Graph-Based Deep Learning for Fraud Detection in ETH Transaction Networks

Gelinas, Stephen
sgelinas@ucsd.edu

Yamamoto, Kazuma
kayamamo@ucsd.edu

Zhou, Ethan
ezhou@uscd.edu

Abstract

Blockchain technology is a quickly growing field, helped greatly through its widespread applications in the financial field. However, the space has been under attack through phishing fraud, posing itself as a major threat to blockchain security. According to the FTC, cryptocurrency scams have cost online users over $1 Billion since 2021; there is a clear demand for the development of effective fraud detection strategies to prevent and detect cybercrimes. With access to Ethereum transaction networks, we can model and train phishing detection as a node classification problem. With graph-based machine learning approaches, we can more effectively flag fraudulent activity in Ethereum networks and reduce the amount of phishing activity with Ethereum, and improve user experience with safer transactions as blockchain technology gains even more traction.

1 Introduction

Blockchain technology allows for the process of recording transactions and tracking assets between two parties without a third party. Recently, many applications have been using blockchain technology, exchanging cryptocurrency such as Bitcoin and Ethereum. However, the amount of phishing scams, money laundering, and other cybercrimes have risen within these blockchain platforms. These crimes are difficult to investigate and identify the offender due to the high frequency of transactions. Traditionally, banks are utilized as the third party or intermediary to constantly monitor and check when a new account is opened or suspicious transactions are made. However, in our case with blockchain, people are easily able to create wallets and make transactions freely, thus creating the concern that it may be difficult to check for suspicious or crime-related transactions. Due to the nature of blockchain technology, transaction information is held public and anyone can attain this information, but this may not be sufficient to detect suspicious activity. As blockchain and cryptocurrency becomes more popular, the number of transactions and wallets increase also, causing detection of suspicious transactions to be more difficult and time-consuming.

Due to the potential cybercrimes that may occur from blockchain, many studies have been conducted to detect suspicious transactions in expansive financial networks. Some of these studies have utilized timestamps, incidental expenses, or the number of transactions as the features for their models. In addition, some studies have also utilized graph embeddings for model optimization and evaluation of fraud detection. However, there are only a few studies out that have been conducted using graph neural network (GNN) models. Recently, a deep learning approach graph convolution algorithm has been popularized to automatically generate features using nodes and edges of graphs. The Graph Convolutional Model (GCN) applies neural network models to graph data, and it learns by updating feature values of each node and edge based on the graph structure. In our study, we will use public Ethereum transaction networks as our graph data, and we will examine GCN in comparison to multiple other graph models to determine the most effective method in detecting phishing fraud transactions. Our source code is available here.

`https://github.com/KazumaYamamoto2023/DSC180B-Q2-Project`

2 Methods

2.1 SVM

Our group first started off with a support vector machine model (SVM) to act as our baseline. This is a supervised machine learning algorithm with the objective to find a hyperplane in an N-dimensional space (N being the of features) that classifies the data points. Because this is a supervised learning approach, we filtered out the unlabelled data and achieved an accuracy of $\sim$48% with this classification model.

2.2 Node2Vec

Node2vec as an algorithm solves the problem of transforming the information inherent within a network into a workable numeric representation by transforming each node into a vector. The first step in this process is done through a random walk algorithm. In our case, the edges between the nodes are given weights corresponding to the transaction amount between accounts. These weights are used to simulate random paths between nodes from the network. In the next step, the skip-gram model works with these paths to learn and creates a node embedding for each node. How this is done is that it reads the random paths taken, and learns which nodes are likely to precede another node. These embeddings then allow us to determine the makings of a fraudulent account. We achieved the best results through the pytorch implementation, with roughly a $\sim$76.6% accuracy on the test set. The pytorch package allows us to make use of the large amount of unsupervised nodes present in our dataset, improving performance.

2.3 GraphSage

GraphSage is most known for its inductive nature which allows it to better adapt to previously unseen nodes. In a setting like Ethereum transaction networks where the network evolves with new nodes and edges every day, there would be a benefit to using a model that does not have to be retrained every time new information is introduced. Additionally, in comparison to methods like node2vec, GraphSage has the advantage of being able to learn from node features. GraphSage works by starting at a node and aggregating information on its neighbors to generate the embeddings. The pytorch package for this model produced an accuracy of $\sim$81.9%, a noticeable upgrade compared to node2vec. This difference can be attributed to the fact that in a largely unsupervised dataset, an inductive approach may outperform the deductive approach.

