Bitcoin: Identifying Theft?

Chaney C. Lin
chaneyl@princeton.edu

Abstract

This report reviews our analysis of a year's worth of Bitcoin transactions, using models based on singular value decomposition, support vector machines, and k-means clustering. The results appear to indicate some structure reflecting Bitcoin theft during the year of transactions under study.

1 Introduction

Bitcoin is a virtual currency that is gradually gaining mainstream acceptance. The USD to Bitcoin exchange rate was \$6.18 on January 2, 2012, rising to a peak of \$979.45 on November 25, 2013, and declining to \$235.62, as of May 2, 2015 [1]. As with all alternative currencies, one of the primary headwinds for gaining traction is lack of credibility and confidence as a form of payment. In this project report, we review our attempts at elucidating structure within a year's worth of Bitcoin transactions.

2 Data Description

All Bitcoin transactions are tracked in a public ledger called the "blockchain", where senders and receivers are referred to by their "addresses". Addresses are usually single-use, but people often trade through services called "wallets", whereby one address is used to label all of their transactions.

The training data set consists of roughly a year's worth of transactions, from March 2012 to March 2013, totaling 3,348,026 transactions among $N = 444{,}075$ unique addresses. The test data set consists of 10,000 pairs of addresses, which did not interact in the training data. 1,000 of them either later interacted (after March 2013, before May 2014) or were declared as interacting to include more positive test points.

We fit models to the training data, to predict whether the pairs in the test data traded or not. We evaluate models based on how well they predict the 1,000 true interactions in the test data.

3 Methods

3.1 Data processing

The data was downloaded from `http://www.princeton.edu/whao/COS424/Bitcoin` on April 13, 2015. It was initially in triplet form (i, j, c_{ij}), where i denotes the sender, j the receiver, and c_{ij} the number of times the pair (i, j) interacted, for only those where c_{ij} is nonzero.

We looked at two forms of the data: (1) the raw counts; and (2) a binary thresholded count, where we set all nonzero c_{ij} to one. We call the latter the binary representation of the data.

The data was converted from triplet form into a sparse matrix format compatible with the models we applied; the spare matrix format is from the SciPy library [2]. We shall refer to the transaction matrix both as $\mathbf{C}$ and as C_{ij}, with the latter also referring to the element in the ith row and jth column (transactions between sender i and receiver j). In dense matrix format, the raw data would occupy $N^2 \times 4$ bytes of memory (about 800 GB), and the binary representation

would occupy N^2 bits of memory (about 25 GB). These large memory requirements constrained the available methods of analysis.

3.2 Models

The models were applied using built-in functions of the scikit-learn Python library [3]. Default parametrizations were used, unless otherwise specified. Below, we define each model and explain our motivation for them.

Singular value decomposition (SVD). This model first factorizes $\mathbf{C}$ into its singular value decomposition $\mathbf{C} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$, then keeps only the largest k singular values in $\mathbf{\Sigma}$, which we denote by $\mathbf{\Sigma}'$, zeroing the other singular values. The transformed matrix is then $\mathbf{C}' = \mathbf{U}\mathbf{\Sigma}'\mathbf{V}^*$. Intuitively, it filters out the signal from the noise. One major advantage of this method is that it is extremely fast, requiring just matrix multiplication. We identify the "probability" of a transaction between sender i and receiver j as a properly rescaled $C'_{ij} = \sum_{m=1}^{N}(U\Sigma')_{im}V^*_{mj}$. Before rescaling, the values do not necessarily lie within $[0, 1]$, so they do not represent true probabilities. For predictions p_i, we rescale by $p'_i = (p_i - p_{\text{min}})/(p_{\text{max}} - p_{\text{min}}) : [p_{\text{min}}, p_{\text{max}}] \to [0, 1]$, where $p_{\text{min}}, p_{\text{max}}$ are the minimum and maximum predicted values. This is an order preserving map.

k-means clustering (k-means). We run multiple models based on k-means clustering. The clustering step partitions the rows of $\mathbf{C}$ into k clusters. That is, each address i is given a label $l_i \in \{1, 2, \ldots, k\}$. We denote the centroid of cluster j by $\mathbf{x}_j$. We have used $k = 10, 100, 1000$. Below, we describe the four different models. Note that Cluster Methods 2, 3, and 4 each have only one nontrivial point of the ROC curve, because their assignments are binary (0 or 1).

