Fraud Detection in Signed Bitcoin Trading Platform Networks

Danielle Sullivan, Tuan Tran, Huaping Gu

December 10, 2018

Abstract

In modern e-commerce and cryptocurrency networks user generated reviews provide traders some transparency into the trustworthiness of a potential trading partner. In the application of bitcoin networks, a signed social network is a graph where each user is a node and edges are created per transaction review with edge weights corresponding to trading relationship sentiment. Knowing that users rely on these rating platforms to help them make trading decisions, criminals have taken advantage of this signed social network structure to make themselves or a person or community of their choosing appear more credible.

In this paper, we propose 2 novel ensemble node embedding methodologies ReFex +/- and ReFex REV2 +/- for signed social network applications. For each of these methodologies we extend the original ReFex algorithm, created for positive weight networks, to handle signed social networks. Furthermore, for the ReFex REV2 +/- we add the REV2 fairness and goodness score of a user to the basic ReFex feature list. We then use these embeddings along with ground truth labels to develop models to classify unlabeled nodes. We apply our embedding methodologies to Bitcoin trading platform dataset, and show that including REV2 scores as features of the ReFeX embeddings can help improve the prediction accuracy. Specifically, we were able to achieve an average AUC score of greater than 90% using a decision tree model trained on our ReFex REV2 +/- node embeddings. This model is robust to the percent training data used, with an average AUC greater than 90% for all percentages of training data tested between 10%-90%. Finally, we used our node embedding as inputs to K-Means/HDBSCAN to see if nodes of the same classification clustered together. The result of clustering nodes based on their feature vectors show that for clusters with many ground truth nodes, these nodes are typically dominated by one type of ground truth users, either good or bad. This indicates that our algorithm succeeds in producing embedded vectors that could be used to separate between good and bad users.

1 Introduction

Bitcoin is a pseudo-anonymous decentralized cryptocurrency built on the blockchain technology [13], in which each user is uniquely identified by their anonymous public keys and authorize payments via their private keys; their transaction history, however, is transparent. Thanks to the use of blockchain

as a transaction ledger, bitcoin is fully decentralized and resistant to data modification. Unfortunately however, it is not by any means immune to scams, frauds, and theft. These risks include account hacking, platform hacking, fraudulent cryptocurrency and exchange platforms.

Users in a bitcoin network could indicate their trust to another user; thus a Bitcoin network could be thought of as a weighted graph with positive edges indicating trust and negative edges indicating distrust. Because users in a Bitcoin network prefer to trade with others who have a history of positive feedback ratings, there is a huge monetary incentive for fraudulent users to increase their perceived trustworthiness. The bitcoin network is anonymous and decentralized, making it almost impossible to link a fraudulent users to their deceitful transactions. The anonymous nature of the Bitcoin network makes relying on features outside of those provided by the network structure difficult. As a result, the existence for an algorithm that could adequately represent the features of nodes in a Bitcoin network through embedded vectors is crucial for many machine learning and fraud detecting job on a Bitcoin network.

In this paper, we present ReFex +/- and ReFex REV2 +/- feature learning algorithms to generate embedded vectors for each user in a Bitcoin network. We experimented with the two algorithms ReFeX [5] and REV2 [9] and combined them in multiple ways to find the best method for feature learning. Our result, ReFeX +/-, produce a feature vector for each node by combining the feature vectors received from two ReFeX runs, one on positive edge subgraph of the Bitcoin network, and one on the negative edge subgraph. This algorithm could be combined with features received from REV2 to improve the representation capability, resulting in ReFeX REV2 +/-.

Our contribution from this paper includes:

- Experiment with multiple ways of combinig ReFeX and REV2 to improve the feature learning capability, resulting in two algoritms ReFeX +/- and ReFeX REV2 +/-
- Show that these two algorithms gives better learning capability compared to current algorithms.
- Explore several methods of classifying nodes based on the embedded vectors returned by our algorithm, and showing how HDBSCAN [12][3] and Decision Tree could be sucessfully combined with our algorithm to give a better understanding of a Bitcoin network structure.

2 Related Work

2.1 BIRDNEST Algorithm

One tactic that businesses often employ to increase the popularity of their products is using fake positive ratings. The fraudulent users giving these ratings could be detected through their skewed rating distribution and the irregularity of their ratings. The BIRDNEST fraud detection algorithm, presented by Hooi et al. analyses the review distribution and average ratings of users to identify fraudulent reviewers and fraudulently rated products using Bayesian inference [6]. Abnormality in review intervals, such as huge spikes in the number of ratings given in a short period of time, or periodic ratings, are also taken into account.

BIRDNEST was tested on Flipkart product reviews network, where it identified 250

of the most suspicious users of which 211 we identified as fraudulent users. While the it delivers impressive results, BIRDNEST does not take advantage of more useful information that could be extracted from the review network structure, such as network cluster, nodes' roles, among others. It also specifically focuses on e-commerce sites and thus could be unsuitable for detecting uncommon schemes in e-commerce, such as Ponzi [2] schemes.

