Unsupervised Clustering of Bitcoin Transaction Data

Midyear Report

1

AMSC 663/664 Project

Advisor: Dr. Chris Armao

By: Stefan Poikonen

Bitcoin: A Brief Refresher

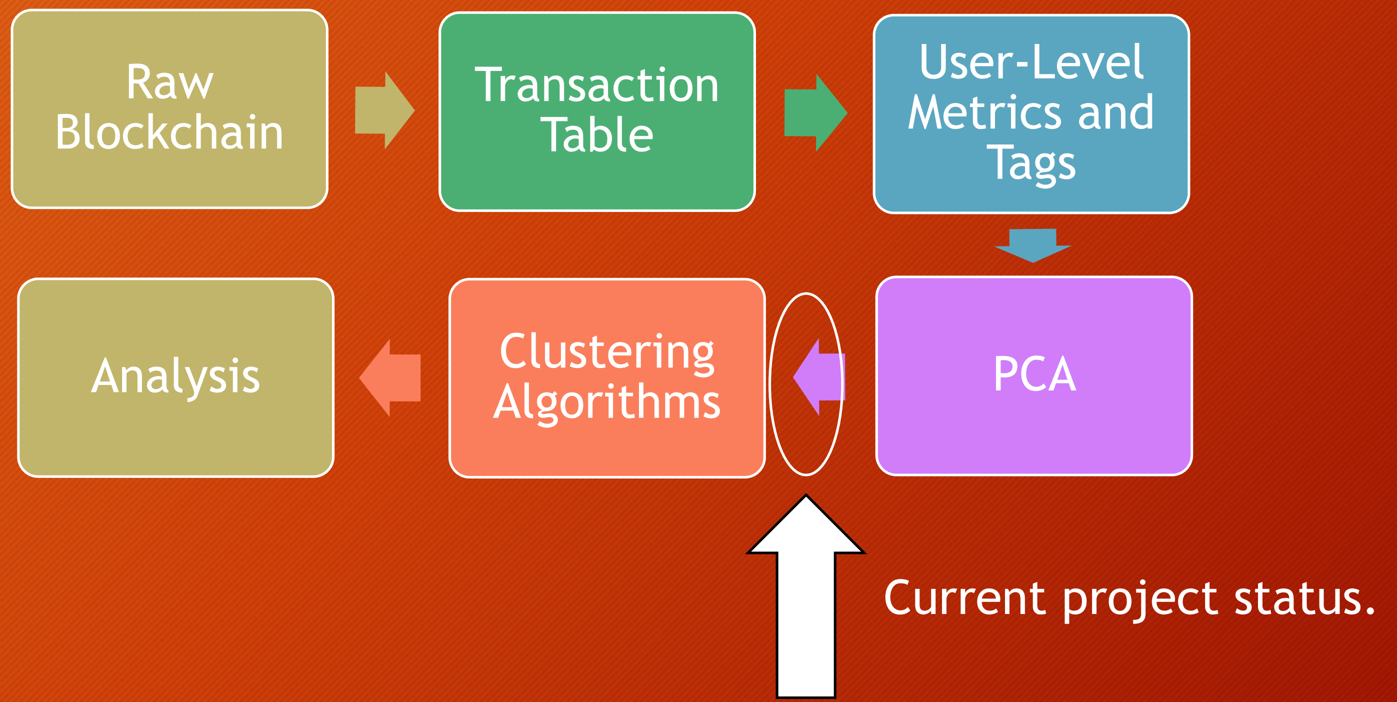
- Bitcoin is a decentralized cryptocurrency used for digital transactions
- The Bitcoin Network was first implemented January 1st, 2009
- In early 2014 market capitalization of Bitcoin surpassed $8 billion
- Utilizes Private/Public Key structure for signature and verification
- History of every transaction is stored in the publically available "Block Chain"

Lee, Timothy B. "What Is Bitcoin?" Vox. Vox, 14 Oct. 2014. Web. 01 Dec. 2014.