Cluster Method 1 assigns the probability of interaction between pair (i, j) as the jth component of the centroid $\mathbf{x}_{l_i}$, i.e. $p(i, j) = (l_i)_j$. This probability represents how "close" the center of the ith cluster is to the jth individual.

Cluster Method 2 first obtains the clusters of the trade partners of i, the set $L_i \equiv \{l_j : C_{ij} \neq 0\}$. We say (i, j) has traded if receiver j belongs to a cluster in L_i, i.e. $p(i, j) = 1$ if $l_j \in L_i$, 0 otherwise. This models the situation where individuals tend to interact with the same clusters.

Cluster Method 3 first obtains the partner clusters of all individuals within a given cluster, the set $T_K \equiv \{L_i : l_i = K\} = \{l_j : l_i = K, C_{ij} \neq 0\}$. We say (i, j) has traded if receiver j belongs to a cluster in T_{l_i}, i.e. $p(i, j) = 1$ if $l_j \in T_{l_i}$, 0 otherwise. This models the situation where clusters tend to interact with the same clusters. For small numbers of clusters, it is expected to have a high degree of false positives.

Cluster Method 4 first obtains the trade partners of all individuals in cluster K, the set $\Psi_K = \{j : l_i = K, C_{ij} \neq 0\}$. We say (i, j) has traded if receiver j is in Ψ_{l_i}, i.e. $p(i, j) = 1$ if $j \in \Psi_{l_i}$ 0 otherwise. This models the situation where clusters tend to interact with the same individuals.

Support vector machine (SVM). This model is built on support vector machines, and similar to SVD, it is a robust model that can filter noise. For a given test data pair (i, j), we first build a support vector from the training data that separates rows that have traded with the receiver j, and n randomly chosen rows (not equal to i) that have not traded with j. Then we classify i according to which side of the support vector lies the ith row of $\mathbf{C}$. We use $n = 20$. Moreover, there is only one nontrivial point on the ROC curve, so we report only the accuracy.

3.3 Evaluation

The models are evaluated by their computational speed and generalization error. Computational speed is evaluated by two metrics: the fitting time t_F, which is the wall time (in seconds) spent fitting the model to the training data, time spent clustering for the k-means models; and the predicting time t_P, the wall time predicting the labels of the test data.

Generalization error is evaluated using the following two metrics: area under the ROC curve (AUC), and the accuracy (ACC). In terms of the number of true negatives (TN), true positives (TP), false negatives (FN), and false positives (FP), the accuracy is defined as $\text{ACC} = \frac{TN+TP}{TN+TP+FN+FP}$. For this data set, TP refers to a correctly identified interaction; TN a correctly identified non-interaction; FP a non-interaction predicted as an interaction; and FN an interaction predicted as a non-interaction. For models with at most one nontrivial point on the ROC curve, we provide only their accuracy. For models which have more points, we provide the maximum accuracy among points along the curve.

Table 1: The top left contains results for SVD models; the bottom left contains results for the SVM model; the right contains results for k-means based models. For SVD, we report the number of singular values N_s. For SVM, we report the number of randomly chosen zero values n. For k-means based models, we report the method (1-4), the number of clusters k, and the predicting time t_P in seconds. For both, we report the type of data (r for raw, b for binary), the fitting time t_F in seconds, the area under the ROC curve (AUC) and the maximum accuracy (ACC).

