Machine Learning Engineer Nanodegree

Capstone Proposal

Michael Boker
May 1st, 2017

Domain Background

For a large part of the history of the stock market, math played almost no part in the speculations made by investors. Predictions were made on a qualitative basis, affected by the reputations and characteristics of the people at a business, the product or service sold by the company, and the general landscape of the market at the time. Beginning in the 1950's, quantitative analysis began to make its way onto the scene.[1] This involved mathematical models used to find predictive trends in the market, which could be used to assist in making wise investments. As computers became more capable and ubiquitous, quantitative analysis saw more adoption by investors. Eventually, many of the trader middle-men were cut out, and automated trading began gaining popularity. Now, some estimates say that 70 percent or more of trades are made automatically by computers, with no human involvement.[2]

The analysis behind this automatic trading has mostly been comprised by mathematical models based on trends in the stock values. Quantitative analysis is very powerful, but it cannot completely replace the usefulness of a human in stock analysis. A human can pay attention to the things going on in the world, and provide some insight which may not be found in the numbers. Current events around the world can have a major impact on the stock market, and these effects cannot be predicted by stock values alone until the values are already changing, and it is too late.

Problem Statement

Advancements in deep learning have made it possible to train computers to analyze events in the world, like a human would. Deep neural networks are commonly used for determining the sentiment of a body of text. This would be useful in augmenting the existing quantitative analysis being used today. A model could be trained to learn the correlations between current events and fluctuations in the stock market.

The idea for this project is to use news headlines to predict market behavior. Deep neural networks seem useful for analyzing the news for a day, and predicting the resulting effect on the stock market. Specifically, predictions will be made as to whether the values of stocks would rise or fall in a given day. Ideally, models could be created for predicting the behavior of individual stocks, and predicting the amount of change. However, this project will focus on one of the world’s most popular stock indexes, and will predict a general increase or decrease in value.

Datasets and Inputs

The Dow Jones Industrial Average (DJIA) is a stock index, which incorporates the stock values of 30 of the largest companies in America.[3] An index is the average of a group of stocks. It reflects the overall performance of that group. The DJIA is commonly used as an indicator of the overall performance of the stock market as a whole. The behavior of individual stocks is less predictable than that of a group of stocks, due to the effects of probability being more reliable over the aggregate a group. The DJIA would be a good source of information for testing the usefulness of deep learning in stock price prediction.

A Kaggle dataset already exists for purposes similar to this project, and it contains data from 1990 days, ranging from the 8th of June, 2008 to the 1st of July, 2016.[4] This dataset pairs a collection of the top 25 news headlines from each day with a label indicating whether the DJIA rose or fell that day. The news headlines are taken from the WorldNews subreddit, and are ranked based on user up-votes. This seems to be a pretty good indicator that these are articles that the general public finds interesting. Therefore, they are articles that are most likely to elicit a response from the public, which could have an effect on the DJIA.

Solution Statement

Recurrent Neural Networks (RNN) are a popular deep learning technique for text analysis and can be powerful for sentiment analysis. RNN's can be used to learn context in a body of text, and not simply learn based on the words present in some text, as would be done with a simple feed-forward neural network and a bag of words. For example, given a headline such as "United States Prepares to Wage War on Obesity," an RNN would take the entire headline into consideration. It might be able to determine that the phrase "Prepares to Wage War" is not referring to literal warfare. A traditional feed-forward network would see the word "War" and probably determine that this article is about actual warfare, because it does not consider the surrounding context. The long short-term memory (LSTM) model is a type of RNN, which is a particularly useful tool for sentiment analysis. It can be used to learn from and predict on entire bodies of text, remembering the context of the entire piece as it goes along. Other RNN's have limits in the length of context that they can remember, due to problems encountered during back propagation when too much context is retained.[5] Therefore, an LSTM RNN will be used for this project.

Benchmark Model

For a benchmark model, a feed-forward network trained on vectors built out of a bag of words will be used. This is a commonly used approach to sentiment analysis, and can be quite effective. As mentioned previously, this approach falls short of RNNs in its ability to take contextual information into consideration when determining the impact of a word. Its classification performance will be evaluated by the same metric as the RNN approach, giving an objective baseline to compare the final model against. This model will be trained and tested on the same data used by the final model.

Evaluation Metrics

For evaluating the performance of the models, F score will be used. F score seems to be a good metric for ensuring that a model truly learned from the data, and did not just learn how to cheat from the data. If the values in the training data skew towards one label or the other, and the models learn to be biased towards that label, they will receive low F scores. F score works by taking into account both the precision and the recall of a model. Precision is the probability that, given the model predicts a label, that label is actually correct. Recall is the probability that, given a label is correct, the model predicts that label.

Accuracy score would be a viable alternative evaluation metric. However, there may be a skew towards daily drops or rises in the DJIA, and we don't want the model to learn to favor one classification over the other. Accuracy score can be misleading in cases where the data is skewed in a particular direction. Misclassification in either direction could be devastating for an application that is used to influence market decisions, so precision and recall of both classes will be important. Thus, F score is the best option.

