Rohan Kandi

Jia Cheng Li

Using Historical Data to Predict Stock Price

Introduction

Millions of shares of stocks are bought and sold every day with the sole purpose of making a profit. However, this task can be very complicated, almost to the point of being random as thousands of variables could affect the movement of a stock share. While there are mathematical models to improve prediction accuracy, we are experimenting with Machine Learning. We will be using time-series data for 4 technology stocks- Apple, Amazon, Microsoft and Google- all having a delay between 1-30 days. These companies were chosen because we wanted to focus on technology and experiment with growth patterns for tech stocks, which could potentially be applied to other burgeoning companies. This data was acquired from Kaggle. Our goal is to experiment how far ahead we can predict with each stock using regression and classification, and what are the similarities and differences between the models for each stock.

Approach and Results

Our first experiment was to predict the stock price. While exploring the data, we observed the price trajectory of our 4 sample stocks increased exponentially over the long term. However, in the time span of a month, there is a lot more randomness, which cannot be modeled mathematically. Thus, we hypothesized that using Machine Learning techniques with time series will provide better results. We tackled the problem using Multiple Linear Regression, L1 Regularization, and Convolution Neural Networks. We started by transforming the data using delays. For example, if we wanted to predict 1 day ahead we would shift all the prices 1 row down. We made this delay function and used it to return a delayed version of the independent and dependent variables for the following interval of days ahead: 1, 7, 14, 21, 28. We hypothesised that as the number of days into the future increases, the loss will increase as well. To train and test the data, we used the conventional 70:30 ratio. Since we are using time-series data, the train and test data were not shuffled.

We started by making a simple baseline, where we simply predicted the previous interval's price for the next data point.

	1 day	7 days	14 days	21 days	28 days
Apple	0.27	1.34	2.60	4.56	6.68
Microsoft	0.24	0.74	1.28	1.65	1.97
Google	4.25	12.33	22.52	34.98	43.06
Amazon	2.64	10.41	16.57	23.21	32.35

Table 1: Baseline for prices

We first decided to use multiple linear regression using delay in our time-series data. We expect that this method will give us high loss, but it will serve as a good starting point. We also expect to see that as the interval of prediction gets larger, the delay model will get a better loss than the baseline. To get the most optimal loss we calculated the loss of each delay from 1-30. Using this, we got the best delay for each day we are testing, and for each company. Using multiple linear regression with the most optimal delays we got the following results:

	1 day	7 days	14 days	21 days	28 days
Apple	0.83	2.45	3.68	4.95	6.33
Microsoft	0.39	0.97	1.34	1.61	1.85
Google	6.92	19.65	28.51	38.67	45.79
Amazon	5.85	15.80	22.90	29.01	33.45

Table 2: Multiple Linear Regression results.

The loss was originally scaled to the price of the stock, but we decided to take the mean of the L1 loss so that it would be easier to compare the losses. As shown in the data, the loss seems to increase as the number of days into the future increases. This aligns with our hypothesis, so we decided to reduce the loss for our next experiment.

Since our feature set was getting large (120 features for 30 day delay), we realized there may be overfitting, we decided to use L1 Regularization. We used L1 instead of L2 because L1 fits the graph better when there's more outliers. After applying L1 regularization, most of the loss after the first 7 days ahead decreases. Regularization did not seem to have an effect on the first or seventh day ahead. Only for Microsoft, did regularization not help for all days. Hence, L1 regularization improves generalization if we predict at least 14 days ahead.

We also decided to try and use 1-dimensional convolutions to work on our time-series data. We hypothesized that the convolution might do a better job finding patterns that contribute to the growth and decline to certain prices. The network had one layer of convolution with 1 channel and kernel of size 3. The length matched the size of the features. We decided to mostly work with Microsoft, and tried finding the best loss for all delays for intervals 1 and 7. We also decided to create one model with batch normalization and dropout and one without. While the losses converged most of the time, delays seemed to have a random effect on the loss on the testing set. Tweaking the learning rate led to lowering training loss, however, the testing loss grew suggesting overfit. Batch normalization and dropout significantly improved loss; however, not enough to beat MSFT baseline.

