Bitcoin Transaction Patterns

Problem Formulation

Bitcoin's price relative to the US dollar is extremely volatile. The amount of Bitcoin actually being traded can also be very volatile. As a result of this volatility there is a wide range of price points and exchange rates to observe in the history of Bitcoin. We want to see how people are using Bitcoins and when people decide to use them.

The Bitcoin blockchain contains information of how much Bitcoin is exchanged, when it is exchanged, and to and from what wallets for every exchange made with Bitcoin from its genesis in 2009 (Frequently). Using all of this data, we can effectively look at how Bitcoin is moved from wallet to wallet and hopefully discover patterns in this movement which can give us insight to the Bitcoin network. These patterns are interesting from a social standpoint because it can provide insight on how people use the currency relative to traditional methods.

Our problem is important from an economic standpoint because it is the largest publicly accessible ledger and includes data that is not normally accessible to companies such as visa such as peer-to-peer transactions and black market trade. Such analysis while limited to the Bitcoin network also could be relevant to traditional currencies. According to Forbes, a man referred to as Mr. Smith made nearly 25 million United States Dollars from Bitcoin trading in 2013. The potential for this kind of profit is impressive enough to make Bitcoin blockchain analysis worth exploring, it can help show how people spend currencies with rapidly changing or inflationary prices.

The sheer size of the blockchain makes this an interesting Big Data problem, the entire blockchain is about 250gb of data (Blockchain Size). We, however, will only be using a roughly 100 gigabyte subset of the data starting from 2017. 100 gigabytes of data which translates to hundreds of thousands of transactions is still enough to label this as a Big Data project. The decision to subset the data is twofold. The first is to minimize running times due to project time constraints. The second reason is so that early bitcoin transactions, which represent a very different context than transactions after the massive trader speculation took over the market, do not influence our results as much. Using the information of when and how people move Bitcoin can help us look at the habits of Bitcoin traders, exchanges, washer, and normal people. This information could be used by anyone who is interested in the spending habits in differing economic scenarios and provides potentially useful insight into the development of other blockchain based technologies.

Methodology to Solve Problem

After looking at the problem several times and looking at the feedback, we decided that we would be using Scala and Apache Spark to help us with this project. We decided that Apache Spark would be best to use for efficiency in speed and Scala would complement it nicely since Apache Spark tends to react well to Scala. As our solution had an “iterative” approach utilizing Spark is more efficient than MapReduce as data stays in memory between each iteration instead of needing to be output to disk.

We decided to utilize the mean-shift clustering algorithm. This allows a natural number of clusters to emerge from the data without us specifying beforehand. The algorithm works by shifting every point to the mean of all points within it’s “neighborhood”. After enough iterations all the points end up stacked into discrete Gaussian clusters. From these clusters we can look at the wallet addresses that are associated within each of those clusters. Using addresses of transaction histories of known wallets we can see what percentage of those addresses fall within each of those categories.

As far implementation. There are several functions that would be written out in Scala. The first function is joinfunc, which joins individual transactions into wallet information. The value stored is the input total, output total and the number of transactions. The piece is shifting which has a couple of parts. The first is to find the neighbors within a defined distance using Euclidean distance. Then, once each neighbor is found we calculate a weight for the point utilizing a gaussian kernel. This generates a probability distribution for our points which is then used for calculating the mean point. The gaussian kernel has a parameter bandwidth which is a smoothing parameter, a smaller bandwidth will lead to more clusters. And then, each neighbor point is multiplied with the weight variable to generate a new shifted point.

The gaussian kernel function was written in according to the mathematical formula and gaussian values were collected on a per point basis to generate n probability functions. Our final function is called notClustered. This function is an alternative method than specifying iterations to stop looping. It checks to see if points are still moving or if clusters have been reached.

There is another file within our project called Bigquery.py. This is a python script that will help with reading in the dataset and making sure that it’s formatted correctly as we will read the data in with Scala. This file allows for the input data for walletClustering.scala to be smooth and require little to no filtering of that data.

In addition we used a bash script by the name of walletGatherer.sh to obtain the wallet history of a few known wallets. These wallets are each from one of four categories, exchange wallets, pool wallets, service wallets, or gambling wallets. The most wallets with the most transactions in their histories from each of those wallets were selected in order to give us a wider range of addresses to use when verifying for correctness from the output of our program.

Dataset

https://www.kaggle.com/bigquery/bitcoin-blockchain

We will be utilizing the Bitcoin blockchain pre-processed by Google Cloud engineers Allen Day and Collin Bookman. This data is hosted on Google's BigQuery API and we will be extracting data from the beginning of 2017 to current day approximate size of 100gb.

The following description is stated.

"In this dataset, you will have access to information about blockchain blocks and transactions. All historical data are in the bigquery-public-data:crypto_bitcoin dataset. It's updated it every 10 minutes. The data can be joined with historical prices in kernels. See available similar datasets here: https://www.kaggle.com/datasets?search=bitcoin.

You can use the BigQuery Python client library to query tables in this dataset in Kernels. Note that methods available in Kernels are limited to querying data. Tables are at bigquery-public-data.crypto_bitcoin.[TABLENAME]."