2.4 GCN

We worked with GCNs in our first quarter on our Clickbait Detection algorithm with TextGCN and we have implemented a version for this task as well. Much of the structure is the same; we create an adjacency matrix based on the edges of the network then propagate steps forwards (3 in our case) then form node embeddings based on these. From there, we can iterate through the nodes and update the node features by hashing the features of each neighboring node. The GCN model from pytorch produced an accuracy of $\sim$79.6%.

2.5 Graph Attention Network

We also tested the performance of Graph Attention Networks (GAT). GATs are a type of graph neural network that leverages masked self-attention which addresses the shortcomings of prior methods based on graph convolutions or their approximations. With GATs, we are able to extract meaningful node features by implicitly specifying different weights got different nodes in a neighborhood, without requiring costly matrix operations nor knowing the underlying graph structure upfront. GAT outperformed GCN and GraphSAGE on node classification tasks on the Cora dataset, so we hypothesized that this model may have similar performance on our dataset compared to other models. The GAT model from pytorch produced an accuracy of $\sim$78.5%, and slightly underperformed compared to GCN and GraphSAGE.

2.6 Adaptive - GCN

The final model we evaluated was Topology Adaptive GCN (TA-GCN). TA-GCN is a graph neural network defined in the vertex domain and outperforms many spectral graph convolutional neural networks (CNNs). TA-GCN utilizes a set of fixed-size learnable filters to perform convolutions on graphs. The topologies of these filters are adaptive to the graph's topology when scanned for convolutions. TA-GCN inherits the properties of convolutions in CNNs for grid structured data, yet doesn't require approximation to compute the convolution, leading to greater performance relative to spectral CNNs. TA-GCN is also computationally simpler than other graph neural networks. Additionally TA-GCN tends to achieve comparable or better accuracy than GCN and GraphSAGE for datasets with larger and sparser graph structures. We hypothesized that the sparsity of our transaction network will be beneficial for TA-GCNs performance in our node classification prediction task. We implemented a TA-GCN model using pytorch, which produced an accuracy of $\sim 82.2\%$, outperforming all of our models for comparison.

3 Appendix

Blockchain technology has been growing in use through its widespread applications in the financial field. However, it has recently attracted increasing cybercrimes with phishing fraud emerging as a major threat to blockchain security. With graph-based machine learning approaches, we can more effectively flag fraudulent activity in Ethereum networks, to reduce the amount of phishing activity with Ethereum, and increase user trust with safe transactions as blockchain technology gains widespread adoption. This problem is similar to our Quarter 1 project in that we are attempting to classify nodes from a network of nodes and edges. However, in Quarter 1 our project required text to be present in the data since we were using TextGCN to classify the nodes. This quarter, our project will not require any text and will rely more heavily on the weights on the edges between nodes in the form of transaction amount. We aim to achieve a high accuracy while working with less information per node-edge than before. There have been some similar analyses in the past, most notably by one of our mentors, Parker Erickson, who has done some analyses into fraud detection using TigerGraph with Graph Attention Networks. However, we propose that we use a richer selection of models and pull more meaningful features from our dataset. We extracted additional node features using GSQL queries by utilizing our dataset's multigraph structure. We computed features of in-degree and out-degree, weighted by the total number of transactions between nodes. Additionally, we computed additional summary statistics such as the sum, mean, minimum and maximum amount of Ethereum sent and received between addresses. We also chose to evaluate the performance of four additional models: Graph Convolution Networks, Node2Vec, GraphSAGE, and Topology Adaptive Graph Convolution Networks. We believe this richer selection of models will lead to better performance in our prediction task, relative to that of Graph Attention Networks, due to the sparsity of our transaction graph's structure. We will evaluate each model by taking the mean testing accuracy in 10 model runs. Our primary output will be a website. Our paper will be successful because we already have access to a dataset with the exact information we need. The dataset contains nodes of accounts and edges of transactions, which is what we are looking for to run node classification.

4 Contributions

Ethan: EDA and adding node features with GSQL queries, baseline SVM model implementation, wrote introduction, wrote methods

Kazuma: Uploading data onto TigerGraph, implementing node2vec, graphsage, semi-supervised GCN, updated Abstract, updated Appendix, made Dockerfile, wrote methods, generated tex file

Stephen: building graph schema, uploading data onto TigerGraph, implementing graph attention network and topology adaptive GCN, hyperparameter tuning, wrote methods, updated appendix