Project Goal

- Categorize Bitcoin transactions utilizing Blockchain data
- Without a training set, this is accomplished via unsupervised learning of transaction data
- Form clusters (K-means, C-means (fuzzy logic), Hierarchical, “CURE” Algorithm)
- Evaluate the efficacy of the clusters
- Evaluate the computational time of each clustering method
- List potential anomalous transactions

High Level Flow of Project



The Data to Transaction Table

- The public ledger, or “block chain” is available to download
- Reid and Harrigan, and Brugere describe in detail transformations from the raw block chain to transaction line tables
- We reproduced their methods to convert the Blockchain to an easily usable transaction table
- Around 50 million transaction lines
- Each transaction line contains the following data elements:
  - Source ID
  - Destination ID
  - Timestamp
  - Amount Of Bitcoin Transferred

Reid, Fergal, and Martin Harrigan. An analysis of anonymity in the bitcoin system. Springer New York, 2013.

Tagged Addresses

- Blockchain.info maintains a database of "tagged" public addresses
- Tags associate a public address with an entity, cause, website, etc.
- These tags have been categorized: gamer, charity, hacking, etc.
- We can compute the number of times a given user has been adjacent to certain categories, or other measures of a users closeness to a particular category of tag

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/

Screenshot from: https://blockchain.info/tags

Other User Level Metrics

- Compute metrics on every user by looping through every transaction.
- User metrics include average transaction amount, join date, maximum transaction, local centrality, page rank (through power iteration), etc.
- Naively looping over every user and transaction is infeasibly large at $O(n^2)$, where n is the number of transactions.
- Sorted transactions by userID $O(n \log(n))$, then computing metrics is only $O(n)$ complexity
- With the computed user-level data, the data set grows from 4 columns to 98 columns.

Sample of Computed User-Level Metrics

IDLineNur	avgDestinationVolumePerMonthInBTC	avgDestinationVolumePerMonthInUSD	avgSourceVolumePerMonthInBTC	avgSourceVolumePerMonthInUSD	btc_value_held_by_user	btc_volume_by_destination_id	btc_	btc	btc	bt	centralityIn	c	cer	isSelfSender	joinDate	number_c	number_c
1	4563.86	71962.9	3695.62	60927.1	1.74E+07	147717	0	##	0	#	0.940041	1	1	1	2.01E+13	22232	44447
2	2502.04	334872	4790.5	642122	-709423	1251.02	0	##	0	#	0.0566038	0	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	42319.5	2.47E+06	938680	218754	###	##	##	#	0.536416	1	1	1	2.012E+13	15680	1958
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	154442	14263.6	0	##	0	#	0.506754	0	0	1	2.012E+13	2295	1368

Clustering Prerequisites: Norming of Data

- There is no natural way to compare “distance” in each column of user data.
- Each column of data has different scale.
- Measuring “distance” between augmented transaction lines in clustering is dependent on this scaling
- It would be possible to learn a metric with “good” scaling, if we had a training set.
- Without a training set we:
  - Normalize data, such that for each column: $\mu \rightarrow 0$ and $\sigma^2 \rightarrow 1$
  - Log transformations to enhance normality of some data columns

Clustering Prerequisites: Principal Component Analysis

- Once data is normed, we use a principal component analysis.
- PCA preserves as much variability as possible with a reduced number of orthogonal components.
- Principal components may be computed through power iteration or singular value decomposition.

