Stock Market Trend Prediction

Using Sequence to Sequence Generation to Predict Future Stock Market Trends

Eric Benson Manner

1 Introduction

The stock market is influenced by many factors and outside forces. Things like a temperamental leader or an overeager tech mogul can cause huge changes in how the prices change from day to day. This makes the prediction of stock market movements incredibly difficult. Large companies such mutual funds take advantage of the general increase in the stock market to make money, but day trading is known to be much more difficult and many often contribute success in this area to luck. This might not be the case, and there are a few people who have found success in day trading.

I propose, as have many others before me, that deep neural networks can prove useful in day-to-day prediction of stock market trends. In this paper, I will analyze and discuss a couple of different Deep Neural Networks often used to leverage time-series data in sequence-to-sequence generation tasks. These networks include a Recurrent Neural Network with a built-in Long Term Short Term Memory (LSTM) gate and a Transformer.

2 Feature Engineering and Data Analysis

The data for this project was obtained *kaggle.com*, but this dataset was compiled from *nasdaq-trader.com*. It contains data of day-to-day stock movements from 1999 − 2020. In creating the model and the model training method, I created feature engineering code and helper functions to format the data in the way that the Pytorch model would need it in. I created methods that do the following: get the data from the zip files; create the sequence (based on a given sequence length) and label, which will be the open price the day following the sequence of data; and return the data in the format [[sequence, label], [sequence, label], . . .]. In each algorithm I used the Adam optimizer.

I also attempted a few different types of sequence-to-sequence type predictors. In the first type, I used an RNN with an LSTM gate to predict the exact day-to-day percent changes. I used the Mean Squared Error Loss in this instance.

In the final two predictor structures, I created a number of classes that represented ranges of day-to-day percent changes. Before embedding, each day in the sequence is a 1-dimensional array of classes. After embedding, each day is a 2-dimensional array of embedded classes. Pytorch's LSTM was not created to handle 4-dimensional input data with dimensions. The 4-dimensions is including the batch dimension. In the second type, I simply flattened each day so the input array was 3-dimensions. Then I used an RNN with a multilayer LSTM gate and Cross Entropy Loss.

Finally, I used an LSTM gate with convolutional layers as the weights *–often called a convolutional LSTM–* to handle the four-dimensional input data. Even though convolutional layers are often used in image analysis, a convolutional layer is a great way to handle large-dimensional input data and is also good at finding related important features in the input data which could be useful in studying stock prices and their movements as related to each other.

While testing of all of these models, the final type was the most successful. Because of the structure of the nature of the problem, it is important to note that predicting a class close to the actual class is much better than predicting one that is far away. In other words, I wanted to penalize class predictions that were not close to the actual prediction as well as the standard penalization for predicting the wrong class. I created a custom loss that used cross entropy, but weighted each term so that classes far away from the actual label were penalized more. For example, with a class label of 2, the custom loss would penalize a prediction of 5 more than one of 3.

3 Convolutional Multi-Dimensional Recurrent Neural Network with LSTM

Seen in the loss and accuracy plots of the neural network, see *Figure 1* we note that after a certain number of epochs, the model begins to over fit on the training data. This has a lot to do with the size of the network and also the number of data that we have in the entire set. There are about 3000 data points. Despite this, it is important to note, that training does improve validation loss and accuracy (up to a point), and the model does learn what it is supposed to.

Displayed in *Figure 2* are the results of training the convolutional LSTM neural network on 74 different stock indices. This resulted in 444 input features. We observe, from the class prediction part, that the model does a decent job of predicting the classes (The accuracy was about 25% on 21 classes). It does significantly better than just random guessing, but also misses most of the time. After generating predictions for 40 days, I also reversed these class predictions back to a percentage by saving the range of each class and selecting a random value in that range, using a uniform distribution. This is the second part of *Figure 2*. The third part is using these percents to generate the actual values.

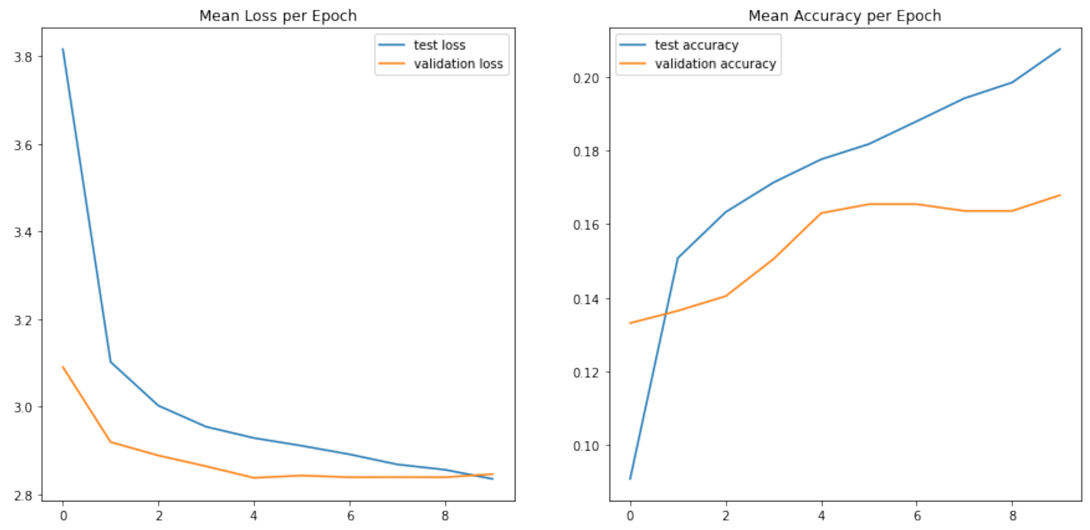
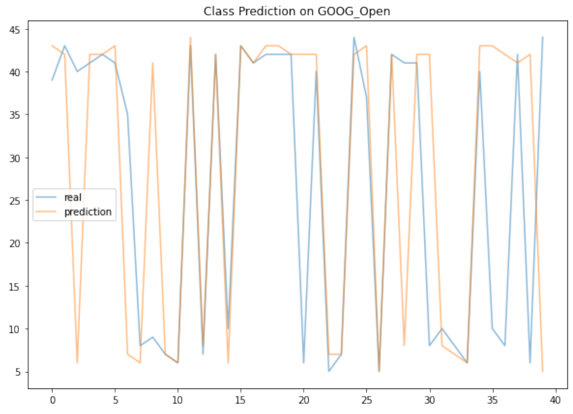
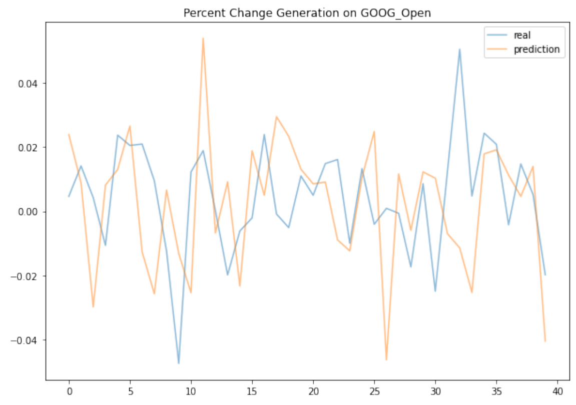