N_s	type	t_F (sec)	AUC	ACC (%)
1	r	2.50	0.696	91.6
2	r	2.49	0.412	91.3
3	r	3.16	0.446	91.3
4	r	4.00	0.515	91.4
6	r	3.96	0.500	91.4
10	r	3.41	0.445	91.5
20	r	35.65	0.328	91.4
50	r	38.97	0.285	90.5
1	b	2.51	0.716	91.5
2	b	3.06	0.720	91.7
3	b	3.21	0.666	91.4
4	b	4.94	0.656	91.4
6	b	4.28	0.655	91.5
10	b	3.96	0.627	91.5
20	b	29.91	0.532	91.2
50	b	40.99	0.443	90.3

n	type	t_F (sec)	AUC	ACC (%)
20	b	2520	-	77.3

Method	k	type	t_F (sec)	t_P (sec)	AUC	ACC (%)
1	10	r	11.69	0.014	0.556	90.1
1	100	r	150.41	0.018	0.695	92.0
1	1000	r	1399.13	0.032	0.507	90.0
1	10	b	10.09	0.014	0.529	90.1
1	100	b	133.26	0.014	0.572	90.7
1	1000	b	2276.4	0.149	0.508	90.1
2	10	r	11.69	1.316	-	20.7
2	100	r	150.41	1.123	-	74.4
2	1000	r	1399.13	1.116	-	58.7
2	10	b	10.09	1.084	-	25.1
2	100	b	133.26	1.222	-	63.5
2	1000	b	2276.4	1.36	-	58.5
3	10	r	11.69	108.313	-	10.0
3	100	r	150.41	129.13	-	11.9
3	1000	r	1399.13	99.833	-	16.3
3	10	b	10.09	106.416	-	10.0
3	100	b	133.26	98.451	-	10.5
3	1000	b	2276.4	101.916	-	13.8
4	10	r	11.69	119.494	-	23.9
4	100	r	150.41	126.374	-	84.2
4	1000	r	1399.13	100.203	-	65.6
4	10	b	10.09	115.873	-	26.1
4	100	b	133.26	100.456	-	73.7
4	1000	b	2276.4	103.575	-	62.8

4 Results

4.1 Computational speed

The SVD models were very fast, and their fitting times t_F increased as expected with the number of retained singular values, roughly linearly. There was effectively no time required to predict, since this step was just an inner product for each test point. SVM was quite slow, because an optimization step was performed for every test data point. We expect this time to increase as n increases. For the clustering step of the k-means models, t_F also increased as k increased, roughly linearly. Among the different cluster models, however, there was some variation in predicting time t_P. Methods 1 and 2 took effectively no time to predict, as predictions were simply array searches. Methods 3 and 4 required construction of auxiliary arrays, combining labels of trade partners in 3, and their addresses in 4; these were similar constructions, so as expected, they were comparable in t_P.

4.2 Evaluation and Discussion

The SVD models with an appropriate number of singular values performed best, with top performer the binary SVD with 2 singular values. Given its speed and simplicity, it is by far the best performing model, though it suffers for lack of transparency in its interpretation. We make an attempt here to extract some insight into the results.

Raw vs binary. To motivate insight, we compare the assumptions underlying the raw and binary data sets. The training data contains some pairs of addresses which engaged in thousands of transactions, and some which engaged in only one. Binary thresholding eliminates this information, treating the fact that pairs have traded as the feature,

and not the frequency. This is more consistent with the test data, where interacting pairs trade at most once, and with the variable we are predicting, the probability of trading, not the number of trades. However, this may be useful information, which is reflected in results of SVD and clustering, which we shall explain in the following subsections.

Bitcoin thieves? Delving into the qualitative history of Bitcoin during the time period represented by the data set uncovers that there were multiple large scale thefts of Bitcoin [4]. This may be reflected by the high frequency trades (thieves j stealing from individuals i would have a large number of transactions, as they clean out the wallets of i). An alternative interpretation is that there exist some trading brokers, or popular stores which support Bitcoin.

