Lucas Ege
CS229
Final Project Report

Predicting Bitcoin Price Fluctuations Based on News Headlines

Abstract

Bitcoin has recently become a household name due to its incredible growth, unpredictable volatility, and interesting applications. This increase in interest in the public world has brought increased interest in the data science world, best seen by the amount of classifiers built to determine bitcoin price fluctuations already. These classifiers, like Madan, Saluja and Zhao [1]'s for example, have managed to attain high accuracies in the past by focusing on purely economic data regarding the Bitcoin network - average hashing, number of transactions, etc. In order to distinguish this model, the focus was put almost entirely on the effect of news on the price of bitcoin, under the assumption that bitcoin's price - especially in the last year - is extremely influenced by "hype" which is fueled through what consumers read as the "state" of their world or of Bitcoin specifically. This model utilizes NLP techniques to analyze headline data for the past two years (2016 - 2017) to then feed a neural network to predict the fluctuation of bitcoin in the next day. With a limited training/test set of ~640 examples, the model was able to attain 64% test accuracy in predicting the sign of the change in bitcoin's price in the next day.

Introduction

As detailed in the related work section, there has been significant effort put into predicting bitcoin prices using historical economic data. Bitcoin has become an increasingly important part of economic analysis because it has become the figurehead for the entire cryptocurrency market. While this market of cryptocurrencies is still untested and unverified (as a viable currency) it is undeniably a powerful concept backed by solid technology (blockchain). As a result, predicting this market is arguably as important as predicting the traditional stock market. Besides this, the capacity to make unbelievable returns on investment has also been a key point of interest. The tested assumption of this model was the impact of "hype" on the price of bitcoin, especially in the last year of extremely volatile behavior. This hype was measured using sentiment analysis on a dataset of historical news headlines. Using a sentiment analysis library, TextBlob, I was able to generate a "sentiment" score for each headline, and then weight this score by multiplying in an "objectivity" score from the same library. This gave me a feature indicative of "objective positivity" for each headline, which I was then able to average across all headlines for a given day to get a single input feature for every data point (a single day). This was my most critical feature and consumed most of my development time (as described in the discussion section). Because I wanted to even out my feature set and input more to the network, I also pulled in some very basic historical data on bitcoin, specifically: previous day's close, previous day's volume, and the label for the previous day (the closing value minus the opening value). I also used the previous day's "objective positivity" score as a feature. In an attempt to boost my results, I also added data from Google trends, by pulling

analytics for the search term “Bitcoin,” and using the weighted values for “activity” across the days as another feature. The output of the model is defined as the sign change of bitcoin’s price for that day, which I derived for historicals by subtracting the opening price from the closing price.

Related Work

Now that Bitcoin has cemented itself as the de facto figure head for cryptocurrencies, the interest in predicting and modeling its behavior has skyrocketed. Work in this area has seen a dramatic increase in recent years, which I will detail briefly. An earlier model in 2015 by Alex Greaves and Benjamin Au [5] was only able to achieve accuracy of 56% using neural networks to predict bitcoin price fluctuation using blockchain specific data. As mentioned earlier, Madan, Saluja and Zhao [1]’s model in 2014 showed much more impressive accuracy of 98.7% in predicting bitcoin price fluctuations on a day by day basis, and competent accuracy (50-55%) when narrowed to a 10 minute widow. Hegazy and Mumford’s paper [2] attempts many different techniques for automated bitcoin trading, finding that Boosted Trees provided the best test and training accuracy for their data set. More recently, there has been some significant work using a Long Short Term Memory model to predict bitcoin price fluctuations. Jakob Aungiers [3] was able to develop a model that closely matched the direction of Bitcoin’s price, but was even more volatile than the true fluctuations. Derek Sheehan’s model [4], also using LSTM models, was able to achieve a greater representation of the true fluctuations, eventually arriving at an average error of 0.04 and 0.05 for Bitcoin and Etherium, respectively. While all of the models were able to achieve solid representations of the Bitcoin market, I thought the work done by Madan et al. [1]’s modelling was the most impressive and therefore the best place to focus my efforts. It would seem the “state of the art” (determined by the topics of recent papers) involves these LSTM models, but these (from my understanding) are molded specifically for time series data much more granular (by minute/second) than my data (daily).

Dataset and Features

This model utilizes three main sources of data to detail historical Bitcoin data, historical news headlines, and search trends regarding Bitcoin. The first two data sets (Bitcoin historicals and news headline historicals) were taken from Kaggle from the sets titled, “Cryptocurrency Historical Prices,” and “A Million News Headlines.” Although both datasets provided data before 2016, I chose to restrict the data set to 2016 and on, since I believed this was the period most influenced by “hype” and most characteristic of the extreme inexplicable volatility we have seen. Both of these sets were on a daily basis, so my first step was to match the two datasets by joining on the date. News headlines were separated into multiple rows of headlines for the same day, so after joining I had to combine these headlines into one metric for each day. My initial efforts focused on utilizing a “Bag of Words” representation where each word is represented by its own feature and the data is the frequency of each word in the summation of all headlines for that day. I also tried to use a Word2vec model to generate these features.

