Generating Probability Distribution for Future Stock Prices

Jeffrey Sun, Shoaib Lari, Julius Zhang

CS 230 Deep Learning, Winter 2020

https://youtu.be/NBXE7OOwAMQ

https://tinyurl.com/stkprobdist

Stanford Computer Science

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.

Introduction

We use historical stock market data to predict probability distribution for stock price for a day in future. A probability distribution, as opposed to an expected predicted price, lets one make a more refined judgment call for a buy or sell decision.

Input

We used the Quandl stock price dataset from the Hong Kong exchange. The input features include bid, ask, high, low prices, P/E ratio, volume, and turnover for one full year. 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.

Output

The target price is the predicted stock price for a certain day after the input time-series. In order to normalize this target value, it is represented as a percent change from the current price, rather than the nominal price.

To predict the price probability distribution 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.

The loss is the mean-squared error between the actual (target) and predicted value.

Models

We experimented with many different models. Our 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 has 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.

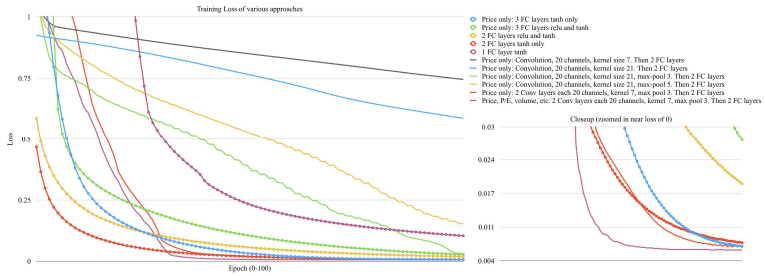
This softmax-based model ran for 500 epochs, with a training rate of 0.005 using the Adam optimizer. The training set contained 900 stocks, the dev set contained 300, and the test set contained 300

Results

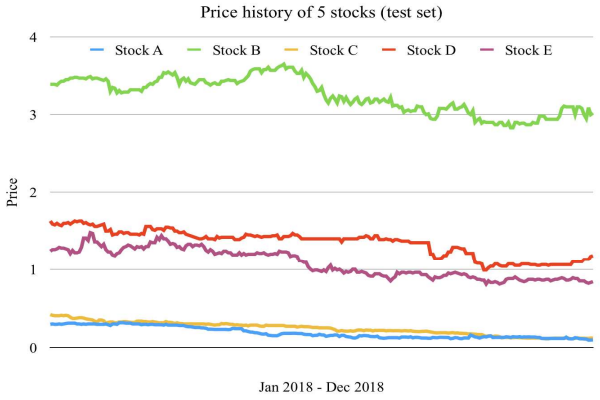
Below are the training, development, and test set losses for the approach with 2 convolutional layers with max pool. The train/dev/test split is 900/300/300 stocks.

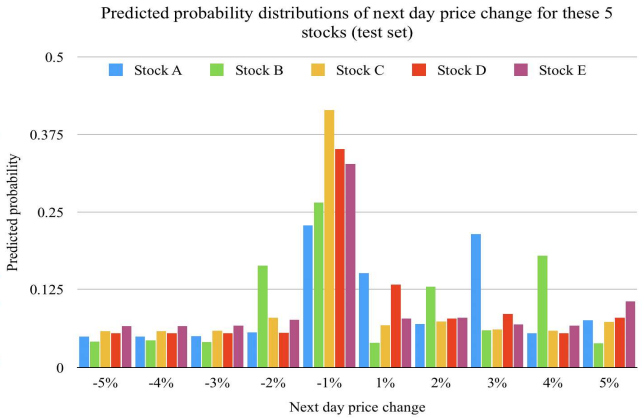
Training set loss	Development set loss	Test set loss
0.0686	0.0826	0.0815

Below are our loss curves for various models.



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.





Stanford University