SVD. The relative magnitude of singular values exhibits a slower dropoff in the binary data than in the raw data. The sequence of ratios of singular values $\gamma_i \equiv s_{i+1}/s_i$ for the binary data starts out as $\{1.16, 1.19, 1.14, 1.02, 1.11, 1.01, \dots\}$ and drops consistently to around 1 around γ_{15}. This indicates that in the binary data, there continues to be signal in several of the subleading singular values. In contrast, for the raw data, $\gamma_i = \{3.53, 1.32, 2.22, 1.58, 1.00, \dots\}$, and remains close to 1 thereafter, indicating that the largest singular value captures a disproportionally high amount of the signal in the raw data, though there continues to be some signal present in the first few subleading terms, with $\gamma_3 > \gamma_2$.

This structure is reflected in their predictions. For the binary data, there does not exhibit a sharp decrease in the AUC, but rather, a gradual, almost monotonic decrease, whereas for the raw data, retaining just one singular value has a substantially larger AUC of 0.696, compared to with two, which has an AUC of 0.412. The fact that $\gamma_3 > \gamma_2$ is also reflected in the AUC, whereby the AUC increases from raw SVD with 3 singular values to with 4.

That both raw SVD and binary SVD performed comparably for few singular values, suggests that they share some dominant structure, possibly some clustering structure. In some sense, SVD smooths out the initially nonzero transactions onto the initially zero transactions; by retaining the frequency of trades in the raw data, there is higher concentration around the higher frequency pairs. The max accuracy of all SVD models were comparable, around 91%.

k-means. All the results for the clustering methods have the caveat that they may be sensitive to initial conditions, depending on the number of local minima in the feature space. It is almost certainly the case that our high-dimensional feature space contains several local minima. Nonetheless, there appears to be some structure in the clustering results that might inform future investigation.

We note that k-means on the raw data, for the same k, almost always outperforms k-means on the binary data. This further supports the hypothesis that the frequency of trades is predictive. The max accuracy of Cluster Method 1 was similar to that of SVD, around 91%, which supports the interpretation of "smoothing around the existing interactions" for SVD. However, it did not perform as highly with respect to AUC. The other cluster methods, which were only evaluated based on accuracy, did not perform as well. Cluster Method 3 has an extremely low accuracy, around the minimum of 10% (all false positives); this is attributable to its high false positivity rate, since most clusters trade with members of most other clusters, though as expected, it performs better with larger k. The accuracy of the other cluster methods dramatically improve as k increases from 10, indicating that the optimal number of clusters is greater than 10. However, given the sensitivity to local minima, we would have to run the models several times for any given k, to determine what range of k is optimal.

SVM. The SVM model, given its random component, should be run multiple times to gain insight into its generalization error. Given its slow speed, however, we only performed one trial, the result of which had an accuracy competitive with cluster methods 2-4, but far lower than SVD and cluster method 1.

5 Conclusion

Our feature space is sparse and large. The size constrains the space of models that can be implemented in practice. Dimensional reduction of the feature space is natural in this context, but only with methods that are suited for large matrices. One avenue to explore further is online methods, which can circumvent storage of the full matrix.

One extension to study the hypothesis that the signal in clustering and SVD is due to high frequency receivers, possibly representing Bitcoin thieves, would be to divide the data set into one that trades with likely culprits, perhaps measured by the in-degree, and one that does not, then perform the methods on the subsets separately.

Other extensions include repeating the clustering method multiple times to address the issue of local minima, and to find the optimal range of k; similarly for the SVM model, to evaluate generalization error, and to find the optimal range of n.

References

[1] Coindesk (2015). Bitcoin price index chart. URL http://www.coindesk.com/price/. [Online; accessed 2015-05-02].

[2] Jones E, Oliphant T, Peterson P, et al. (2001–). SciPy: Open source scientific tools for Python. URL http://www.scipy.org/. [Online; accessed 2015-05-02].

[3] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, et al. (2011) Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12: 2825–2830.

[4] Bitcoin Forum (2014). List of bitcoin heists. URL https://bitcointalk.org/index.php?topic=576337. [Online; accessed 2015-05-02].