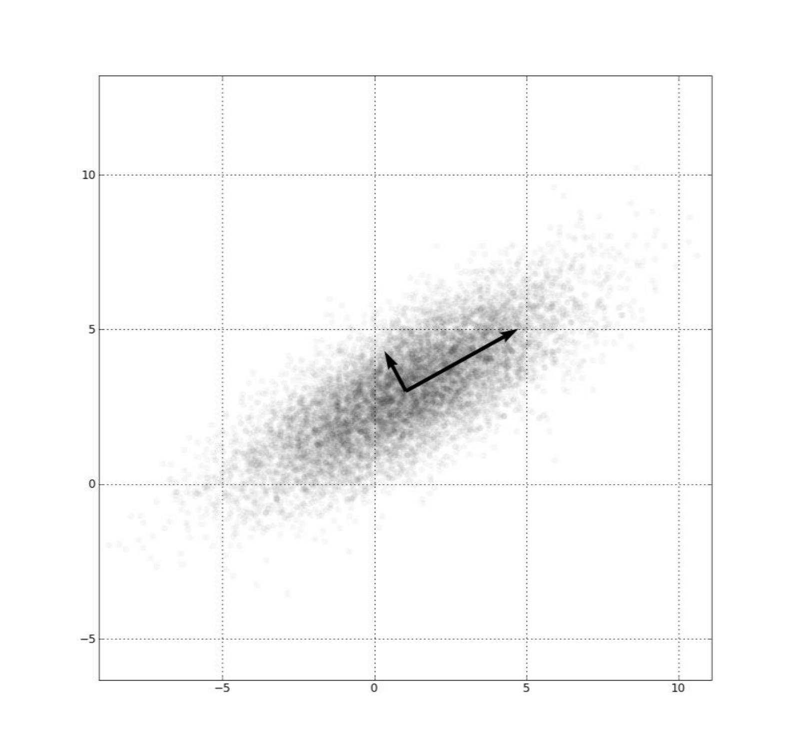


Illustration of PCA on 2-D sample points: arrows point in the direction of the two principal components. (Image credit: Wikimedia Commons)

Principal Component Analysis: Algorithms

Power Method to compute *p* principal components of matrix *A*

- For i = 1 to p:
  - Start with arbitrary x_0 vector.
  - Repeat until $x_{n+1} \rightarrow x_i$:
    - $x_{n+1}=Ax_n$
    - $x_{n+1}= x_{n+1}/||x_{n+1}||$
  - $\lim_{n \rightarrow \infty} ||Ax_{n+1}||/||x_{n+1}|| = \lambda_i$
  - $A = A - \lambda_i x_i x_i T$
  - Store x_i and λ_i as principal component.

Singular Value Decomposition

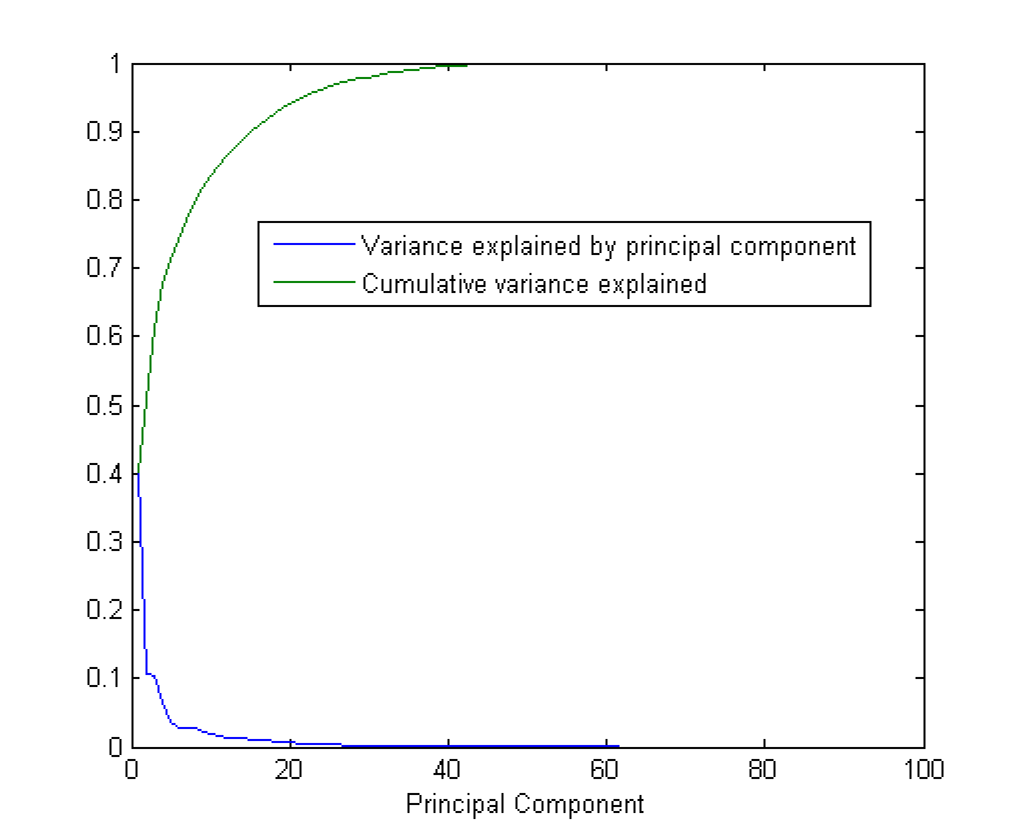
- SVD decomposes an m x n matrix *A* as: $A = USV^T$
- Where *U* is *m x n*, *S* is *n x n*, *V* is *n x n*, and U and V are orthogonal.
- Key insight: $A^TA = VS^2V^T$, is diagonalization of A^TA.
- The columns of V are the eigenvectors of A^TA.
- S is a diagonal matrix containing the eigenvalues of A in descending order.
- Computational cost $O(mn^2)$

