Unsupervised Learning of Bitcoin Transaction Data

AMSC 663/664 Midyear Report
Advisor: Dr. Chris Armao
Stefan Poikonen

Table of Contents

Abstract: Bitcoin is largest cryptocurrency, surpassing $8 billion in market capitalization in 2014. Building on the works of Reid and Harrigan, and Brugere, we compile a sequence of metrics (derived from the Blockchain directly and tag data) for every distinct user ID on the Bitcoin network. We augment transaction line data with user-level data metrics of both the source and destination users of the transaction. We then implement Principal Component Analysis (PCA) to reduce dimensionality, followed by K-means clustering. Other clustering techniques and analysis with be included in the next phases of the project.

1 Project Background/Introduction

Bitcoin is the largest decentralized virtual currency with a market capitalization surpassing $8 billion in early 2014 [Aiken, 2014]. A Bitcoin does not exist as a file or physical entity. Rather, a public ledger maintains a log of all past transactions in the BTC network. To spend a Bitcoin, a user applies his/her private key to act as a form of digital signature. The signed transaction is then sent to users on the Bitcoin network for verification. Blocks of transactions are verified by miners by solving cryptographically hard problems. Miners are then awarded Bitcoins for their work. This award is how new Bitcoins come into circulation, thus the minting of Bitcoin is also decentralized, in contrast with traditional currencies with a central bank [Nakamoto,2008].

The Bitcoin currency is still quite young, launched on January 1st, 2009. Consequently, research on Bitcoin (and other crypto-currencies) is in its infancy. Some research concerning global transaction volume, exchange rates, and even deanonymization of large sets of users within the Bitcoin network has been published. [Biryukov, et al., 2013][Moore and Christin, 2013]. However, there exists little research in the way of categorizing Bitcoin transactions, utilizing available Blockchain data and tagged public addresses. Likewise many clustering algorithms have been applied to other financial transaction types, yet there is not significant research into these techniques being applied to cluster Bitcoin transactions.

2 Project Goals

Cluster each transaction within the Bitcoin network utilizing various unsupervised machine learning algorithms. Measure the effectiveness of these clusters in an objective manner. Evaluate the computational and time requirements to form such clusters. Compile a list of potentially anomalous transactions.

3 Approach

First, I downloaded raw Blockchain data spanning from January 2009 (the origin of Bitcoin) to mid-2014. Reid and Harrigan describe detailed methods for deanonymizing and merging some users on the Bitcoin network who utilized multiple public addresses. Brugere furthered there work by describing methods to compile a conveniently formatted transaction table containing approximation 50 million lines. Each line contains source ID, destination ID, timestamp, and transaction amount.

Next I downloaded a table of historic BTC/USD conversion rates (with time granularity of one day). As BTC has seen an enormous rise in value relative to major world currencies since its inception, each transaction amount was converted to into USD to normalize transaction values. This was accomplished by multiplying the BTC value of the transaction by the BTC/USD conversion rate that matches the timestamp of the transaction.

Next we note that source ID and destination ID may serve as index variables by which we may aggregate user-level statistics Examples of statistics that were computed for each user are:

- Total USD sent
- Total USD received
- Highest USD value transaction
- Total number of transactions as sender
- Total number of transactions as recipient
- Timestamp of first transaction
- Timestamp of most recent transaction
- Average timestamp of transactions, weighted by USD
- USD value of BTC still possessed by user

- Ratio of average value in USD of incoming transaction/outgoing transaction

In addition Blockchain.info contains a database of tags for certain public addresses. Therefore, one can associate the owner of this public address with these tags. These tags may be categorized (i.e. gambling, gaming, electronics vendor, YouTube channel, etc.) For each user, we calculate the number of transactions had with different categories of tags.