2.2 REV2 Algorithm

Kumar et al developed the REV2 algorithm to predict fraudulent users in ratings platforms such as e-commerce platforms. REV2 defines three interdependent quality metrics to predict fraudulent users product quality, review reliability, and user fairness. The authors then established six logical relationships between these metrics: for example, a fair user is more likely to give reliable reviews, a good product is more likely to receive a reliable positive review, etc. By formalizing these relationships through mathematical formulas, they define a system of linear algebra equations calculating each score which can be solved for through iterative methods.

The authors of REV2 also integrates other methods such as smoothing coefficients to deal with the "cold start problem" and BIRDNEST to punish suspicious user patterns such as burstiness in active time, etc. The complete REV2 formulation is then solved iteratively. The REV2 authors compared their algorithms performance to nine state-of-the-art fraud detection algorithms, with REV2 outperforming all of them on extensive experiments on 5 datasets. Additionally, the algorithm has a linear running time with the number of edges. However, we believe that by augmenting the REV2 with more bitcoin network-specific local features, performance could be improved.

We believe that by adding simple node and egonet level properties, we could lerage more network properties to aide in node classification. Network properties such as in and out degrees and edge count within an egonet will be similar for the good and bad users in the network. This belief is based on the fact that good an bad users haven shown interact in ways that create distinct node level and egonet network embeddings. For example, Pandit et al. explain how fraudsters interact with users via the a handful of "accomplice nodes" but never directly with other fraudulent or good users. Conversely, friendly users may interact with a couple accomplice node in addition to many other friendly nodes who will make 2-way trades with them. [14] These unique trading patters will result in unique network properties.

2.3 ReFeX

An important task in learning about a network is to encode the structure of the network into feature vectors for use in a wide array of learning tasks. Henderson et al propose ReFeX [5], which is a recursive algorithm for learning about the structure of the neighbor network of each node. The algorithm starts by assigning the degree of each node as the starting feature. In addition to degree, other features can be appended to the feature vector. The algorithm then recursively appends the mean and sum of the feature vectors of the neighbors of a node to its feature vector. By recursively apply this algorithm, we end up with a feature vector that could represent the structure of the network on a local scale.

One notable drawback of the algorithm

is that the dimension of feature vectors increases exponentially, causing the curse of dimension for learning algorithm using ReFeX's result. The authors propose to mitigate this problem by pruning features that are highly correlated. Another drawback of this algorithm in respect to signed social network applications, is that it does not differentiate between negative and positive weight edges in a network. Kim et al. have explored how positive and negative weight edges have very different interpretations and should be treated differently [7].

3 Dataset Description

For our experiments, we use the signed social network dataset from the Bitcoin trading platform Bitcoin Alpha. To calculate the REV2 fairness and goodness score, we use the REV2 code provided to convert the dataset to a bipartite graph, where one set of the graph is the rater user and the other set is the user whose trustworthiness is being rated (product). Edges in the graph are always directed and originate in the rater set of nodes with destination always being a node in the set of users being rated. The edge weights correspond to the strength of a raters sentiment. The ground truth of each dataset was determined by the respective platforms founder. Users are identified as trustworthy if the founder identified them as trustworthy, a trustworthiness rating $\leq$ .5. Conversely users were identified as non trustworthy if the found assigned them a trustworthiness $\leq -.5$. We use the original user to user network (non-bipartite graph) to extract the network properties used in the ReFeX and ReFeX +/- algorithms.

# Users	# Edges	% fair	% unfair	% labeled
3782	24185	3.57	2.70	6.27

Table 1: Bitcoin Alpha Dataset Statistics

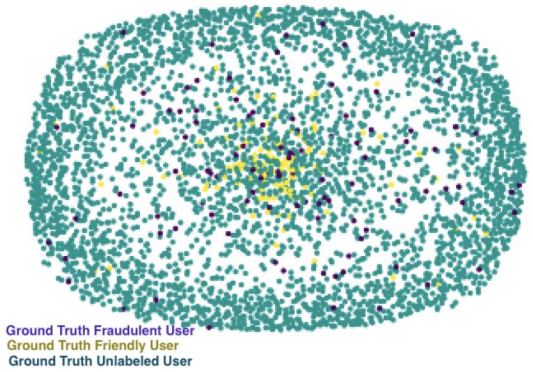


Figure 1: Original Network Plotted Using Networkx Spring Layout

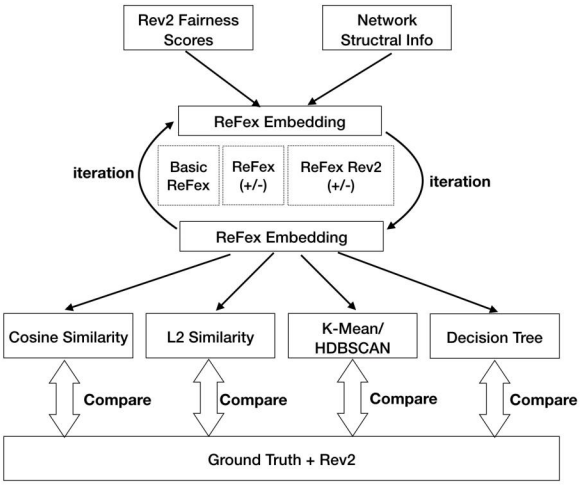
4 Methodology and Algorithm

For our project, we developed an ensemble model that uses basic node level and egonet level properties such as node in and out degree, and intra egonet edge count in addition to the more complex REV2 score to create recursive node embeddings using the ReFeX algorithm. We then use these embeddings along with ground truth labels to develop algorithms and models that can potentially be used to classify unlabeled nodes.