O'Leary, Dianne P. "Chapter 5: Matrix Factorizations." Scientific Computing with Case Studies. Philadelphia: Society for Industrial and Applied Mathematics, 2009. 68-75. Print.

Principal Component Analysis: Results

- Sampled 100,000 data points (smaller *n*) to reduce memory usage.
- Utilized Matlab's built in *pca* function.
- Run time on the order of a few minutes

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



K-means Clustering: A Description

- Suppose we choose *p* principle components, and now have *n* data lines, each in p-dimensional space.
- This algorithm initiates k centroids.
- Next it loops through all *n* data vectors and computes the distances between each data vector and each centroid.
- Each data vector becomes of a member of the nearest centroid. (Opportunities for heuristic optimization?)
- The algorithm is *O(nkip)* where n is the number of transactions, k the number of clusters, and i the number of iterations.
- Though n is large (~50 million), k, i, and p may be chosen as small.
- Highly parallelizable.

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

K-means Clustering: Pseudocode


```
//K-means Pseudocode
double dataMatrix[n][p]; //Data matrix containing p principle component for each of n transactions
int membershipVector[n]; //Integer value is cluster that each of n transactions is assigned to
for(i=1 to k)
{
    randomly_initiate_centroid(c_i[p]);
}
bool centroids_unchanged=false;

iterations_passed=0;
while(centroids_unchanged==false || iterations_passed < max_iterations)
{
    centroids_unchanged=true;
    for(j=1 to n)
    {
        for(i=1 to k)
        {
            compute distance(dataMatrix[j] to c_i);
        }
        membership[j] = index of nearest cluster;
        if(membership[j] != membership[j] from previous iteration)
        {
            centroids_unchanged=false;
        }
    }
    iterations_passed++;
}
```

Clustering Validation

- There exist dozens of implementations of K-means and accompanying small data sets listed online.
- In the remainder of the semester I will test my implementation utilizing these datasets. If I arrive at the exact same clusters after numerous examples, this will validate my code.
- Detailed analysis of K-means and other algorithms is scheduled for the second half of the project.
- In general, as the number of clusters increase we expect:
  - Decrease in distance to cluster centroids
  - Greater compactness within clusters
  - We may compare performance of various clustering with "area under curve"
  - As the limit we should expect as $k \to n$, average distance to nearest centroid $\to 0$.

Time Line

- Now-November 15: Data transformation, parsing, user-metric computation, tag-metrics, etc. [Done]
- November 15-December 15: PCA [Done] and K-means clustering [In progress]
- February 1-March 31: Fuzzy C-means clustering, CURE clustering, other clustering algorithms (time permitting)
- April 1 – April 25: Analysis of cluster quality, parallelization (time permitting)
- April 25 – May 15: Paper and presentation

Deliverables

- C++/Python code for transforming data to transaction line table [Done]
- C++ code for computing user-level metrics [Done]
- C++ code for computing tag-related metrics [Done]
- C++ code for normalizing data prior to PCA [Done]
- C++ code for computing K-means clusters [In Progress]
- 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 [Done]
- Final Reports [Spring]
- Weekly Reports [In Progress]

Summary

- Parsing Blockchain and creating usable transaction tables were major overhead components of the first half of this project and are now done.
- Data norming and PCA are also complete.
- K-means is scheduled to finish before end of semester.
- Next semester more clustering algorithms will be implemented, leading to mathematically interesting results, analysis and performance comparisons.

The End

- Questions?