DETECTING BITCOIN FRAUD WITH GRAPH EMBEDDINGS

Sarah Spiteri, Cristina Susanu, Andrés Virgüez
Barcelona School of Economics Class'22
Master Project Summary
e-mail: sarah.spiteri@bse.eu, cristina.susanu@bse.eu, andres.virguez@bse.eu

The anonymity of the financial assets based on distributed ledger technologies (DLTs), such as Bitcoin, Ethereum, Ripple, or DogeCoin, to name a few, have accelerated the expansion of fraudulent transactions and criminal activities on the trading platforms. Although the decentralised nature of this technology lies at the core of cryptocurrencies, growing concerns are raised regarding the legal, regulatory, and institutional implications of their underlying activities.

The objective of this project consists in finding the most suitable methods to improve the classification of fraudulent and non-fraudulent Bitcoin addresses by extracting covariates associated with Bitcoin transactions via graph embedding techniques. To meet this objective, the project was split into three main parts: (i) graph sampling; (ii) feature creation by extracting embeddings and by pre-processing non-embedded ones; and the (iii) training of multiple classifier algorithms and assessing expected results.

First, given the large size of the full Bitcoin network in the available timestamps, five methods of graph sampling were tested in order to have a set of representative sub-graphs that would be more reasonable to handle during the classification task. Second, two graph embedding algorithms were applied to the sampled sub-graph to extract the latent features of each node, thus converting the graph into a low-dimensional vector space representation. Next to the newly created features, several non-embedded attributes were added based on the topological attributes of the nodes in the original graph, bringing thus useful information related to the initial network properties. Finally, with a concatenated training data set consisting of both embedded and non-embedded features, we proceed with the classification task by running multiple machine learning algorithms to identify higher-risk fraudulent bitcoin addresses.

Our findings suggest that the choice of the embedding algorithm makes a difference in the final results, the same being true for the graph sampling technique. We also find that several non-embedding features related to the original graph structure are particularly relevant for subsequent predictions, such as in the case of self-loops. In terms of classification performance, the XGBoost classifier tends to consistently return the highest levels of accuracy based on the scores of F1, Precision, and Recall. In this sense, the average F1-score for both classes, summarised in Table II, is 0.932 for the node2vec embedding algorithm and 0.906 for trans2vec.

I. Sub-sampling the Bitcoin Network

In this step, we have experimented with five distinct methods of graph sampling. Four of these methods are following the approaches found in the literature on blockchain forensics, while the last one is a method of our own that mixes the traditional procedures of graph sampling with the specific needs of our study, characterised by a highly imbalanced distribution of classes.

The agreed target of graph sampling was of 50 thousands nodes conditioned by the computational restrictions that we faced. The graph sampling algorithms used to extract a representative subgraph were:

- **Random walk sampling (RWS)** - selects at random one node and then simulate a random walk by selecting neighboring nodes uniformly and randomly on the graph;
- **Centrality sampling (CS)** - based on a random selection of nodes among the list of non-fraudulent addresses while also making sure to include all of the fraudulent nodes. Given that the Bitcoin network allows for the "mixing" of addresses and the opening of multiple accounts per user, most of the non-fraudulent addresses are nodes with low degrees, which led us to decide on keeping only the nodes from the full graph whose eigenvector centrality is within the top 15%;
- **Degree sampling (DS)** - broadly follows the previous CS algorithm, but instead of sampling nodes completely at random, we sample nodes based on a probability distribution. The probabilities of being sampled are assigned to nodes based on their degree, such that the higher the degree the more likely a node is selected;
- **Neighbours sampling (NS)** - selects at random two equally sized sets of high risk and low risk addresses and include all of their neighbours to construct the sampled graph. The inclusion of neighbours increases exponentially the size of the graph, hence, the size of root nodes was restricted to 600;
- **Mixed sampling (MS)** - combines the RWS method with the inclusion of the full set of high risk addresses. In this case, the random walk selection of addresses was performed only over the low risk nodes, to select 44,574 (50,000 - 4,426) addresses. This was done to reduce the class imbalance issue in the random walk methods.

Finally, it is also important to highlight that for all the above graph sampling techniques, we always made sure that the sampled graphs meet the general edge condition where $E^s - 1 > V^s$.