4.1 Embedding Methodologies

For our experimentation, we define 3 different node vector embeddings. These embeddings are all based on the ReFeX algorithm. The baseline ReFeX vector embedding for our experiments is $\tilde{V}_u^{(i)}$ where i is the level of recursion. $\tilde{V}_u^{(0)}$ is the set of 4 basic local node features. Prior to recursion, there is a a feature vector $\vec{V(u)} \in R^4$ for every node u. The features include the following: (1)

Figure 2: Methodology and Algorithm



the in-degree of node v, (2) the out-degree of node v, (3) the number of edges within the egoNet of v, (4) the number of edges connecting the egoNet of v to the rest of the graph. We then recursively extend the features of v, using the following formula:

$$\tilde{V}_u^{(1)} = [\tilde{V}_u^{(0)}; \frac{1}{|N(u)|} \sum_{v \in N(u)} \tilde{V}_v; \sum_{v \in N(u)} \tilde{V}_v]$$

Where N(u) is the set of us neighbors in the graph. If $N(u) = \emptyset$, set the mean and sum to 0. After K iterations, we obtain the overall feature matrix

$$V = \tilde{V}^{(K)} R^{3^{K+1}}.$$

For our experiments for each of the embedding features, we will do 3 iterations. Since we includes the neighbors edges and its egoNet edges, with the iterations, we can detect the users who have similar local and global (neighbors neighbor) network pattern, then help us detect the community the user belong to.

Because we believe there is an intrinsic difference between the meaning of negative and positive weights within a network, our second version of vector embeddings,ReFeX +/-, is the same as the traditional ReFeX algorithm but we separate the network into a network of all positive weights, and a second network of all negative weights. We then append the ReFeX embedding of the positive weight only network to the embedding for the negative weight only network.

$$\tilde{V}_{+/-u}^{(1)} = [\tilde{V}_{+u}^{(1)}; \tilde{V}_{-u}^{(1)}]$$

Finally we define the ReFeX REV2 +/- vector embedding, which is nearly identical to the ReFeX +/- embedding with 2 additional local feature added, REV2's Fairness(v) and Goodness(v) values.

We combine REV2 features together with other network structural features and compare the results with and without REV2.

4.2 REV2 Score Calculation

We calculated REV2 user fairness and goodness scores by running the code provided by the REV2 authors on the same dataset used for their paper's results with the parameters they noted gave the best results. The parameter values which gave the best results are the following:

$$\alpha_1 = 0, \alpha_2 = 0, \beta_1 = 0, \beta_2 = 0,$$
$$\gamma_1 = 0.01, \gamma_2 = 0.01, \gamma_3 = 0$$

We experimented with other combinations of values for these parameters, and appended each resulting pair of fairness and goodness to our vector embeddings. However, we did not see any improvement in prediction performance. Because of this, we decided to only use the pair of fairness and goodness values resultant from all 0 parameters.

In REV2, the original network is transformed into an bipartite network, where nodes

are divided into two types, users and products. User nodes only have out-coming ratings, and product nodes only have in-coming nodes. Only user nodes have fairness score, while product nodes have goodness score. These are the values we use in our ReFeX REV2 +/- algorithm.

4.3 Similarity Algorithms

To quantify how similar the vector embeddings of 2 nodes are. We use the Cosine and L2 similarity formulas.

4.3.1 Cosine Similarity

For any pair of the nodes u and v, we use cosine similarity to measure how similar two nodes are according to their feature vectors x and y:

$$Sim_{cosine}(x,y) = \frac{x.y}{|x|^2.|y|^2} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}.\sqrt{\sum_i y_i^2}}$$

$$|x|^2 = 0 \quad or \quad |y|^2 = 0, \quad Sim(x,y) = 0.$$

4.3.2 L2 Similarity

Similarly, we also calculate the l2 distance for each pair of the nodes u and v.

$$Sim_{l_2}(x,y) = \sqrt{\sum((x-y)^2)}$$

4.4 Clusterings

The embedded feature vectors have a very high dimension, posing a challenge to effectively learn and visualize about them. To mitigate this issue, we first use t-SNE [11] to reduce the number of dimensions down to two. Then we apply two clustering algorithms and graph their results to visualize clusters with a high number of good/bad ground truth nodes.

4.4.1 K-Means Algorithm

K-Means Algorithm: We first use this classic clustering algorithms to cluster our data points. One downside of K-Means is that the clusters produced are usually round in shape and this cluster shape characteristic does not represent the structure of the our generated node embedding vectors naturally.

4.4.2 HDBScan

We also use HDBScan [12][3] to overcome this cluster shape issue. HDBScan uses the density of the nodes in the vector space to detect the nodes that are naturally connected with one another. The algorithm builds a spanning tree of the nodes in the vector space and group nodes that could be reached through short edges together. The downside is that this algorithm may and up grouping nodes that are very far from each other to the same cluster if the happen to be closely connected to a string of other nodes.