In the dataset, there are 13 columns of blocks, 14 columns of inputs, 11 columns of outputs and 17 columns of transactions. The columns have several variables. The hash variable is the has of the block. The size is the size of the block data in bytes. The variable, stripped_size, is the size of block data in bytes excluding witness data. The weight variable is three times the base size plus the total size. The number variable is the number of the block. The version variable is the protocol version specified in the block header. The variable, merkle_root, is the root node of the Merkle tree, where the leaves of the tree are transactions hashes. The timestamp variable is the block creation timestamp specified in the block header. The timestamp_month variable is the month of the block creation timestamp specified in the block header. The nonce variable is the difficulty solutions specified in the block header, while the bits are the difficulty threshold. The coinbase_param variable is the data specified in the coinbase transaction of this black and lastly, the transaction_count variable is the number of transactions included in the block.

In addition to the pre-processed Bitcoin blockchain wallet histories from WalletExplorer.com were used to determine the correctness of the output of our program. Each of these cvs for the history of a wallet contains information about the date of a transaction, the address of the wallet that is receiving or giving coins, the amount of coins being sent or received, the balance of the wallet, as well as a transaction number. We used four csvs total, the history for the Bittrex wallet for the example of an exchange wallet, the history of BTCCPool wallet for the example of a pool wallet, the history of Xapo for the example of a service wallet, and the history of LuckyB for the example of a gambling wallet. This wallets were selected because they had the greatest amount of transactions for each of the given categories, giving us more data to compare our results to. From all of the data that was given from each of these, we were mainly interested in the addresses of the wallets that were sending and receiving Bitcoins from there wallets, as a result we filtered out everything that was not a wallet address and we kept no duplicates. We also discussed using additional wallets from each category in order to increase the number of wallet addresses available to evaluate the output of our system, but we ultimately decided that each of the category wallets we used were sufficiently large.

Discussion and Analysis

As a result of consistent network, cluster, and technical difficulties we were unable to get any form of results from our project. We hoped we would be able to use the information from wallets about the amount of money transferred to and from wallets as well as the number of transactions to see if we could categorize the wallets. This would have helped find the attributes that should be observed when trying to determine wallet behaviors from the blockchain. In addition we would have also been able to adjust the size of the kernels while doing shift means in order for them to be more or less inclusive.

Because a result was not able to be produced, we are unable to evaluate our approach. Had we have gotten results from the project, we would have read in the addresses of known wallets in categories such as gambling wallets and exchange wallets. We then would have seen what percentage of each of those wallets in those categories were in each of the clusters that were generated from our program. Using this information we could adjust the parameters we have been using for clustering and attempt to have better defined categories. This way unknown wallets could be marked as potentially belonging to one of these categories for further inspection.

There were several problems that occurred during our project. We had several issues working with Amazon Web Service and making a cluster with them, therefore we started to use a Hadoop Cluster with Apache Spark on the Colorado State University network machines instead

to solve that problem, but it took a long time to implement. With HDFS, some of the group members weren't able to upload any kind of files and or read them in. There was also an issue with Apache Spark, where it stopped working for some of us and refused to start. Jake had issues with his cluster, where it would be permanently stuck in safe mode. Kurt had issues with the cluster as well. Whenever Kurt ran anything, it would fail due to a communication error with the block manager. At one point, when we were trying to read the dataset, it would give us invalid data with extra special characters in between every letter and space. This was because the source file generated when downloading from Google's BigQuery was encoded as utf-8 instead of ASCII. There was also problems in accessing the spark cluster through ssh. Suchi had problems with accessing the spark cluster overall and running the code locally wasn't helpful in understanding the code and whether it worked or not.

Project Contributions

Suchi: I would like to acknowledge that I wasn't much help during the project since I was significantly busy and didn't completely understand what we were trying to do. I acknowledge that I didn't help in the coding at all. There was an effort to help with the code, but it didn't work or help to solve any of the problem. To add to the problem, my cluster was down and Apache Spark did not work for me at all. However, I hope to make up for that absence by writing the report and by getting the presentation started as well. I think we worked individually rather than as a team, but that's no excuse for my absence.

Kurt: I wrote the script which fetches data from Google's BigQuery and writes it out to a local file. I wrote out initial pseudocode for team members to build upon and implemented much of the clustering algorithm including the shift functionality and some data pre-processing.

Jake: I wrote the the bash script to gather wallet data from WalletExplorer.com. I then create a portion of the Scala code in walletClustering.scala that read in all of wallet data. I also helped to create code to read in our wallet information from hdfs. I also helped create the function notClustered. I also helped with bug fixes when possible.

Bibliography

Bishop, Jordan. "Meet The Man Traveling The World On $25 Million Of Bitcoin Profits." *Forbes*, Forbes Magazine, 7 July 2017, https://www.forbes.com/sites/bishopjordan/2017/07/07/bitcoin-millionaire/#3c77c9966261.

"Blockchain Size." *Blockchain.com*, https://www.blockchain.com/en/charts/blocks-size.

"Frequently Asked Questions." *Bitcoin*, https://bitcoin.org/en/faq#general.