1Q4G4ZJ1AN1aHkC9YnPQGWYEAxJrW62rJL	Wikileaks	http://wikileaks-donation.weebly.com/
1Dorian4RoXcnBv9hnQ4Y2C1an6NJ4UrjX	Dorian Nakamoto fundraiser	http://www.reddit.com/r/Bitcoin/comments/1ztjmg/andreas_im_fundra...
1DzBEBqzrNsRg8oeRbGWNUr4V2VSjdS7iQ	Wheelchair Fund	http://www.reddit.com/user/IamAlso_u_grahvity/submitted
1436j9Kw2veuQbY1FzPd4VFGZzejLEBjhb	FileZilla Donations	https://filezilla-project.org/

Figure 1: Above is a screenshot of some sample tags, that may be found at http://blockchain.info/tags

We then collated this data by transaction line. That is, the original transaction line data, along with all computed information about the source and destination user of a transaction was combined into one line. A collated transaction line includes all of the following:

1. Transaction ID
2. Source ID
3. Destination ID
4. Timestamp
5. Amount in USD
6. Total USD sent by Source User
7. Total USD sent by Destination User
8. Highest USD value transaction by source user
9. Highest USD transaction by destination user
10. # of transactions by source with gambling tagged sites

11. # of transactions by destination user with gambling tagged sites

The above displays only 11 sample data elements per line. The first 4 were all included in the original transaction table; the other 7 were computed elements.

There are a total of 94 computed data elements per transaction. They were computed by utilizing other blockchain transactions, tag data, or additional pieces of data (such as historic BTC/USD exchange rates). Combined with the original 4 data elements per transactions line, we reach a total of 98 data elements per transaction line.

Define matrix D to be the data matrix of dimension $5.0 * 10^7$ by 98, which contains all $5.0 * 10^7$ transactions and associated data elements.

In expectation of performing a principal component analysis, each column of D was normed. That is each column is transformed to have mean of 0 and variance of 1. Let $D[i]$ for $i = 1, 2, ..., 98$ be the columns of D. Normed data matrix E was computed as follows:

```
for ( i =1:98)
    E[ i ]=D[ i ]-mean(D[ i ])
    E[ i ]=E[ i ] / sd (E[ i ])
end for
```

We then performed a principal component analysis utilizing the Matlab *pca* function, utilizing a subsample of only 100,000 transactions, due to memory constraints. (An iterative method may later be coded, which does not require subsampling.) Variance explained by principal component and cumulative variance explained were computed.

Next various clustering algorithms are applied. The first is K-means, which has been implemented. In the following months C-means clustering with fuzzy logic, CURE Clustering Algorithm (utilizing a sampling method), and the Hierarchical clustering will be implemented. These methods will be applied utilizing a varying number of clusters, and varying number of principal components from the preceding principal component analysis.

Once clusters have been formed, we will consider efficacy of clusters. In

addition we will consider those transactions whose distance away from the nearest centroid is unusually large; these maybe anomalous transactions. In addition, deeper analysis into the makeup of major principal components may be given.

4 Scientific Computing Algorithms

4.1 Principal Component Analysis

Suppose a data set has n observations, and each observation is d dimensional. This data may be represented by a data matrix D of size n by d. Principal component analysis (PCA) transforms the data set into $p \leq d$ dimensions in such a way that the basis vectors for each of the p dimensions are orthogonal to one another and preserve as much of the original variability of the data as possible. PCA is necessarily linked to certain eigenvalues and eigenvectors related to the covariance matrix of the data.

To do so, it is possible to perform a singular value decomposition (SVD). An SVD of the matrix D^* is decomposing it as follows:

$D^* = USV^T$,

where U is n x p, S is p x p, V is p x p, U, V are orthogonal, and S is a diagonal matrix. Of key importance is the following observation:

$D^{*T}D^* = VS^2V^T$

Notice the above representation writes the covariance matrix $D^{*T}D^*$ in diagonalized form. [O'Leary, 2009] A standard eigendecomposition may be appied to $D^{*T}D^*$ to form VS^2V^T. Taking the elementwise squareroot of a diagonal matrix S^2, results in the matrix S. Finally we compute the matrix U as $U = D^*SV^T$.

The matrix V has columns that are right eigenvectors of the covariance matrix. These columns are precisely the principal components. Matrix S

is known as the singular value matrix, which contains the singular values of D^* along its diagonal in decreasing magnitude. The matrix U is often called the score matrix, where each row represents the original data as a linear combination of the principal components.

Alternatively, one may perform iterative methods to compute the principal components of a data matrix. This has the added benefit that one may choose to compute only the first few principal components, if desired. This may reduced computational time and memory usage.

