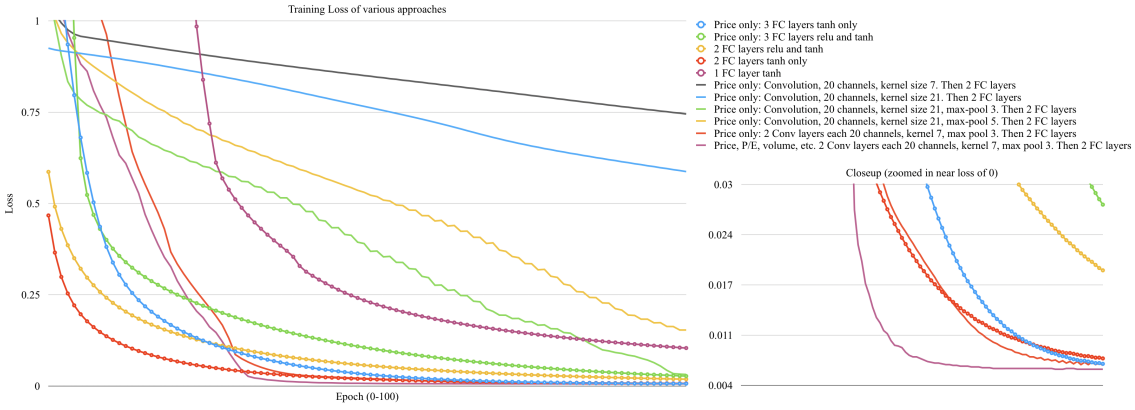
The initial method explored is to take the time-series data and apply various deep learning techniques to predict the future price. The target value is the historical price of a stock on a certain day, and the features used to predict the target value are the daily features from the time-series history from the preceding year. In order to normalize this target value, it is represented as a percent change from the day before, rather than the nominal price. The loss is the mean-squared error between the actual (target) and predicted value. After this method, we then introduce the notion of predicting the price probability distribution, so the target y value changes from a being price change, to being a 10 dimensional one-hot vector to indicate which range the price change is contained in.

5 Experiments/Results/Discussion

Experimentation began with simple fully-connected layers operating on price history only, to determine a baseline. These techniques used a learning rate of 0.0001, and ran for 100 epochs. The experiments were run with a 1-year history (2018) on a small set of 200 stocks, to speed up model structure and hyperparameter experimentation. Early results showed that a model with 3 fully-connected layers would outperform models with fewer layers, and that the tanh activation function was suitable.

Next, experimentation began on convolutional layers. At first, these convolutional layers operated on price history only. A learning rate of 0.0005 was used to speed up training so that comparable results could be achieved in 100 epochs. Various kernel sizes, channel counts, convolution layer counts, and max-pool sizes were used, and 2 convolution layers with max pool of small kernel size was found to be suitable.

To further enhance the model, seven additional time-series features were introduced: bid, ask, P/E ration, daily high, daily low, volume, and turnover. Along with price, we had 8 features, and we fed the expanded feature set into a convolutional model from previous experiments. This new model outperformed our previous models:

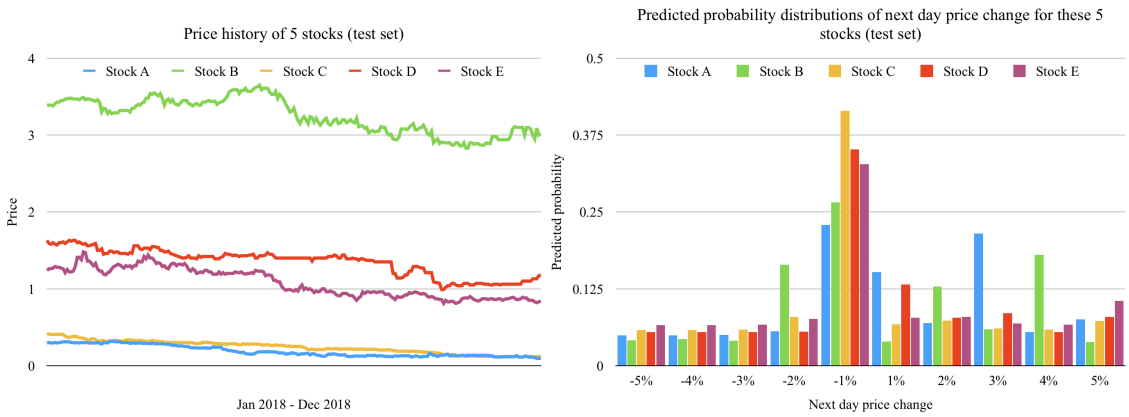


Below are the training, development, and test set losses for the approach with 2 convolutional layers with max pool. The train/dev/test split is 900/300/300 stocks.