5 Experimental Evaluation

In this section we assess the effectiveness of our methodologies by presenting our experimental results. All of our experiments were conducted on the bitcoin Alpha dataset. Our experiments show these major results (i) We compare our custom algorithms (ReFeX +/- and ReFeX REV2 +/-) to standard ReFeX and the original REV2 algorithm and see that, REV2 performs significantly better than our custom models. However our ReFeX

+/- algorithm performs the best in comparison to all of our custom models is robust to the amount of training data, and offers slight improvement in comparison to the plain ReFeX algorithm that ignores edge weight sign. (ii) We also show that using our vector embeddings we are able to make use of classification and clustering algorithm such as Decision Tree, k-Means, and HDBSCAN to detect the groups of users.

5.1 Baselines

We compare our ReFeX +/-, and ReFeX REV2 +/- algorithm performance with the original ReFeX algorithm and REV2 algorithm. The baseline algorithms are implemented as described in their corresponding subsection in the background section.

5.2 Experiment 1: Naive Similarity Evaluation

For our first experiment, we completed a naive analysis of our embeddings from the ReFex +/- and ReFex REV2 +/- embeddings. This gave us a rough idea on whether the REV2 features improve the classification precision with our augmented ReFeX methodologies.

We do above calculation for ReFex +/- embedding with and without REV2 features, from the diagram below, it is obvious that REV2 features helps on prediction. Because of the naiveness of this method, we not expected higher precision (Y axis), but it does help us make the decision to move forward with our analysis below.

In this step, we also compared the performance of similarity algorithms Cosine and L2, overall L2 shows better performance on prediction. Hence in our final algorithm above,

Algorithm 1 Experiment 1: Naive Similarity Evaluation

```
for K in range(5 to 50 by 5) do
  for user u in ground truth do
    for other node v in Graph do
      similarity score = (sim_L2(u, v) + sim_cosine(u, v))/2
    end for
    nodes = scores.sort(desc)
    top_k = nodes[:K]
    last_k = nodes[:-K]
    p_top_k = intersect(top_k, gt nodes of same class)/K
    p_last_k = intersect(last_k, gt nodes of opposite class)/K
    precision = (p_top_k+ p_last_k)/2
  end for
end for
```

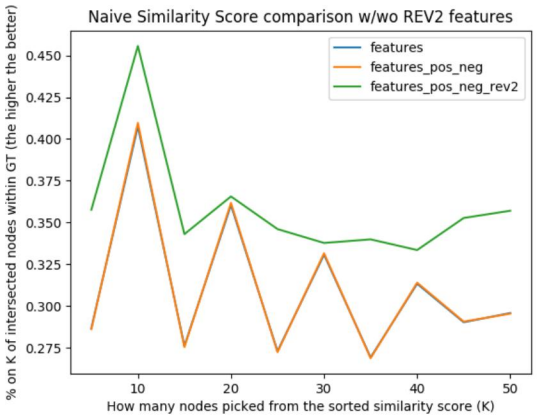


Figure 3: Naive Similarity Score (ReFeX +/-) vs. (ReFeX +/- Plus REV2)

we use an average value of cosine and L2. In this experiment, we believe the selection of similarity algorithms does not impact the final result.

We also aware that bad user's friends not have to be bad user, and same for good user. In this Naive algorithms we does not care about the precision, but focus more on the precision **delta** between (ReFeX +/-) and (ReFeX +/- Plus REV2).

5.3 Experiment 2: Supervised Similarity Prediction

For this set of experiments, we aim to rank the users on how similar their node embeddings are to fraudulent "anchor node" embeddings. We define anchor nodes as fraudulent nodes we have selected from the ground truth data at random. Using the anchor nodes, will give all of the other users in the network a score based on their similarity to the set of fraudulent anchor nodes. We define this similarity score as:

$Score(u) = max(a \in I_a Sim(a, u))$ where I_a is the set of anchor nodes.

The similarity formulas we evaluated for our experiments were Cosine similarity and L2 similarity.

We measure the performance of the similarity scores for each of the embeddings using the Average Precision scores, which measures the relative similarity orderings of each of the nodes from the embedding algorithms. Performance is evaluated on all nodes with ground truth labels, and corresponds to the area under the precision-recall curve. Figure 4 shows the precision@K results, displaying how the Average Precision values change as we increase the number of anchors nodes. Overall, using the similarity of a node's embedding to an anchor node is not an effective classification technique as seen in the figure. Furthermore, including the ReFex +/- and REV2 features in our embeddings makes almost no difference in Average Precision, leading us to believe our anchor node based classification mode is ineffective using our embbedding methodologies.

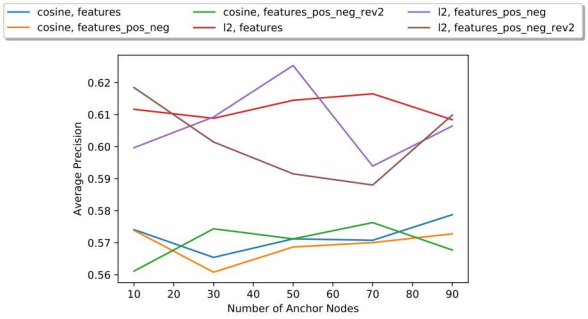


Figure 4: Average Precision of Similarity Based Prediction

5.4 Experiment 3: Supervised Embedding Based Classifier

For our third set of experiments, we trained decisions trees to predict the class of a node based on the vector emeddings produced from our 3 node-embedding algorithms. Decision trees are a type of supervised classification model, which are trained by splitting at decision nodes with respect to certain metrics, such as minimizing information impurity post-split. Classification on test data can then be made by walking down the tree to a leaf node after the tree has been trained.