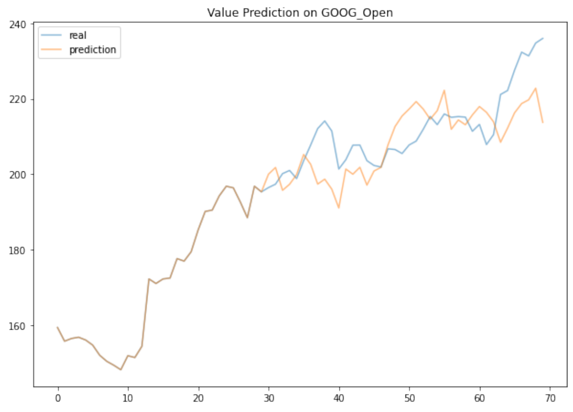
Figure 1: *The train and validation loss and accuracy plots for the stock predictor model*



(a) Class Predictions



(b) Percent Change Predictions



(c) Stock Market Predicted Trend

Figure 2: *Results of the sequence generation for the stock prediction model. This is on the Google open price. The model is not amazing at predicting exact values, but when looking at the general trend of the stock movements, it does a much better job. This could be a result of the custom loss function that I created.*

CS 474 Final Project Time Log

Eric Benson Manner

Stock Market Trend Prediction with Sequence to Sequence Generation

Category	Time Spent	Max/Min
Research/Reading	3:00	5:00
Prep Work	6:00	10:00
Designing/Testing	21:00	20:00
Total	30:00	30:00

Category	Date	Time Start	Time End	Time Total	Comments
Research/Reading	2/19/2021	8:00	10:00:00 AM	2:00	*Researched what I wanted to do and began to formulate idea by writing a research proposal.*
Research/Reading	3/20/2021	9:30:00 AM	10:30:00 AM	1:00	*Found datasets to be used in the project. There were a lot of places to look at, but I just ended with kaggle datasets. There is a lot of stock data out there, but the kaggle datasets allowed for an easier look at all the data in one place. We sacrifice the most recent data for easier feature engineering and data engineering.*
Prep Work	4/10/2021	9:00:00 AM	1:00:00 PM	4:00	*Examined datasets and created robust data selection process for the model. Instead of unpacking entire dataset, we can look at the zip files and select which ones we want to look at. The zipped file is huge and unzipping the entire thing takes up too much memory.*
Prep Work	4/12/2021	9:30:00 AM	11:30:00 AM	2:00	*Wrote the data formatting part of the code for the recurrent neural network. This part normalizes data to percent changes on each feature column, split data into classes based on range, generate [data, label] pairs based on sequence length, and creates the data loaders using pytorch data loader module.*
Designing/Testing	4/12/2021	11:30:00 AM	3:30:00 PM	4:00	*Did some more feature engineering and began testing and iterating on Recurrent Neural Network with LSTM. This was with the Mean Squared Error on the percent change. Cr*
Designing/Testing	4/14/2021	9:00:00 AM	3:00:00 PM	6:00	*Tested and iterated on second model type, reduce percent change values to class predictions (based on range), and flatten before embedding and putting through the Recurrent Neural Network.*
Designing/Testing	4/16/2021	8:30:00 AM	2:30:00 PM	6:00	*Designed custom loss function and also created convolutional RNN to be used for unflattened class predictions. The convolutional RNN can look at interactions between features a little better than a pure LSTM gate. Also, this allows for larger dimensional input so that we do not have displace the features in the time series to fit the desired model.*
Designing/Testing	4/17/2021	9:30:00 AM	1:30:00 PM	4:00	*Iterated on third model type. Wrote sequence generation code for testing the model and created the write up for turning in the final model. Chose the third model type as the one that I would display the results of. It was the most successful and this idea showed promise. I also played around with the class generating method, generating ranges so the Convolutional RNN can identify subtleties a little better and follow the trend of the stock price a little closer.*
Designing/Testing	4/19/2021	10:00:00 AM	11:00:00 AM	1:00	*Cleaned up results and finalized project write up.*
				0:00	
				0:00	