Table I shows the average results across the 10 samples created for each method for the fraudulent class using features from the node2vec and trans2vec embeddings, both with a dimension of 64. The out of sample prediction results show that the use of the Centrality sampler (CS) outperformed the others, for both node2vec and trans2vec embeddings. Additionally, the results favour the Node2Vec embedding. For the subsequent steps, we proceed therefore with the CS sampled graph.

TABLE I
Outsample prediction results of a Random Forest Classifier for each sampling method and embedding

Method	Embedding	F1	Precision	Recall
DS	*node2vec*	0.451	0.476	0.429
	trans2vec	0.250	0.376	0.188
MS	*node2vec*	0.338	0.350	0.326
	trans2vec	0.229	0.278	0.195
NS	*node2vec*	0.286	0.320	0.261
	trans2vec	0.142	0.209	0.109
CS	**node2vec**	**0.493**	**0.590**	**0.424**
	trans2vec	**0.280**	**0.423**	**0.209**
RWS	*node2vec*	0.080	0.118	0.062
	trans2vec	0.106	0.178	0.076

II. **Embeeded and non-embedded features**

To extract the graph embeddings, we make use of node2vec and trans2vec algorithms. Their aim is to maximise the likelihood of preserving the initial neighbourhood of nodes in a low-dimensional vector space representation.

In comparing the two embedding algorithms, we find that node2vec outperforms trans2vec, despite the richer set of attributes included in the latter one, such as the amount of Bitcoin funds transferred and the time of transactions. This might be due to the fact that trans2vec was originally designed to detect fraudulent addresses in the Ethereum network, which potentially highlights structural differences between the blockchains.

The goal of this step is to extract 64 covariates that will be subsequently used for the classification tasks. Despite the robustness of embedding methods in representing the original networks, they might perform worse when dealing with a severe class imbalance distribution of data. To address this, several non-embedded features were added to the training dataframe.

The non-embedding features that we used were:

- network level - number of self-loops made by each node;
- account level - number of times an address appears as input or output to a transaction in the full network, as well as the difference between the two counts;
- transaction level - descriptive statistics of the amounts and fess paid or received within a transaction for each node.

Self-loops and the total fees paid associated to input addresses proved to be significant predictors, along with other fees-related features.

III. **Classification methods and results**

For the final step, we train the classifier model on four separate algorithms:

- **Logistic Regression** - was chosen because it is a classifier that is easy to implement, interpret and very efficient to train;
- **KNeighborsClassifier** - given the community structures in the network, this method was tuned to find the optimal number of neighbours and the weight assigned to them;
- **Random Forest Classifier** - very fast and easy to implement algorithm that provides accurate and robust results while being relatively less prone to over-fitting;
- **XGBoost Classifier** - a very powerful and flexible algorithm for which the learning rate and maximum depth of the model were tuned to prevent over-fitting on training data.

The performance of the classifiers was assessed based on three scores: F1, precision and recall. Where the precision score captures a profit-maximising perspective, recall reflects a regulation cautious perspective, and F1 balances the previous two.

From the four classifiers, XGBoost achieved the best scores in terms of classification accuracy. These scores are presented in table II.

TABLE II
Out of sample prediction results of best classifiers, XGBoost

	Illicit			Average
Embedding	F1	Precision	Recall	F1
node2vec	0.502	0.513	0.429	0.932
trans2vec	0.332	0.321	0.344	0.906

These results were achieved by standardising the non-embedding features, balancing the data using SMOTE, and tuning the hyperparameters of the classifiers.

IV. **Concluding remarks**

Our findings indicate that methods developed for one cryptocurrency do not necessarily work as well in others. In the case of the Trans2Vec embedder, more tests need to be done to adjust the hyperparameters of this model. In using the standard parameters for the two embedding methods we highlight that for future research further experimentation with these parameters should be done to improve classification results for the fraudulent class. These experiments should be based on the properties of the individual blockchain networks as research has shown that these networks have different properties. We also encourage a study of embedding techniques across different cryptocurrencies.

The results also highlight the difficulty of having good prediction scores within the fraudulent class. This is common throughout the literature. Sadly, this highlights a detachment of these implementations in the literature versus the industry. If a business effectively screened roughly half of fraudulent prospective clients it would be reprimanded by a regulator. With increased regulatory pressure for businesses to perform Know Your Client like checks on crypto users, we expect businesses to collect more fruitful data regarding prospective clients. This data could make a significant difference in the performance of classification models. This was indicated in our research as non-embedding features were deemed to be important to make predictions by classifiers.