Figure 5 summarizes the AUC@K results, showing how the AUC values change as we increase the number of nodes used in the training set. This figure shows the results from our algorithms in addition to the average AUC reported from the REV2 authors

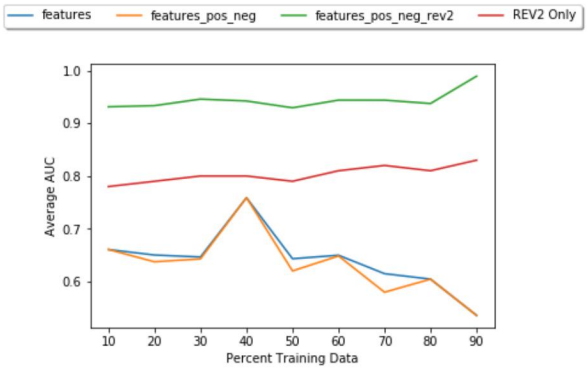


Figure 5: AUC of Decision Tree Classifiers compared to Reported REV2 Results

in their paper. Here we see, our vector embeddings from ReFeX REV2 +/- perform the best, consistently getting an average AUC score greater than 90%.This leads us to believe that the behavioral and network properties captured in the REV2 fairness and goodness scores, in addition to the positive and negative network ReFeX embeddings gives the best feature set for classifying a node. Interestingly, we see no improvement in performance when we create embeddings on traditional ReFeX compared to our ReFeX +/- which treats the positive and negative weight edges properties of the network as different features.

5.5 Experiment 4: Clustering Users

For this experiment we cluster users based on their embedded feature vectors and check which cluster has a high concentration of ground truth good and bad users. To improve the effectiveness of the algorithm and facilitate visualization, we preprocess the feature vectors with t-SNE [11] to reduce each feature vector to only two elements without sacrificing their similarity. Then we cluster these new feature vectors using k-Means and HDBSCAN, and plot these clusters along

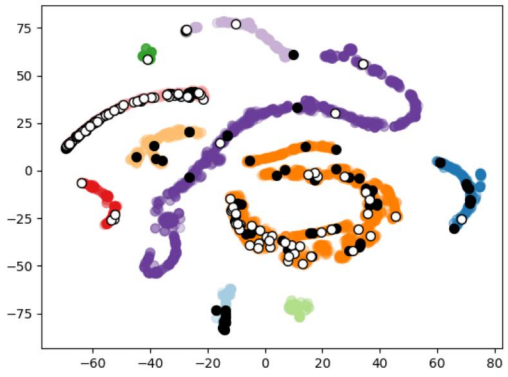


Figure 6: Clustering of ReFeX REV2 +/- embeddings using HDBSCAN

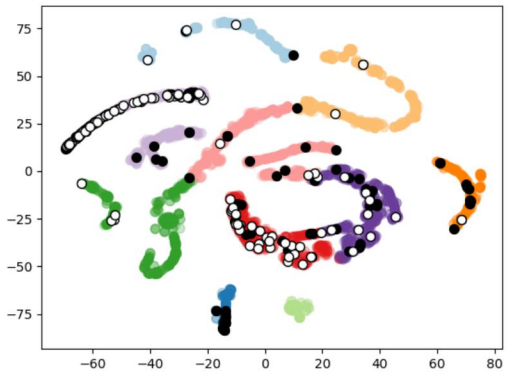


Figure 7: Clustering of ReFeX REV2 +/- embeddings using k-Means

with the ground truth good and bad users. The results are shown in Figure 6 and 7. In both figures, the black nodes are ground truth bad users, white nodes are ground truth good users, and each color is a cluster.

Our first observation is that the ground truth users concentrate much more in some clusters than others, with some clusters having very few ground truth datapoints. This puts a limit on testing the accuracy of our algorithm. On the other hand, ReFeX REV2 +/- gives embedded feature vectors that are naturally clustered, and the clusters that do

have many ground truth users all have their ground truth users overwhelmingly lean towards one side, either good or bad users. This result indicates that ReFeX REV2 +/- successfully captures the main properties of the nodes in the network.

For the clustering algorithm itself, we see that HDBSCAN clusters users much more naturally than k-Means and would be our recommendation for the clustering method for our algorithm. On improvement that could be made to HDBSCAN is the set a maximum diameter of each cluster to avoid creating a sprawling cluster.

6 Conclusion

In this paper, we presented the ReFex +/- and the ReFex REV +/- ensemble node embedding algorithms.

- Algorithm: For both ReFex +/- and the ReFex REV +/-, we define a way to extend the ReFex algorithm to directed signed social networks. Additionally in ReFex REV +/-, we extend the ReFex embedding model beyond it original base feature, and include the REV2 goodness and fairness scores of a user to capture complicated behavioral and network properties.
- Effectiveness: While our "anchor node" similarity classification methodology is ineffective compared to current state of the art fraud detection algorithms, our decision tree based model performed well. It consistently achieved greater than 90% average AUC while being robust to the amount of training data used.
- Cluster Representation: The result of clustering nodes based on their feature vectors show that for clusters with many ground truth nodes, these nodes are typically dominated by one type of ground truth users,, either good or bad. This indicates that our algorithm succeeds in producing embedded vectors that could be used to separate between good and bad users.

