Predicting transactions in Bitcoin blockchain data

Gregory Gundersen
Princeton University
ggundersen@princeton.edu

Prakhar Kumar
Princeton University
prakhark@princeton.edu

Abstract

We predict future Bitcoin transactions for pairs of addresses based on historical transaction data. The training dataset consists of transaction counts between sender-receiver address pairs that interacted within a one-year period. The test data consists of address pairs which did not interact within that one year period but may have in the future. We convert the training dataset into an extremely sparse adjacency matrix of all possible combinations of addresses with each entry corresponding to the number of transactions and then use matrix completion methods to reconstruct a low-rank approximation of the data which can be used for prediction. We find that two matrix completion methods, singular value decomposition and nonnegative matrix factorization, generalize significantly better than random guessing.

1 Introduction

1.1 Background

Bitcoin is an online virtual currency introduced in 2008 by Satoshi Nakamoto [5]. All transactions are cryptographically signed and recorded on a public ledger called the blockchain. The blockchain allows the global, peer-to-peer network of Bitcoin users to validate all transactions and thereby eschew a centralized bank. See [6] for a detailed explanation of how Bitcoin works. The upshot is that with Bitcoin, everyone is the bank and the blockchain is an account of the entire history of all anonymized transactions.

In this paper, we predict transactions between pairs of addresses in a portion of the Bitcoin blockchain. The training dataset consists of transaction counts between sender-receiver address pairs that did interact, i.e. there are no zero counts, within a one year period. Therefore the entire training dataset consists of positive examples. The testing dataset consists of some pairs of addresses which did not interact and some which did in a subsequent year. We find that matrix completion methods to estimate the probability of transactions between a sender-receiver pair work better than traditional classification models. In particular, we investigate two matrix completion methods, singular value decomposition (SVD) and nonnegative matrix factorization (NMF). Their performances were evaluated on accuracy, precision, recall, and area under the curve (AUC). We found that NMF generalizes better on the test data in comparison to SVD.

In addition, we explored an SVM-based classifier on a modified dataset in which the additional features are the indegree and outdegree of both the receiver and sender. However, we found that the test accuracy was always around 90% with low accuracy and recall despite constructing a balanced training set consisting of equal number of positive and negative labels.

1.2 Related work

Matrix completion methods are frequently used for recommender systems [3]. This is because recommender systems typically rely on incomplete and/or sparse data to provide suitable recom-
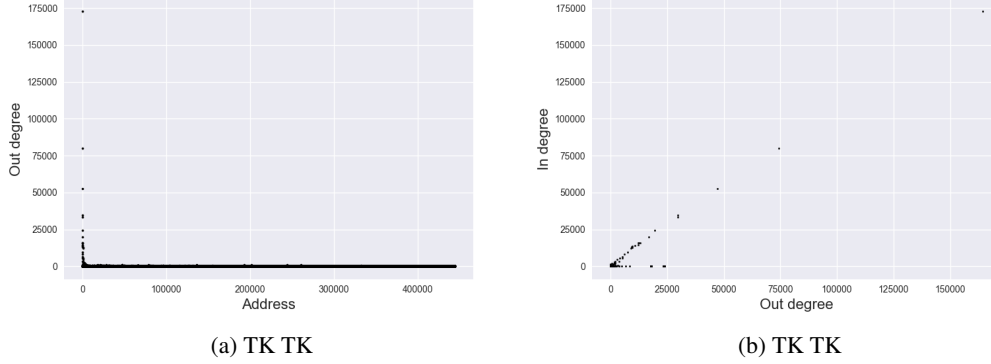
054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100
101
102
103
104
105
106
107

(a) TK TK

(b) TK TK

Figure 1: TK TK

mendations to the users for services like Amazon, Facebook, and Netflix. The sparsity is common because in most scenarios a user cannot rate or even be aware of most items. The central challenge, then, is to develop methods to fill in missing recommendations based on the user and others' recommendations. The key assumption is that user preferences are generally guided by a small number of underlying factors and so decomposing the data into a low-rank representation creates a more general model.