However, as I described later in the Discussion section, this model was flawed in part because it created a feature set far larger (>40,000) than the size of my example data set (~650). My final model utilizes a NLP library, TextBlob [6], which provides me with both a "sentiment" score and an "objectivity" score for each headline's text. I then averaged these scores over all the headlines for a given day, and utilized this as a feature. The Bitcoin historicals were much easier to work with, as I simply used their values (normalized in order to be represented efficiently) as features. In an effort to increase accuracy, I also pulled in data from Google Trends by searching on the key word "Bitcoin" to try and model this "hype" behavior better. The data set from this consisted of weekly "weights" for the popularity of this search term over the entire period, which I then extended to each day and added as a feature. I was also able to use the Bitcoin historicals to generate a label defined to be 1 if the price went up that day, and 0 otherwise. By the end of it, an example data point looked like this:

Avg Sentiment Score	Prev. Day Bitcoin Close	Prev. Day Label	Prev. Day Sentiment	Prev. Day Adjusted Volume	Weighted Sentiment Score	Google Trends Rating	Label
1.39543220e-02	4.10052000e+03	1	-3.09499800e-03	3.76424000e+04	3.63496351e-01	11	0

Methods

For this task, I utilized many of the canonical classification models available to us. To start, I used a logistic regression on my bag of words feature set to try and classify the sign change of bitcoin for the coming day. Logistic regression is a supervised learning algorithm that works by finding the best fitting weights to model a relationship between a set of input features and their corresponding set of labeled outputs (of which there are two possible options: 1 or 0). This ideal weights are found by utilizing some form of gradient descent (stochastic or batch GD, for example) to minimize a loss function or maximize a likelihood function (of the data occurring). The canonical likelihood function for logistic regression is the log-likelihood, defined as:

$$\begin{aligned}\ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))\end{aligned}$$

Figure 1.

Where h(x) is defined as the logistic function of the weights dotted with the input. I ended up feeding this model a vector of >40,000 features, one for each possible word in all of the headlines, which was then modeled with >40,000 weights, one for each feature, which were

then optimized using stochastic gradient descent to model the data I input. This model gave poor testing results, but great training results, indicative of high bias and overfitting. Displeased with my results here, I switched to using the sentiment analysis framework I previously mentioned to define a feature, meaning my feature space is now a much more sensible 7 features large. With this smaller feature space, I then ran a linear classifier to model this new feature set. Linear classification works similarly to logistic regression except it produces a continuous output, rather than a discrete output (like 0 or 1 in the previous example). My hopes were to develop a classifier for the change in price of bitcoin over a day, but this was met with very poor results, despite my change in feature set.

From this point I decided to delve into using neural networks to model this relationship. Specifically, I used the Deep Neural Network classifier provided by TensorFlow. This Deep Neural Network includes multiple layers of linked "neurons" which each have weights from each input feature or previous layer to themselves. These weights are then dotted with the input feature values or the value of the previous layer, and then passed through an "activation function" to normalize the values. The hopes with this model is to find some relationship more complex than linear that can be represented through this more verbose model. The strictly larger model also allows for much more fine tuning, meaning we can express more complex relationships, or multiple linear relationships in one model, which was the hope in utilizing this model.