7 Acknowledgements

We would specifically like to thank Jure Leskovec for the great class this semester and Srijan Kumar for his thoughtful guidance on our project. Additionally, we would like to thank our TAs and classmates for their quick and informative responses on Piazza. This made learning complex (but interesting!) material possible.

References

[1] cs224w group project github repo, https://github.com/dr-lab/cs224w.

[2] Bartoletti, M., Pes, B., and Serusi, S. Data mining for detecting bitcoin ponzi schemes. *CoRR abs/1803.00646* (2018).

[3] Campello, R. J. G. B., Moulavi, D., and Sander, J. Density-based clustering based on hierarchical density estimates. In *Advances in Knowledge Discovery and Data Mining* (Berlin, Heidelberg, 2013), J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds., Springer Berlin Heidelberg, pp. 160–172.

[4] Grover, A., and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (2016), ACM, pp. 855–864.

[5] Henderson, K., Gallagher, B., Li, L., Akoglu, L., Eliassi-Rad, T., Tong, H., and Faloutsos, C. It's who you know: Graph mining using recursive structural features. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2011), KDD '11, ACM, pp. 663–671.

[6] Hooi, B., Shah, N., Beutel, A., Günnemann, S., Akoglu, L., Kumar, M., Makhija, D., and Faloutsos, C. BIRDNEST: bayesian inference for ratings-fraud detection. *CoRR abs/1511.06030* (2015).

[7] Kim, J., Park, H., Lee, J.-E., and Kang, U. Side: Representation learning in signed directed networks. In *Proceedings of the 2018 World Wide Web Conference* (Republic and Canton of Geneva, Switzerland, 2018), WWW '18, International World Wide Web Conferences Steering Committee, pp. 509–518.

[8] Kumar, S., Cheng, J., Leskovec, J., and Subrahmanian, V. S. An army of me: Sockpuppets in online discussion communities. *CoRR abs/1703.07355* (2017).

[9] Kumar, S., Hooi, B., Makhija, D., Kumar, M., Faloutsos, C., and Subrahmanian, V. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (2018), ACM, pp. 333–341.

[10] Kumar, S., Spezzano, F., Subrahmanian, V., and Faloutsos, C. Edge weight prediction in weighted signed networks. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on* (2016), IEEE, pp. 221–230.

[11] Maaten, L. v. d., and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research 9*, Nov (2008), 2579–2605.

[12] McInnes, L., Healy, J., and Astels, S. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software 2*, 11 (2017), 205.

[13] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system.

[14] Pandit, S., Chau, D. H., Wang, S., and Faloutsos, C. Netprobe: A fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th International Conference on World Wide Web* (New York, NY, USA, 2007), WWW '07, ACM, pp. 201–210.

Appendix

7.1 node2vec Algorithm

An important part of learning about roles of nodes in network is to categorize them into suitable buckets. By correctly grouping similar nodes, we could quickly categorize nodes given the known roles of a few nodes and predict the possibility of connection between nodes given the graph's structure. For our bitcoin signed network, this means if we could find a few initial fraud users we could quickly find similar fraud users and predict their future transaction, timely preventing damage from their scam. This process is often achieved through node embedding, which is the process of representing a node in a network as a low dimensional feature vector.

Grover et al. proposes node2vec, an algrorithm designed to categorize the roles of nodes in a graph reliably using only the structure of the graph itself [4]. node2vec uses random walks from nodes in a graph, with parameters to prevent returning to a node already in the walks and sets bias in favor of a certain length of walk. Since the algorithm only utilizes features of the graph structure itself, it is trivial to generalize for a wide variety of problems. The algorithm is also proved to outperform all other algorithms solving the same problem, and scale linearly with the number of nodes in the graph. It however does not specify an approach to negative node weight, which is prevalent in a bitcoin signed network, and thus could be unsuitable for our challenge.

7.2 SIDE Algorithm

The SIDE algorithm for extracting node embeddings by Kim et al., which leverages theories from socio-psychology to embed nodes from a signed social network into representative vectors [7]. SIDE utilizes three main techniques. First, it uses balance theory to predict the sentiment of two nodes based on the sign of the links between them. Second, it takes advantage of the principle that two nodes with similar properties are more likely to be connected. And third, it realizes that connections are asymmetric and determined on a per node basis.

The SIDE Algorithm is made up of 2 stages. The first stage builds a set of random walks starting a each node, and the second stage of the algorithm uses these random walks to perform gradient descent on the node embeddings. The likelihood optimization equation is based on the principles described above. This algorithm is more suitable than node2vec for signed networks as it takes the sign of edges into its calculations. However, this algorithm may not be suitable for fraud detection applications as frausters have shown to avoid each other in real world networks, in attempts avoid being caught in mass. Pandit et al. explain how fraudsters interact with users via the role of an accomplicate but never directly with other fraudulent users.[7]

7.3 Sockpuppet

Many online communities have sockpuppet users, fake users who create multiple identities to deceive others or manipulate discussions. Srijan et al did the study of the sockpuppet [8] in 9 online communities to identify basic features that can be used to identify sockpuppets: similar names, emails, linguistic traits, arguments, suspicious behaviors, abnormal network structure, etc.