This approach can be used for this work predicting Bitcoin transactions because we assume the number of factors which determine whether or not two users will interact is much smaller than the number of addresses in the dataset. The significant sparsity of our dataset makes it ideal to implement matrix factorization methods for identifying low-rank decompositions of the original data.

The Netflix Prize is a popular example of recommender systems employing large scale SVD and other matrix completion methods to estimate the probable preferences of users for movies [1].

2 Methods

2.1 Data

Description. The training dataset consists of transactions between 3,348,026 sender-receiver pairs and the number of times the sender paid the receiver. The transactions were made between March 2012 to March 2013. The testing dataset consists of sender-receiver pairs which did not interact in the training set time period but may have by the following year, May 2014. The full dataset contains 444,075 unique addresses, and therefore over 19.7 billion possible unordered pairs of addresses. A binary adjacency matrix is extremely sparse, with only 1.7×10^{-5} nonzero values.

Exploration. To better understand the data, we constructed a smaller dataset with three columns: address, outdegree, and indegree. We found a positive correlation between indegree and outdegree (Figure 1a). Surprisingly, while some nodes have high outdegree without a high indegree, no nodes have a high indegree but a low outdegree. We expected the opposite: online merchants would receive many payments but not send money often.

In addition, we also found a strong relationship between the address and the outdegree (Figure 1b) and indegree (not shown). Lower address numbers have higher indegrees and outdegrees. This suggests that the addresses themselves will be useful as features.

2.2 Features

We approached prediction through two views of the data: matrix completion of a binary adjacency matrix and supervised learning on a dataset with graph properties as features.

2

For matrix completion, the dataset is a $444,075 \times 444,075$ adjacency matrix, far too large to store in memory. We used the Python library `scipy` for several representations of sparse matrices which work with `scikit-learn`'s SVD, NMF, and LDA implementations [2] [7].

For supervised learning, we generated a new dataset with six features of the implicit graph in the transaction dataset: (1) sender address, (2) receiver address, (3), sender outdegree, (4) sender indegree, (5) receiver outdegree, and (6) receiver indegree. The number of transactions was dropped from the training dataset because it is not available in the test dataset. We tested two supervised classifiers on this dataset, SVM and logistic regression. Because a full dataset of all possible combinations of addresses is too large (over 19.7 billion samples), we subsampled the dataset, selecting 1,000,000 samples with a hyperparameter pp indicating the percent positive.

2.3 Classifiers

Spotlight: Nonnegative matrix factorization Nonnegative matrix factorization (NMF) is a method for approximating a matrix as the outer product of two other, typically smaller, matrices. Formally, a matrix X is factorized into matrices V and U such that:

$$X \approx V \times U \tag{1}$$

Where X is an $n \times m$ matrix, V is an $n \times k$ matrix, and U is a $k \times m$ matrix. The value k is a hyperparameter of the model which indicate the number of latent variables that approximate V (Figure 2).
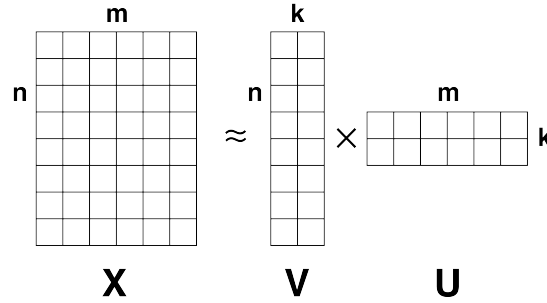

Figure 2: Matrix X is factorized into matrices V and U.

The key assumption is that the approximated matrix has low-rank. In other words, not every cell in the matrix is a free variable. Instead, each cell value is the dot product of two k-dimensional vectors. The value of the low-rank assumption is that it reduces the number of free parameters of a model. In the most extreme case, $k = 1$, a model that would typically have $n \times m$ parameters can be approximated by just $n + m$ parameters. In general, the number of parameters is $k \times (n + m)$.