If one wishes to compute only the first p principal components, the method is as follows:

---

for(i=1:p)

- Initiate arbitrary vector x_0
- Repeat until $x_{n+1} \to x$
  - $x_{n+1} = D * x_n$
  - $x_{n+1} = x_{n+1}/\|x_{n+1}\|$
- Store $\lambda_i = \|D * x_{n+1}\|/\|x + n + 1\|$
- Store $y_i = x$
- $D = D - \lambda_i * y_i * y_i^T$

end for

The stored values y_i and λ_i are the ith principal component and ith singular value respectively. Because the method is dependent on power iteration of a matrix, there are potential issues with separability of eigenvalues, which may cause slow convergence. If separability can be ensured and only the first few eigenvalues are required, then this is generally preferred over SVD.

4.2 K-means clustering

Again suppose we have n observations in p dimensional space. K-means clustering is an algorithm that attempts to select k centroids in p dimensional space that will minimize the distance between the observations and the nearest centroids. Finding an absolute minimum distance is NP-hard. In practice the implementation of K-means clustering usually uses a heuristic method known as Lloyd's Algorithm (or sometimes just called the k-means algorithm) which offers substantial time speed-ups.

Lloyd's Algorithm begins by arbitrarily choosing k of the n observations, and designating these as the initial centroids of the clusters. Remaining data points are then assigned to the cluster associated with the nearest centroid. Each cluster centroid is recomputed. Again each data point is then assigned to the nearest centroid. The process of updating centroids and reassignment of observations to the cluster with the nearest centroid continues until one of a number of termination criteria occurs.

Pseudo-code of the algorithm follows:

```
1  //K-means Pseudocode
2  double dataMatrix[n][p]; //Data matrix containing p principle component for each of n transactions
3  int membershipVector[n]; //Integer value is cluster that each of n transactions is assigned to
4  for(i=1 to k)
5  {
6      randomly_initiate_centroid(c_i[p]);
7  }
8  bool centroids_unchanged=false;
9
10 iterations_passed=0;
11 while(centroids_unchanged==false || iterations_passed < max_iterations)
12 {
13     centroids_unchanged=true;
14     for(j=1 to n)
15     {
16         for(i=1 to k)
17         {
18             compute distance(dataMatrix[j] to c_i);
19         }
20         membership[j] = index of nearest cluster;
21         if(membership[j] != membership[j] from previous iteration)
22         {
23             centroids_unchanged=false;
24         }
25     }
26     iterations_passed++;
27 }
```

Each distance computation in p dimension space is $O(p)$. There are three nested loops of constant size n, k, and i. Therefore, the complexity of the algorithm is $O(nkip)$. Due to independence of the computations within inner

loops, there exist great potential for parallelizing Lloyd's Algorithm.

4.3 Other clustering Methods

Other clustering methods will be applied in the later stages of this project. They include the following:

- C-means Clustering with Fuzzy Logic: similar to K-means, but allows for partial membership to clusters.
- Hierarchical Clustering: Agglomerative hierarchical clustering involves starting with many small clusters, merging the most similar. Similarity of clusters may be measured utilizing several different metrics.
- CURE Clustering Algorithm: a hierarchical clustering algorithm which is more adept at handling extreme points.
- Approximate and/or parallelized versions of the above algorithms (time permitting)

More detailed explanations of these algorithms will be given following their implementation.

5 Implementation

Code will be implemented primarily C/C++ and run on a desktop with an Intel i5-3570K CPU and 16GB of DDR3 RAM. Some initial parsing of the block chain may be done in Python as convenient. If time permits, CUDA and/or OpenMP might be used for CPU and/or GPU parallelization respectively for key computationally intensive segments.

6 Validation Methods

First let us distinguish between two forms of validation. The first is validation that the algorithms are implemented correctly. The second is validating

that the use of such algorithms has been effective. At this stage of the project we can only validate the correct implementation of PCA and K-means

Regarding principal component analysis, we used the *pca* function of Matlab. If later iterations of the project code demand a separate implementation of *pca* it is possible to compare resulting principal components, singular values, and score matrix to those that were derived in Matlab. As is, we assume the Matlab *pca* function is valid.