Training set loss	Development set loss	Test set loss
0.0184	0.0262	0.0363

Finally, we added a softmax layer to the output, and reformatted the target value to predict which range the price change would be contained in. By analyzing our baseline models from above, we chose the structure of this model to be the best-performing model that we had found before, and we attached a softmax layer to the end of it. This final model has a first convolutional layer of 20 channels, kernel size of 7, stride of 1, with a max-pool of size 3. The second convolutional layer had 1 channel, with kernel size of 7, stride of 1, and a max-pool of size 3. The third layer is a fully-connected layer of size 84, and the fourth layer is fully-connected of size 10, resulting in a softmax of size 10. This softmax-based model ran for 500 epochs, with a training rate of 0.005 using the Adam optimizer. The training set contained 900 stocks, the dev set contained 300, and the test set contained 300.

Below are 5 example stocks from the test set, where the price history is shown on the left as a reference, and the model's predicted price distribution is shown on the right. Additionally, the training, dev, and test set losses are included:



Training set loss	Development set loss	Test set loss
0.0686	0.0826	0.0815

6 Conclusion/Future Work

We find that a deep learning model for predicting stock price (fully connected or convolutional) can serve as a reasonable baseline model. Furthermore, using softmax to predict the discretized range of price change produces reasonable and richer price probability distributions. Our specific results showed a test set MSE loss of 0.0815 when predicting the correct magnitude of percentage changes.

The validity of generating probability distributions in this way has several promising implications, such as being able to model more detailed aspects of risk and price movement. For other aspects of future work, we'd like to expand the number of features used, and extend this predicted probability distribution to generate simulated future price history.

7 Contributions

Jeffrey extracted data from Quandl, ran preprocessing, and built, trained & experimented with fully-connected, convolutional, and a softmax-based models. Shoaib did research on models and algorithms, and identified and resolved issues in our model implementations, including in the filling of missing values, and shape inconsistencies in the models. Julius experimented with tweaking parameters on a set of stock price prediction models.

References

1. Feng, G., He, J., Polson, N. G. (2018) Deep learning for predicting asset returns. arXiv preprint arXiv:1804.09314. https://arxiv.org/pdf/1804.09314.pdf

2. Chen, L., Pelger, M., Zhu, J. (2019) Deep Learning in Asset Pricing. Available at SSRN: https://ssrn.com/abstract=3350138

3. Sirignano, J., Cont, R., (2019) Universal features of price formation in financial markets: perspectives from deep learning, Quantitative Finance, 19:9, 1449-1459. https://doi.org/10.1080/14697688.2019.1622295

4. Yu, P., Yan, X. (2020) Stock price prediction based on deep neural networks. Neural Comput Applic 32, 1609–1628. https://doi.org/10.1007/s00521-019-04212-x

5. Qiu, J., Wang, B., Zhou, C. (2020) Forecasting stock prices with long-short term memory neural networks based on attention mechanism. PloS one, 15(1), e0227222. https://doi.org/10.1371/journal.pone.0227222

6. Naik, N., Mohan, B.R. (2019) Intraday Stock Prediction Based on Deep Neural Network. Natl. Acad. Sci. Lett. https://doi.org/10.1007/s40009-019-00859-1

7. Yun, H., Lee, M., Kang, Y. S., Seok, J. (2020) Portfolio management via two-stage deep learning with a joint cost. Expert Systems with Applications, ISSN: 0957-4174, Vol: 143, Page: 113041. https://doi.org/10.1016/j.eswa.2019.113041

8. Quandl Hong Kong Exchange Daily Stock Prices https://www.quandl.com/data/HKEX-Hong-Kong-Exchange

9. PyTorch https://pytorch.org/

10. NumPy https://numpy.org/