Prediction for matrix completion methods such as NMF are as follows. First, we approximate the observed data X using matrix factorization (Equation 1). We then complete the matrix by multiplying the two matrix factors together:

$$\hat{X} = V \times U$$

In principle, $\hat{X}$ is a matrix that captures the latent structure in X. We can then base predictions on the value in the completed matrix. For example, since our matrix X contains sender addresses as rows and receiver addresses as columns (or vice versa), then the prediction for whether or not the ith address made a transaction with the jth address would be found in $\hat{X}[i, j]$.

There are many algorithms for solving matrix factorization for NMF [4]. One straightforward way is to use gradient descent to minimize the difference between the observed data and the factorized estimates. Formally, given observation X_{ij}, we want to find low-dimensional vectors v and u such that $X_{ij} = v_i \cdot u_j$ by minimizing the following objective function:

$$f(u, v) = \sum_{i,j \text{ observed}} (X_{ij} - v_i \cdot u_j)^2$$

Ultimately, the learned matrices V and U are uninterpretable. But like many machine learning methods, we assume that if we can approximate the data well with fewer numbers, we might capture generalities in the data. Other dimensionality reduction methods such as SVD and PCA rely on this thinking as well.

Classifiers. We used five classifiers from the `scikit-learn` Python library [7], two which perform supervised learning on graph features and two which use matrix completion. Unless stated otherwise, we used the default parameters for each classifier as implemented by `scikit-learn`.

Supervised learning classifiers.

For both supervised learning models, the full dataset was too large. Instead, we trained each model on random subsamples of the full dataset. In each experiment, the size of the subsample was 1,000,000. We varied the percentage of positive examples in each subsample (pp) and report the results in Table 1.

- **Linear Support Vector Machine** (LSVM): A linear SVM classifier finds a hyperplane which maximizes the distance of the hyperplane from the data being classified. Since `scikit-learn`'s default `SVC` class was extremely slow, we trained a `SGDClassifier` model, which is a linear classifier trained with stochastic gradient descent. We used the hinge loss function, which makes the model equivalent to linear SVM. We trained it online for 100 batches with a batch size of 100000 and 0.1% positive examples.

- **Logistic Regression** (LR): Logistic regression is regression with binary labels. We used `scikit-learn`'s predict˙proba function to get probabilities for the ROC curve and used a threshold for prediction of 0.5.

Matrix completion methods.

- **Singular Value Decomposition.** (SVD): SVD is a matrix factorization and dimensionality reduction technique. It decomposes the original data into a product of 3 matrices: U, Σ and V where Σ is a diagonal matrix while U and V are left-singular and right-singular matrices respectively.

- **Nonnegative Matrix Factorization** (NMF): NMF decomposes multivariate data into the product of two non-negative matrices. By restricting the rank of the matrices, we can create a dense, low-rank approximation of the data. See Section 2.3 for details.

- **Latent Dirichlet Allocation** (LDA): LDA builds a low-rank approximation of the data by using small number of "topics". In our case, we choose `n_topics = 5`. Each topics explains the broad grouping under which each data point can be placed.

3 Results

The results for the performance of SVD and NMF were evaluated on various metrics such as accuracy score, precision, recall and F-1 score. The results have been summarized in Table [TK table]. Moreover, ROC curves for both the methods were analyzed to compare the performances for varying classification thresholds [TK table] . It can be clearly observed from the two ROC curves that NMF outperforms the SVD since the performance of SVD based approach falls below the random guess at a certain threshold.

SVM and Logistic regression. TK TK.

NMF. TK TK.

