Predicting Bitcoin Price Fluctuation Based on News Headlines

Lucas Ege
Stanford University

Predicting/Motivation

When this project was proposed, Bitcoin was just in the middle of its meteoric rise (~$5,000 compared to $16,000 today). I wanted to see if I could build a model to predict Bitcoin's price fluctuations. Specifically, I wanted to use news data (specifically headline samples) as features, because an assumption I held was that the behavior of buying/selling Bitcoin is largely influenced by "hype" which is driven by news.

Data

Data was gathered from two main datasets – "Cryptocurrency Historical Prices" and "One Million Headlines", both sourced from Kaggle. A lot of work ended up being put into parsing and forming this data into a useable form – I had to append both data sets where the features included sentiment analysis of the headline data and the outputs were sourced from the cryptocurrency fluctuations.

Features

I currently use five feature inputs in my model, three of which are directly from the data : previous day's volume (adjusted to storable size), previous day close, previous day fluctuation (close – open). For the other two features, I utilized a sentiment analysis package, "TextBlob," to extract a "sentiment score" for each headline, generating two features – current day's headline sentiment score averaged over all headlines, and the previous day's average sentiment score.

My initial models used a feature set best described as the "bag of words" model, i.e. keeping track of how many of each word were used in headlines for a specific date. This, however, provided poor results.

Models

Linear Classification:

With an input vector of 5 features, the model performs gradient descent to minimize loss (squared mean loss) over the input.

Deep Neural Network:

Utilizing forward propagation and backwards propagation, the model minimizes loss similarly to linear classification, utilizing many more dependent weights along different layers (these layers being one of the main factors I tweaked to improve performance).

Model	Train Accuracy	Test Accuracy
Bag of Words Features		
Logistic Regression:	73%	54%
Current features		
Linear Classification	59%	58%
DNN layers (10, 20, 10)	58.2%	56%
DNN layers (1024, 512, 256)	67%	62%

My training set contained 500 examples, while the test set contained 168 examples, roughly an 80-20 split of my available data.

Future Work

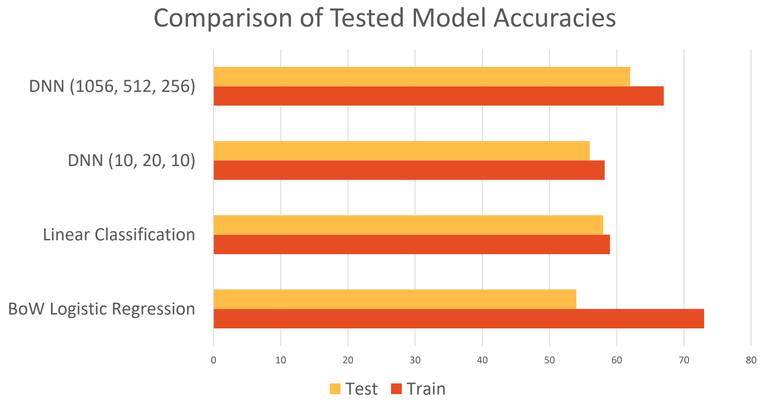
If given more time, training the network with more features from more data sets would likely provide great returns. Specifically, I think taking data from a resource like "Google Trends" or similar analytical platforms that can target specific key words (like Bitcoin, or the crypto market in general). I would also like to train the model on the data from very recent months, because I believe that these past months have been most influenced by "hype" so far in the Bitcoin lifespan. I do believe the model could be greatly improved upon to reach much better accuracies (such as in Madan et al's paper, "Automated Bitcoin Trading via Machine Learning Algorithms"), however since I chose for my project to focus mainly on the idea of news headline data influencing the price, my results weren't nearly as good as theirs. In that vein, I believe focusing on feature sets pertaining mainly to financial data – specific transaction level data for Bitcoin, number of unique buyers of bitcoins, and the hash rate of the Bitcoin network – would provide greater accuracies.

Discussion

A costly mistake on my part was focusing a large portion of my initial efforts on building a model around a "bag of words" feature set, which ended up being a huge rabbit hole that only served to stunt my project's progression. On this note, another dissatisfying use of my time was taken parsing the datasets into a usable form, and then matching the two datasets, and crafting features from them.

I instantly saw better results when I adopted the use of the library "TextBlob" which provided a metric, "sentiment" given an input of a string of words, which I then fed each headline for a specific date and averaged over the amount of headlines to get an "average sentiment" for any given day. This was done under the assumption that "positive" days might persuade the market to buy Bitcoin, and vice versa.

Results were better under this model, as well as when I introduced a neural network, but still ended up being just slightly better than random guessing. This did not reach my expectations for my goal with the project because I ended up needing to test an alternate assumption (regarding overall sentiment of news in general, not just bitcoin news) and the results were not exceptional. Another confounding factor in my accuracies is how volatile the market has grown, even just in the past year, making it much more difficult to predict.



References

[1] TextBlob, "Sentiment Analysis", Online.
[2] TensorFlow, "Deep Neural Networks", Online.
[3] Madan, Saluja, Zhao, "Automated Bitcoin Trading via Machine Learning Algorithms"

Contact
Lucas Ege
Stanford University
Email:lucasege@stanford.edu