Sequential Model Write Up

Scott Eberle

August 2020

1 Introduction

In recent years, huge amounts of effort and resources have been spent on developing deep learning technologies within the field of machine learning. The idea behind most deep learning technologies is to force computers to learn in the same way that humans learn, by providing them with learning tools that resemble the human brain. Deep learning has shown promising results because researchers have shown, unlike many other types of learning algorithms, as deep learning technologies are provided more data, they produce more accurate results. These ideas seem to suggest if there is enough data available for some phenomena, then it is possible to create a deep learning model that is able to accurately predict this phenomena.

Data on the U.S. stock market is abundant and readily available, which makes it an ideal data source for a machine learning project. In this project, I explored using historical data to predict the behavior of stock values in the future. I used the Keras deep learning library, which is built on the TensorFlow open source library, along with sklearn and pandas for some data manipulation. **Disclaimer: This is NOT meant to be used to make decisions regarding investments or any such actions. This was simply an exercise in machine learning**

2 Finding and Cleaning Data

The data used in this project originated from This Kaggle Data Set. This data set has years of daily opening prices, closing prices, and a few other data points on numerous stocks and ETFs within the U.S. Stock Market. The first step in this process was to determine what data would be used to train our Machine Learning model. The program allows the user to select as many stocks and ETFs as the user would like, as long as the data set contains information on the desired stock or ETF. The user submits the ticker of every stock and ETF and then the program gathers all of the valid data from the data set.

The stock market data was handled using the pandas python data analysis library. First, all of the data was read from CSV files into DataFrames. For the purpose of dimensionality reduction, the program then creates a new set of data that is the change in price of a stock or ETF from when the market opens until it closes. All of the individual DataFrames are then merged into one DataFrame that contains the daily change in stock or ETF price for all of the input data. The user then gives the ticker of the stock that will be the target of the predictions of the machine learning model. The data is manipulated so that the change in value of the ETFs and Stocks in the input data set will correspond with the change in value of the target stock for the following day. The data is offset by one day, so that the machine learning model, can be used to make a prediction of future values. Because changes in the value of a stock or an ETF can vary significantly, the was input data was standardized using the StandardScalar from the sklearn library to obtain more accurate results.

3 The Model and Evaluation of the Model

The stock market data has one input value (the change in price of the stock or ETF) per input stock, and one output value (the change in price of the target stock/ETF), which lends itself well to a Keras Sequential Model. The Sequential Model is essentially just a set of layers with different nodes where the output of one layer becomes the input to the next layer. The model for one test run is summarized below:

Layer (type)	Output Shape	Param Num
dense (Dense)	(None, 9)	99
dense (Dense)	(None, 5)	50
dropout (Dropout)	(None, 5)	0
dense (Dense)	(None, 3)	18
dense (Dense)	(None, 1)	4

A layer is considered "dense" when the output of every node in layer a is connected as an input to every node in layer $a + 1$. This is the most common layer for a neural network, and it is well suited to create complex and successful models. The dropout layer is a way of incorporating regularization into the model. Regularization ensures that the model does not over-fit to the training set by occasionally setting random inputs to have no value. This ensures that no part of the model will over-fit to one particular feature within the data.

The size and shape of this model is somewhat arbitrary. I experimented with different sizes and shapes for the model, and determined that the shape above provided some of the best results with respect to Mean Squared Error (without over fitting to the input dataset). Ideally, the larger and more complex a sequential model, the more accurate it should become at predicting the change

in value of the target stock/ETF. In this case, however, because this program was run on my laptop and without any external GPUs, I was not able to create a massive sequential model that might have made more accurate predictions.

To evaluate the model, I used K-Fold cross validation because there is no easy way to validate the predictions of the model without some sort of data pipeline. K-Fold cross validation first splits the dataset into K different groups. Then, over K iterations, one group is held out as the test group, and all of the remaining groups are used as the input data to train the model. Once the model is built, the error between the model's predictions and the data in the test group is calculated. After all k iterations, the Mean Squared Error, or average of difference between the predictions and the actual data points, is calculated. For example, the Mean Squared Error of one such cross validation is -1.46 with a standard deviation of 0.59, which gives an indication of how well this machine learning model might generalize to making predictions of unseen data.

4 Conclusions

In conclusion, this program is interesting in that it provides somewhat reasonable results regarding predictions of changes in stock and ETF prices. An accurate prediction is, however, dependent on the user having enough understanding of the stock market to be able to select appropriate predictors of the target stock or ETF. A random selection of input data to the machine learning model would almost certainly produce less accurate results. To improve the prediction accuracy of this model would require more data and therefore more hardware resources like GPUs so that the model can be trained in a reasonable amount of time.
One significant problem with any type of stock market prediction tool is the unpredictability of the stock market. There is a notable amount of research on the stock market claiming events that should be extremely rare are actually somewhat common. Because of the general fluctuation and volatility of the stock market, it is important to have an extremely accurate model before using it to make decisions involving real money. While this exercise produces interesting results, using machine learning to make money in the stock market is best left to Professionals.