Our implementation of K-means was tested against the Matlab function *kmeans*. Results were identical. (One cautionary note should be given: Since k-means is potentially dependent on initial choice of centroids, our implementation was sure to choose the initial centroids utilizing the same method as the Matlab implementation.) Runtime of our implementation was substantially longer. This is due to additional heuristics applied by Matlab by default.

In the spring, we will implement and validate other algorithms. Our implementation of hierarchical clustering may be compared to that of Matlab. Neither C-means with fuzzy logic or the CURE Algorithm are default Matlab functions. We will test our implementations on other small datasets on which existing C-means and CURE have been documented. Results will then be compared.

After they are implemented, the effectiveness will be gauged. This will consider how rapidly the average distance to centroids decreases an increase in the number of clusters.

7 Test Problems for Verification

We can verify that the various clustering techniques have been properly implemented by running them on sample data sets from the UCI Machine Learning Data Repository and comparing our results to results of previous implementations of the same techniques on the same datasets.

Our primary test problem is clustering tens of millions of transaction lines

in the Bitcoin network. There is no previous implementation on this same exact dataset. However, we may implement pca, kmeans, and hierarhcical clustering using Matlab and compare results to our own implementation.

8 Intermediate Results and Expected Future Results

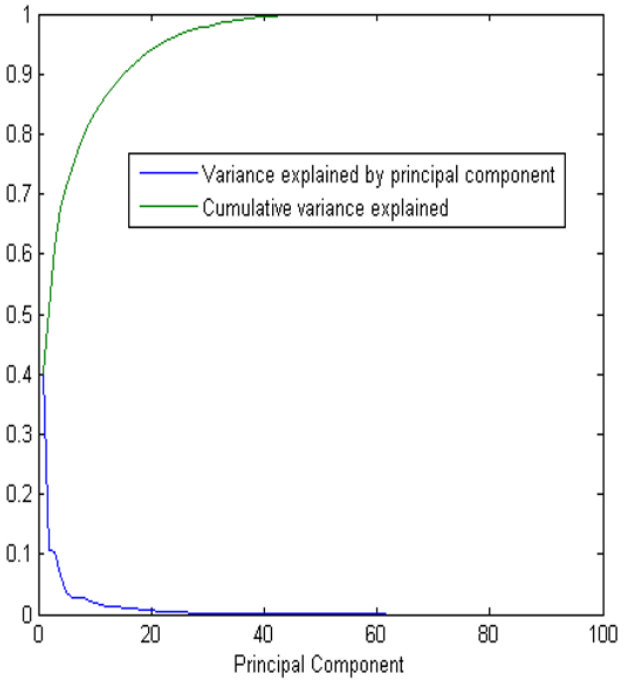
The kind of analysis of the Bitcoin network proposed has no widely circulated precedent. Therefore it is difficult to project end results.

At present data pre-processing, metric computation, PCA and K-means have been implemented. Below is a sample of metrics computed on each of the first twenty users:

IDLineNur	avgDestinationVolumePerMonthInBTC	avgDestinationVolumePerMonthInUSD	avgSourceVolumePerMonthInBTC	avgSourceVolumePerMonthInUSD	btc_value_held_by_user	btc_volume_by_destination_id	btc_t	btc	btc	centralityIn	cc	eisSelfSender	joinDate	number_c	number_c
1	4563.86	71962.9	3695.62	60927.1	1.74E+07	147717	0	##	0	0.940041	1	1	2.01E+13	22232	44447
2	2502.04	334872	4790.5	642122	-709423	1251.02	0	##	0	0.0566038	0	1	2.013E+13	53	188
3	21.7391	180.718	20.868	1741.83	11738.5	472.463	0	##	0	-0.522388	0	-0	1	2.011E+13	134	16
4	42614.4	2.51E+06	42315.5	2.47E+06	938680	218754	###	##	##	0.536416	1	1	1	2.012E+13	15680	1558
5	112.296	4235.25	85.1786	8722.21	62768.2	419.239	0	##	0	0.625	1	1	1	2.012E+13	8	8
6	4.98514	35.2138	4.98466	103.941	6.2013	103.857	0	##	0	0	-0.111111	#	-0	1	2.011E+13	72	48
7	37674.9	844586	37643.5	862915	378820	733406	###	##	##	0.618964	0	1	1	2.011E+13	79032	36400
8	10.2834	168.523	10.2833	717.695	0.312195	93.5789	0	94	0	0	0.1875	0	0	1	2.012E+13	16	13
9	21.5427	642.562	17.3546	1253.52	22244.4	184.549	0	##	0	#	-0.0364964	#	-0	1	2.012E+13	137	18
10	0.137522	2.74662	0.137521	16.9845	0.0186252	2.81003	0	3	0	#	-0.25	0	-0	1	2.011E+13	4	3
11	1.80E+06	1.21E+07	1.80E+06	1.24E+07	-1.64E+07	5.24E+07	###	##	##	#	0.71464	1	1	1	2.01E+13	777380	532534
12	1.58567	13.2182	5.23664	93.6815	-30558.6	21.4065	0	71	0	#	-0.288462	#	-0	1	2.012E+13	52	68
13	2.33403	37.9724	2.33258	363.68	14.5691	37.889	0	38	0	0	-0.582707	0	-1	1	2.011E+13	266	15
14	16.0782	241.746	4.37336	62.5826	76682.2	169.893	0	46	0	#	0.40081	0	0	1	2.012E+13	247	30
15	0.468255	7.18593	0.468106	14.0081	1.98374	10.1143	0	10	0	0	-0.298246	#	-0	1	2.011E+13	57	19
16	199.997	32705	199.997	32999.4	0	99.9983	0	##	0	0	-0.666667	1	-0	0	2.013E+13	3	1
17	339.346	72628.6	339.346	55992.2	0	169.673	0	##	0	0	1	1	1	0	2.013E+13	4	2
18	1589.48	69487.7	1581.58	71316.3	65036.8	21087.2	0	##	0	#	0.587155	0	1	1	2.012E+13	7458	3291
19	258.66	6150.88	258.651	8328.84	55.8545	2353.81	0	##	##	0	-0.27451	#	-0	1	2.012E+13	306	283
20	60017.6	1.36E+06	59840	1.35E+06	2.18E+06	1.19E+06	###	##	##	#	0.485587	0	0	1	2.011E+13	46626	33275
21	1614.75	43749.5	1586.55	50735.6	154443	14363.6	0	##	0	#	0.596754	0	0	1	2.013E+13	2295	1368

A principal component analysis was conducted then on full set of nearly 50 million transactions and associated metrics.

Principal Component	Variance Explained
1	0.4019
2	0.1058
3	0.1045
4	0.0644
5	0.0364
6	0.0281
7	0.028
8	0.0265
9	0.021
10	0.0176
11	0.0152
12	0.0141
13	0.0127
14	0.0124
15	0.0102
16	0.0099
17	0.0089
18	0.008
19	0.0077
20	0.0069
21	0.0056
22	0.0055
23	0.005
24	0.0049
25	0.0042



The left half of the above figure contains the variance explained of the first 25 principal components. The right half of the figure plots this variance explained by principal component (blue) and cumulative variance explained by principal components (green).

Upon examining the first principal component vector, we found that elements 2,4,7,9, and 29 had the greatest magnitude. These correspond to the following columns:

avgDestinationVolumePerMonthInBTC
avgSourceVolumePerMonthInBTC
btc volume by destination id
btc volume by source id
usd volume by destination id

Preliminarily this suggests that volume of BTC transactions (denominated in USD or BTC) is a powerful indicator of other elements of the data set. Further inspection of the next few principal components is still required.

K-means has been completed for $k = 5, 10, 20$ on the score matrix returned from performing PCA on the sumsample. Further analysis of clusters will be given in the second semester.

9 Concluding Remarks

The project thus far is on schedule. As is often the case with such large data sets that pull in data from multiple sources, pre-processing was tedious and computationally expensive. Principal component analysis required subsampling due to memory concerns, but implementation via iterative methods at a later date may solve this issue. K-means has been implemented on the subsampled score matrix returned from PCA. However, during the next semester, when applying K-means on the full data set and utilizing a larger number of clusters (a larger k), it may be beneficial to parallelize and/or apply additional heuristics such as the K-means++ algorithm. Much of the ground work (pre-processing, norming, PCA) has been completed. Once multiple clustering algorithms have been implemented, it will be interesting to compare the clustering results in terms of similarity, time complexity, and distance reduction.