Project Design

For preprocessing the data for this project, some modifications will be made to the headline strings. For each day in the dataset, each of the 25 news headlines will be concatenated into one string, separated by a delimiting value, "[NEW HEADLINE]," which is used to indicate the start of a new headline. All of the strings in the dataset will then have stop words and punctuation removed. Stop words include words such as "a," "the," and "is." They do not bring much meaning to sentences, and there is little that the models can learn or predict from them. This step is particularly important for the feed-forward network baseline model. The stop words could be left for the RNN final model, because the RNN with LSTM units can learn to ignore these words. If we were training the RNN to generate text, it would need these words to be included so that it could learn a realistic syntax of the language. However, since the RNN is only going to be used to classify text, rather than generate it, there is little to be gained from including the stop words. They will be removed to improve training performance for this model.

Once stop words and punctuation are removed, a bag of words will be built with the full dataset. This bag of words will be used to determine the 10,000 most common words for the baseline model. Words towards the end of this range will be examined to see how frequently they occur. If these words do not occur more than once or twice, they will probably not be useful for training. If it so happens that these words occur a very small number of times, the range will be lowered to the 9,000 most common words. This will be repeated up to 5 times (reducing the range to 5,000 words), if words near the range limit do not seem useful. When the range of useful words is determined, these words will be used to build the vocabulary for the baseline model. For the final model, 75,000 words will be used. This dimensionality will be reduced by use of an embedding layer, allowing us to use more of the words in the headlines, while having a manageable dimension space.

Each of the words in the vocabulary will be assigned a numerical index, which will be used to represent that word for the learning models. Before being fed into the models, the rows of data will be shuffled, and split into training and testing sets. Shuffling the data is important, because of the possible presence of trends in the stock market. Shuffling will help our models avoid overfitting to some trend which will not generalize well outside of the training set. The data will be split into 90%

training and 10% testing. The training set will then be divided into 90% training and 10% validation.

For the baseline model, the word indexes we created will be used to convert each string of headlines into a vector. A vector will indicate the frequency of each word in the corresponding string with the number of occurrences of the word placed in the word's respective index in the vector. For example, given a vocabulary of ["she", "knows", "that", "is", "strong"], the sentence, "She knows that she is strong," would be encoded as [2, 1, 1, 1, 1].

These vectors will be fed through a feed-forward neural network with two hidden layers, the first containing 500 nodes, and the second containing 125 nodes. All of these nodes will have a rectified linear unit activation function. The second hidden layer will be connected to a single output cell with a sigmoid activation function. This will be trained with stochastic gradient descent, with batches of 300 days. The number of epochs will be decided by observing when the performance on the validation set stops improving. It will likely be somewhere around 70 to 100 epochs. Different learning rates will be experimented with, and the one that results in the best performance will be used in the final training of this model. Learning rate values that will be tried are 0.05, 0.01, 0.005 and 0.001.

For the final model, the word indexes will be used to create vectors which represent the sequences of words. These vectors will contain the indexes of the words, in the sequences in which they appear in the headline strings, and will each be left padded with zeros to a uniform size. To compare with the example for the feed-forward network, given the same vocabulary of ["she", "knows", "that", "is", "strong"], the sentence, "She knows that she is strong," and a uniform size of 10, the encoding would be [0, 0, 0, 0, 1, 2, 3, 1, 4, 5]. The uniform size should be large enough to fit the contents of most strings, but not unnecessarily large, because this would hamper performance of the model. Upon quick observation, headlines appear to be have a range of about 5 to 45 words, and lengths look to be normally distributed through this range. With 25 headlines for each day, it seems reasonable to make the uniform vector length 650 words. The reasoning for this is that 25 headlines of an average of 25 words equals 625 words, plus 25 words for the “new headline” delimiters.

These vectors will then be passed through an embedding layer, to reduce the dimensionality. This layer will reduce the dimensionality of the vocabulary to 500. The output of the embedding layer will be input into one or two (sequential) layers of LSTM cells. The LSTM layers will most likely consist of 256 or 512 units each. The output of the LSTM cells will be fed into one or two (again, sequential) fully connected layers, of 128 or 256 units each. The output of these layers is finally fed into a single output cell with a sigmoid activation. The decisions between one or two LSTM layers and one or two fully connected layers, as well as the number of cells in each layer, will be made based on experimental performance of the model. One of each layer, each containing the respective lower number of nodes, will be used at first. If the model is underfitting, another LSTM layer will be added. If it still underfits, another fully connected layer will be added. The various combinations of layer sizes will be experimented with to see which result in optimal performance.

This model will be trained using the same procedure as the baseline model. Specifically, stochastic gradient descent with batches of 300 days will be used. Learning rates of 0.05, 0.01, 0.005 and 0.001 will be tested.

Citations

1. McWhinney, James E. "A Simple Overview of Quantitative Analysis" *Investopedia*
2. Salmon, Felix and Stokes, Jon "Algorithms Take Control of Wall Street" *WIRED*
3. "Dow Jones Industrial Average" *Wikipedia*
4. User Aaron7sun "Daily News for Stock Market Prediction" *Kaggle*
5. Olah, Chris "Understanding LSTM Networks" *Colah's Blog*