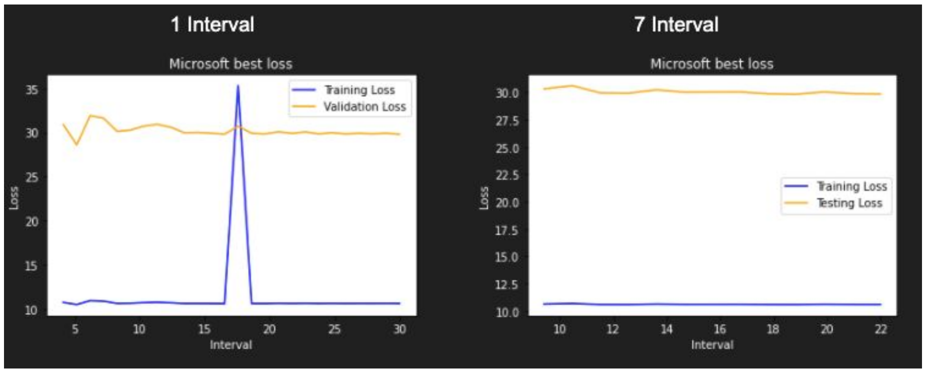

Fig. 1: Without Batch Normalization and Dropout

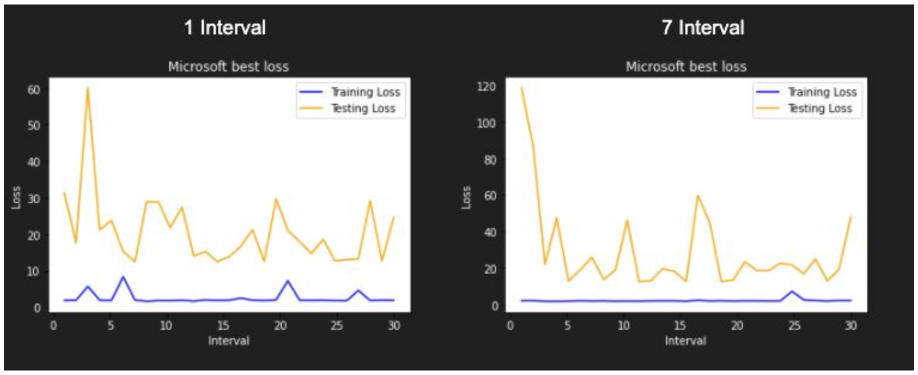

Fig 2: With Batch Norm. and Dropout

For our second experiment, we decided to use Logistic Regression and SVM to make the prediction as we saw that the price models were not always predicting rise and drop in price accurately. For this experiment, we will use accuracy as the metric for comparison. Since our first experiment was to see how well we can predict into the future, we decided to incorporate this into this to see how accurate the models will predict price rise/drop for 1, 7, 14, 21, and 28 days ahead.

For our baseline, we decided to use Logistic Regression since our dependent variable is binary: rise or drop. Since this is the baseline, we decided to use Logistic Regression with LBFGS and no regularization. This will allow us to see if using other models or models with regularization will improve the data. Using this model, we were able to get accuracies of around 50-60%. Additionally, for most of the companies, as the predicted days increase, the accuracy also increases. In contrast, the accuracy for google increased from predicting 1 day to 14 days, but decreased as the days increased.

Using SVM with regularization and rbf kernel, we were able to get relatively the same accuracy as those from the logistic regression. For Apple, SVM performed worse than logistic regression for days 1-14, but performed slightly better for the later days. This trend is also seen in Amazon stocks. However, for Microsoft, logistic regression performed better than SVM for all days. For Google, SVM gave around the same accuracy as logistic regression for days 1-14, but gave higher accuracy for the later days. Hence, logistic regression appears to perform better for Microsoft stocks for all days, and apple and amazon stocks for the first 14 days. While, SVM appears to give higher accuracy for Google for all days. Overall, both logistic regression and SVM gave similar accuracies.

Conclusion

In our first experiment, predicting with delay, we found that L1 regularization did reduce the loss from the baseline for 14 days + ahead, except for Microsoft. However, the regularized loss was not much lower than the non-regularized loss, and the loss only started falling noticeably as the delay got larger. This shows that we have too few features in our dataset and we need to experiment with more features. We also noticed much higher losses for Amazon and

Google than Microsoft and Apple, most probably due to their price values. If we would ever like to compare models (and maybe transfer it to other similar stocks), we will need to center and scale the prices and predictors. The convolutional network did not provide as good of a loss as the baseline. This seems to be because the convolution seems to predict either the average or the median as a fit for the entire model. However, with only a bit of tuning, the loss values were brought down significantly, showing there is significant room for improvement here.

For our second experiment, we found that using SVM only improved the accuracy for all days for Google stocks. As the charts below show, Logistic regression (baseline) seems to be better for predicting days 1-14, while SVM seems to be better for later days. This might have been the case because there seems to be a linear relationship in the earlier intervals between the direction of the stock and the predictor variables leading to similar accuracies, and there's a more non-linear relationship as the interval grows, leading to the SVM performing better.

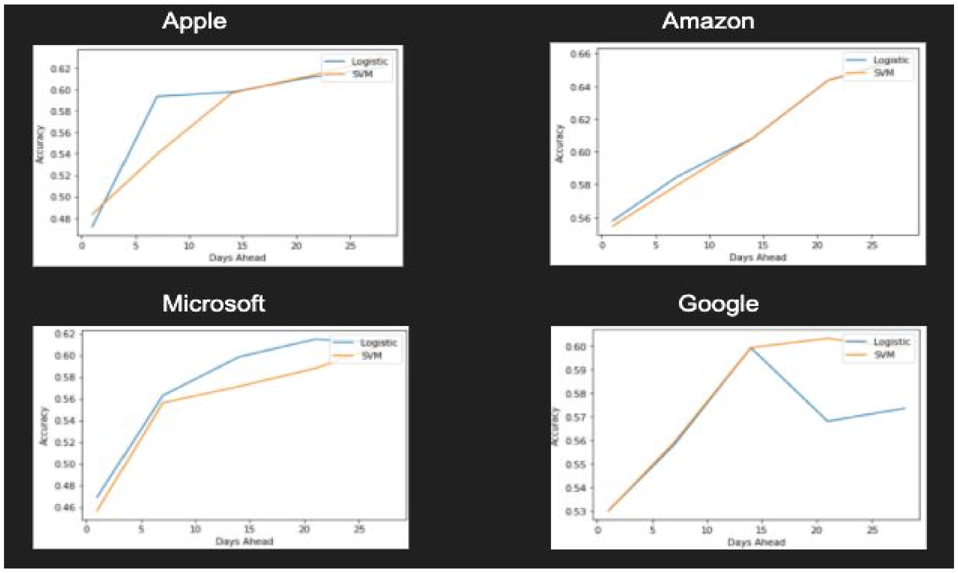

Fig. 3: Classification experiments

If we had more time, we would probably add more features to our dataset, most likely public information. Technology stocks are highly affected by information such as user growth, products sold, costs of operation and cash reserves. All of this information can be experimented with. Secondly, using neural networks could be extremely useful in transforming the data to improve accuracy. For our second experiment, we would also try to implement Kernel Logistic Regression and see if that would have improved the accuracy.