10 Timeline

Phase 1 October 1 to November 15: Data Download, Transformation, etc.

- Presentation/Proposal [Complete]
- Blockchain download [Complete]
- Transformation of Blockchain data to usable transaction line table [Complete]
- Computation of user-level metrics [Complete]

- Computation of tag-related metrics [Complete]

Phase 2 November 15 to November 28: Principal Component Analysis

- Normalize data. Determine whether to apply log transformation on certain columns of data. [Complete]
- Implement Principal Component Analysis via Singular Value Decomposition [Complete]

Phase 3a November 28 to December 15: Implementation of Clustering Algorithms

- Implement K-means clustering [Complete]

Phase 3b January 30 to March 30: Implementation of Clustering Algorithms

- Implement C-means clustering with Fuzzy Logic
- Implement CURE Clustering Algorithm (time permitting)
- Implement other clustering (time permitting)

Phase 4: April 1 to April 25: Analysis of Results

- Running the code of varying k
- The computation of cluster evaluation criterion from (6: Data Validation) above
- Analysis of clustering results
- Identification of potential anomalies

[At this point, if the project is ahead of schedule, I will attempt to parallelize key segments of code.]
Phase 5: April 25 to May 15: Final Paper and Presentation

- Finish final paper
- Make final presentation
- Consider routes of further research

11 Milestones

Milestones coincide with the completion of phases 1, 2, 3, 4 and 5 above listed in the above section.

12 Deliverables

- C++/Python code for transforming data to transaction line table [Complete]
- C++ code for computing user-level metrics [Complete]
- C++ code for computing tag-related metrics [Complete]
- C++ code for normalizing data prior to PCA [Completed]
- C++ code for computing K-means clustering [Complete]
- C++ code for computing Fuzzy C-means clusters [Spring]
- C++ code for other clustering (time permitting) [Spring]
- Evaluation metrics from clustering with different numbers of clusters across different clustering algorithms [Spring]
- First-Semester Progress Report [Complete]
- Final Reports [Spring]
- Weekly Reports [Ongoing]

13 Bibliography

Aiken, Michael. "Future Funds: The Latest on Bitcoin and Cryptocurrency." Diplomatic Courier. Diplomatic Courier, 4 Sept. 2014. Web. 02 Oct. 2014.
Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." (2008): 28.

Biryukov, Alex, Ivan Pustogarov, and R. Weinmann. "Trawling for tor hidden services: Detection, measurement, deanonymization."
Security and Privacy (SP), 2013 IEEE Symposium on. IEEE, 2013.

Ding, Chris, and Xiaofeng He. "K-means clustering via principal component analysis."
Proceedings of the twenty-first international conference on Machine learning. ACM,2004.

Guha, Sudipto, Rajeev Rastogi, and Kyuseok Shim. "CURE: An Efficient Clustering Algorithm for Large Databases." (1998): Web.
<http://www.cs.sfu.ca/CourseCentral/459/han/papers/guha98.pdf>.

Guha, Sudipto, Rajeev Rastogi, and Kyuseok Shim. "CURE: an efficient clustering algorithm for large databases."
ACM SIGMOD Record. Vol. 27. No. 2. ACM, 1998.

Moore, Tyler, and Nicolas Christin. "Beware the middleman: Empirical analysis of Bitcoin-exchange risk."
Financial Cryptography and Data Security. Springer Berlin Heidelberg, 2013. 25-33.

O'Leary, Dianne P. "Chapter 5: Matrix Factorizations." Scientific Computing with Case Studies.
Philadelphia: Society for Industrial and Applied Mathematics, 2009. 73,74. Print.

Raskutti, Bhavani, and Christopher Leckie. "An Evaluation of Criteria for Measuring the Quality of Clusters." Telstra Research Laboratories (1999): Web.
<http://ww2.cs.mu.oz.au/ caleckie/ijcai99.pdf>.