4

|	LSVM	LR			NMF				LDA	SVD
		pp=0.5	pp=0.1	pp=0.01	k=5	k=10	k=20	k=50		
Accuracy	0.8430	0.6235	0.8580	0.8942	0.8891	0.8696	0.8449	0.8468	0.1000	0.8380
Precision	0.3339	0.1755	0.3697	0.4744	0.4350	0.3584	0.2880	0.2940	0.1000	0.2629
Recall	0.5730	0.7480	0.5960	0.5370	0.3680	0.3850	0.3760	0.3800	0.5868	0.6380
F1 score	0.4219	0.2844	0.4564	0.5038	0.3989	0.3712	0.3265	0.3315	1.0000	0.3440
AUC	0.7230	0.6788	0.7416	0.7354	0.7320	0.7329	0.7230	0.7337	0.1818	0.2980

Table 1: TK TK

(a) TK TK

(b) TK TK
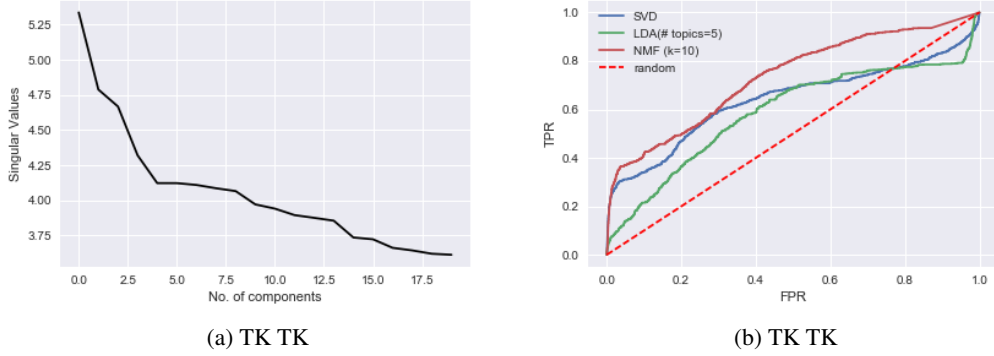
Figure 3: TK TK

SVD. In order to decompose the large sparse data matrix as a product of two low-rank matrices, we analyzed the singular values by plotting these singular values for different components. We decided to select the top 20 components by observing a "knee" in the plot (Figure 3a). Thus, the assumption was that the basic factors determining whether a transaction would take place can be explained by these 20 components.

4 Discussion

Our analysis showed that NMF outperforms SVD and LDA on metrics described in the results section. Moreover, it was observed that LDA was extremely slow even when the number of topics was as small as 5. This could possibly be due to the underlying algorithmic implementation of LDA which is more suited for topic modelling in text documents by iterating over all the words in a document for all the documents in a corpus. The high dimensionality of our dataset makes it computationally intractable to train for larger number of topics.

Two supervised learning methods also worked well. In particular, logistic regression was extremely fast on subsamples of a full dataset of graph-based features. We found that the proportion of positive examples was an important hyperparameter. The model had better accuracy with a lower proportion of positive examples, but a slightly worse recall.

We have therefore approached the task of identifying whether a transaction will occur from two angles. The first viewpoint leverages classification techniques such as SVM and Logistic regression by constructing features from the training dataset using graph properties of indegree and outdegree. The second approach involves viewing the data as a large and sparse matrix of the counts of transactions and employing matrix completion methods like NMF, SVD and LDA to get a low rank approximation of the data and thereby identify the underlying latent structures. In conclusion, the future direction of this work can involve developing complex models which make use of both the classification and matrix completion methods in conjunction with each other so as to better predict transactions.

References

[1] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA, 2007.

[2] Eric Jones, Travis Oliphant, and Pearu Peterson. {SciPy}: open source scientific tools for {Python}. 2014.

[3] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[4] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[5] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[6] Michael Nielsen. How the bitcoin protocol actually works. *Blog posting at www. michaelnielsen. org/ddi/how-the-bitcoin-protocolactually-works*, 2013.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.