The techniques they developed to identify sockpuppet users include filtering IP addresses, linguistic traits, activities and interactions of sockpuppet. The methods used in identifying these traits could be adapted to identifying suspicious traits in the bitcoin network. Some of the feature definitions, like activity features (how they create posts, participate in discussions and subdiscussions), community features (the users they interact with and how positive the interactions are) and post features (word choice, hoaxes and deceptive styles) also can be applied to bitcoin network with little necessary modifications.Since the paper focuses on online forums only, features used in detecting sockpuppets such as linguistic traits and usernames would not be applicable for a bitcoin network.

7.4 Dataset Analysis

We explored summary statistics of the two user classes, fraudsters and friendly users, in our bitcoin network to determine if there were significant general quantifiable differences in the properties of users from each group.

The first summary statistics we looked at were indegree to outdegree and histograms, see figure 8 in Appendix. This histogram revealed that fraudsters had a much lower outdegree median of 2, where the median outdegree for friendly users was 14. This makes sense, as we do not expect fraudulent users make many payments to other users. We also noticed that the average indegree of friendly users was much higher comparatively. Initially, We expected fraudsters to have a high indegree because they were accepting a large number of payments. One possible explanation for this trend is that fraudsters create multiple accounts, and abandon accounts quickly after making a small number of transactions.

Presumably, a fraudsters main goal in a bitcoin network is to maximize profits. One way to achieve this goal, is to have a larger network of users you accept payments from compared to the size of the network of users you make payments to. To explore this hypothesis, we compared the length of the list of unique users a user accepted payments from (incoming edges) to the length of the list of the number of users they made payments to (outgoing edges), see figure 10 in Appendix. An interesting trend we see here is that fraudulent users are much more likely to have a larger network of nodes they accept payments from compared to the network of nodes they make payments to.

Next we wanted to explore the summary statistics for the time between payments for each user type, see Appendix figure 9. In the corresponding plots we see that fraudulent users exhibit the bursty behavior described the the REV2 authors. Fraudsters have a comparatively lower mean and median for incoming, outgoing and all payment types compared to friendly

users.

From the initial summary statistics shown above, we will try adding a behavioral component to the REV2 algorithm, that penalize a users trustworthiness if the have a ration of into coming to outgoing neighbor set $< .25$ and they have and average δ T between transactions < 1 second.

7.5 Algorithm

Algorithm 2 Calculate Per Node Embedding

for user in Graph.getNodes() **do**
fairness = REV2 algorithm score for user's fairness and goodness
ReFeX vector = ReFeX node structural embedding
ReFeX + vector = ReFeX node structural embedding of positive rating network
ReFeX - vector = ReFeX node structural embedding of negative rating network
ReFeX +/- Plus REV2 vector = Include fairness score as feature to calculate an augmented ReFeX node embedding
end for
Analyze these embeddings using 4 different clustering / classification algorithms, Cosine Similarity, L2 Similarity, K-mean and Decision Tree.

7.6 Figures

7.7 Feature Augmentation

We then combine all these features together, and run recursive iteration to augments the features. We use mean and sum as aggregation functions.

Initially, we have a feature vector $\vec{V(u)} \in R^3$ for every node u. In the first iteration, we concatenate the mean of all us neighbors feature vectors to $\vec{V(u)}$, and do the same for sum,

$$V_u^{(1)} = [V_u; \frac{1}{|N(u)|} \sum_{v \in N(u)} \tilde{V_v}; \sum_{v \in N(u)} \tilde{V_v}]$$

where N(u) is the set of us neighbors in the graph. If $N(u) = \emptyset$, set the mean and sum to 0.After K iterations, we obtain the overall feature matrix

$$V = \tilde{V}^{(K)} R^{3^{K+1}}.$$

For each of the embedding features, we will do 3 iterations. Since we includes the neighbors edges and its egoNet edges, with the iterations, we can detect the users who have similar local and global (neighbors neighbor) network pattern, then help us detect the community the user belong to.