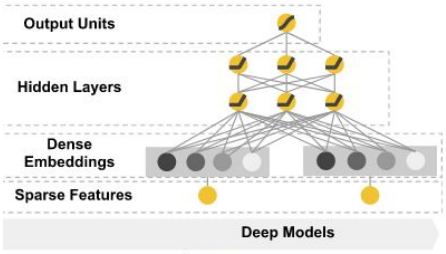


Figure 2.
Credit to: tensorflow.org/tutorials/wide_and_deep

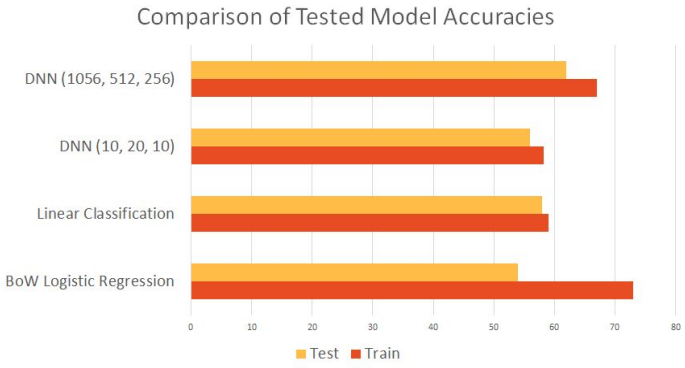
Experiments/Results/Discussion

In an attempt to test my assumption that the fluctuations in bitcoin price could be found to be represented by the "hype" of news headlines, most of my work was spent in building features related to my news headlines dataset. The main metric I used to determine success was accuracy, since in this classification situation with a binary output, accuracy is very telling of the model's performance. A big rabbit hole I fell down was trying to model this by using the "Bag of Words" feature representation, where each word is represented by frequency used in each headline. I ran this with my logistic regression, but was only able to attain 54% test accuracy, and 73% training accuracy. This large difference in accuracies was likely due to the huge feature set overfitting to my small example set, which set off red flags. I tried a couple other methods to get this method to work, such as removing "stop words" like "the" from my dataset, and utilizing some frameworks like Word2vec to try and model these word features more efficiently. However, I was unable to improve these results in any significant way, and concluded that this representation was not really representative of the relationship I was trying to model.

So, despite investing significant time in this method, I decided to try a new route by utilizing a sentiment analysis framework, TextBlob, to provide the scores I mentioned earlier. With this new feature set, I ran linear classification which attained close to 0% accuracy since it was

trying to model continuous values. I switched this to a logistic regression and attained 58% test accuracy and 59% training accuracy. While these accuracies were not good enough for my test (hence the rest of my expansion into using other models), I was happy to see that the bias was small and pointed to the model likely not overfitting.

From here, I moved to utilizing a Deep Neural Network with the same feature set. My initial implementation used a network model with 3 hidden layers, each having 10, 20, and 10 hidden units (neurons), respectively. With this preliminary and small network, I was only able to attain 56% test accuracy and 58.2% training accuracy. I then "beefed up" the network by adding more neurons in each hidden layer, up to 1024, 512, and 256, respectively, and was then able to attain only 0.69 average loss for the test set and 0.78 average loss on the test set, with 62% test accuracy, and 67% training accuracy, the highest values I was able to achieve. The small difference between test and training accuracy also suggests minimal overfitting in this expanded network. I found that adding or removing hidden layers from this only lowered my accuracy, and the same was true for the limited amount of changes in the counts of neurons per layer that I attempted. Most importantly, I found that when I removed the feature for sentiment score that my accuracy was about 4-7% lower for the test set, and the average loss for the test set without sentiment scores was 1.179, significantly more than with that feature, meaning the sentiment analysis feature had a significant impact. The accuracies of these models are summarized in the graph above.



Conclusion

While my accuracy and loss values never achieved the heights I was hoping for, I do believe I was able to successfully test my assumption that the price of Bitcoin is influenced by the hype of news headlines. I was able to achieve the lowest loss on the test set and the highest accuracy when I used a Deep Neural network, which was to be expected when compared to simpler logistic and linear regressions since this deep network can be described as layering multiple of these simpler algorithms together and interleaving their results. If given more time and resources, I would like to investigate other ML techniques, such as the LSTM models I discussed in the Related Works section, or expanding my feature set to better approximate bitcoin specific sentiment using custom NLP techniques. I do believe that the hype of the modern news really influences the price of bitcoin, but I can't deny the fact that the basic economic features related to block level transactions provide a very representative, if not more representative, model of the behavior of bitcoin's price. Thus, I think a model combining all of these features using a more specific algorithm would be able to provide a great representation of this market, if given the right data and time to develop.

References:

[1] Madan, S. Saluja, and A. Zhao, “Automated Bitcoin Trading via Machine Learning Algorithms.” [Online]. Available: http://ai2-s2-pdfs.s3.amazonaws.com/e065/3631b4a476abf5276a264f6bbff40b132061.pdf. [Accessed: 14-Dec-2017].

[2] Hegazy, K. and Mumford, S. (2016), “Comparative Automated Bitcoin Trading Strategies.” [Online]. Available at: http://cs229.stanford.edu/proj2016/report/MumfordHegazy-ComparitiveAutomatedBitcoinTradingStrategies-report.pdf [Accessed 14 Dec. 2017].

[3] J. Aungiers, “Multidimensional LSTM Networks to Predict Bitcoin Price,” Jakob Aungiers, 15-Jul-2017. [Online]. Available: http://www.jakob-aungiers.com/articles/a/Multidimensional-LSTM-Networks-to-Predict-Bitcoin-Price. [Accessed: 14-Dec-2017].

[4] D. Sheehan, “Predicting Cryptocurrency Prices With Deep Learning,” dashee87.github.io, 19-Nov-2017. [Online]. Available: https://dashee87.github.io/deep%20learning/python/predicting-cryptocurrency-prices-with-deep-learning/. [Accessed: 14-Dec-2017].

[5] A. Greaves and B. Au, “Using the Bitcoin Transaction Graph to Predict the Price of Bitcoin,” stanford.edu, 08-Dec-2015. [Online]. Available: http://snap.stanford.edu/class/cs224w-2015/projects_2015/Using_the_Bitcoin_Transaction_Graph_to_Predict_the_Price_of_Bitcoin.pdf. [Accessed: 14-Dec-2017].