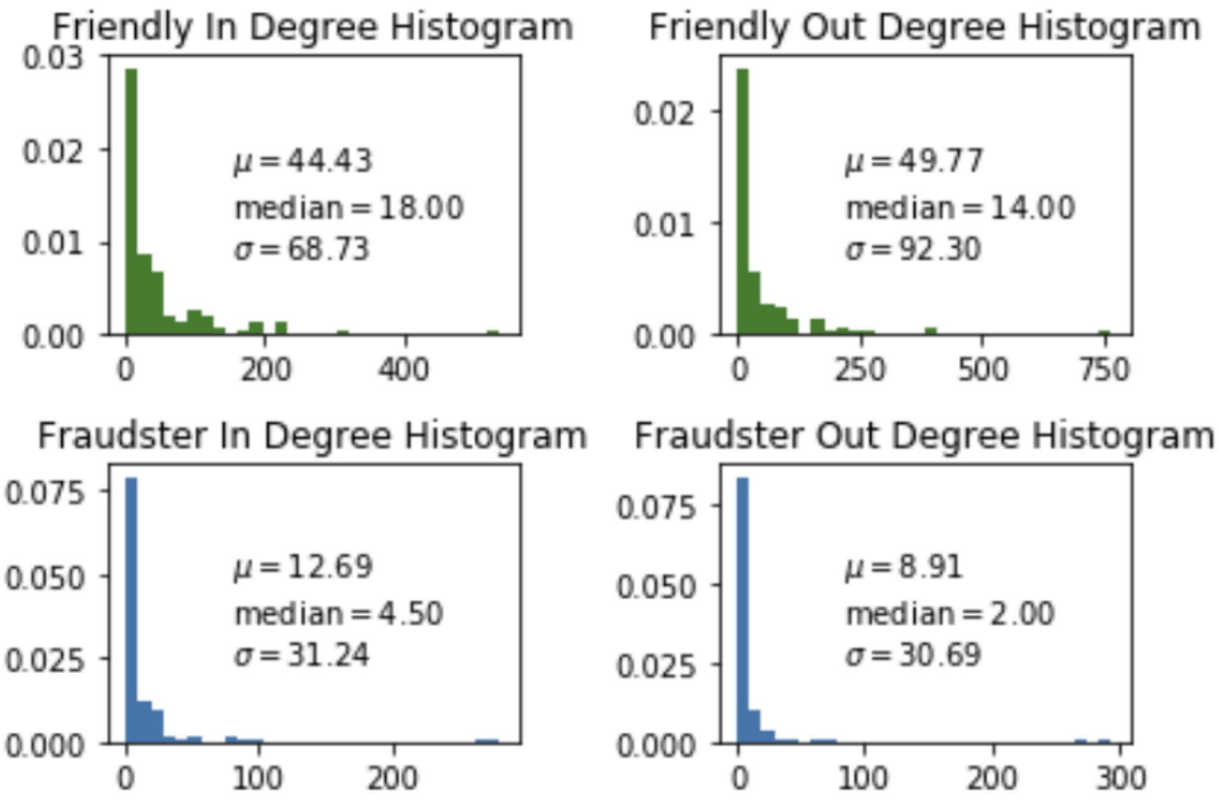


Figure 8: Node Indegree and Outdegree Histograms

Figure 9: Time between transactions

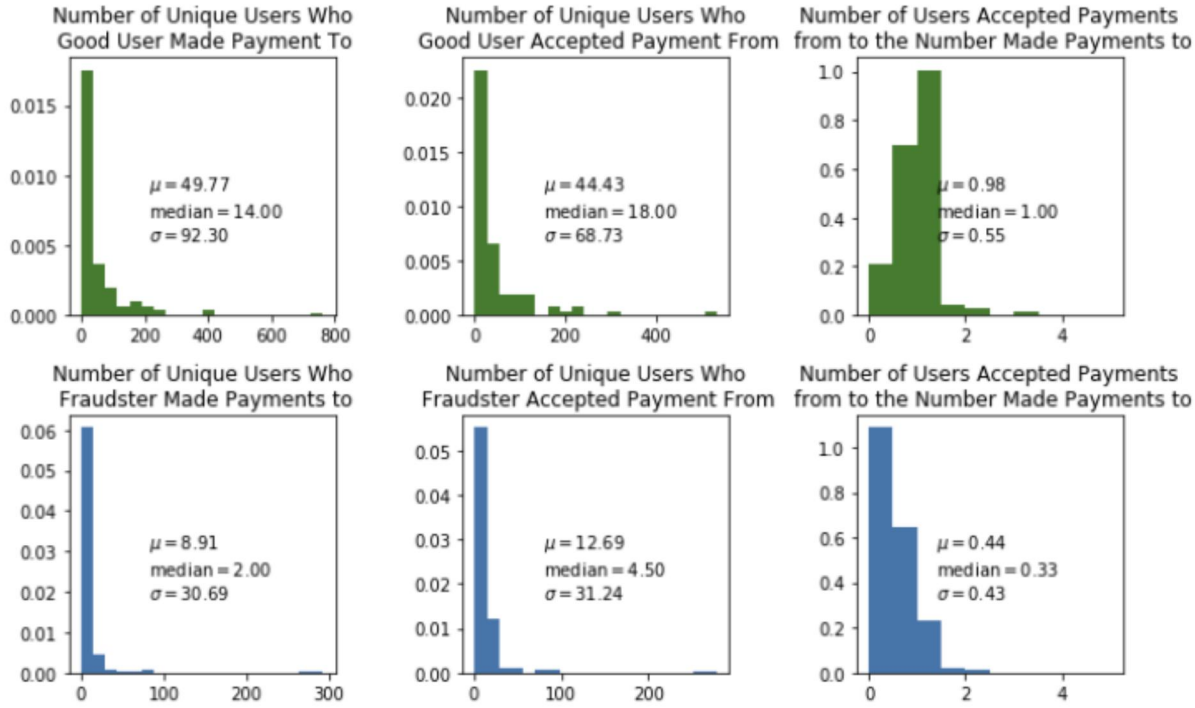


Figure 10: Number of Unique Incoming and Outgoing Trading Partners