SCHOOL OF COMPUTER AND INFORMATION SCIENCE

Anti Money laundering in Bitcoin

Experimenting with Graph Convelational Network and Fedarated learning on graph

Venkata Pallavi Namburi

20MCME12

December 2023

ABSTRACT

Blockchain provides the unique and accountable channel for financial forensics by mining its open and immutable transaction data. A recent surge has been witnessed by training machine learning models with cryptocurrency transaction data for anomaly detection, such as money laundering and other fraudulent activities.This paper presents a holistic applied data science approach to fraud detection in the Bitcoin network with the help of elliptic dataset which contains 203,769 transaction nodes, 234,355 edges , 167 features per transaction.This project is to develop a federated learning model for predicting whether Bitcoin trans- actions are illicit or licit using a Graph Convolutional Network (GCN). The federated learning approach is employed to enable decentralized training on individual clients' datasets, ensuring pri- vacy and security while collectively improving the global model's accuracy.

WHY GCN FOR THE ELLIPTIC DATASET

Graph Convolutional Networks (GCNs) are particularly well-suited for tasks involving graph-structured data, and they have been shown to be effective in various domains, including social network analysis, biology, and citation networks. However, whether a GCN is the "best" model for predicting elliptic datasets depends on the specific characteristics of the data and the task at hand.

Some reasons

- Graph Structure Representation: Elliptic datasets may have inherent graph structures that capture relationships between data points. GCNs are designed to work with graph-structured data, allowing them to leverage the relationships and dependencies within the data.
- Node-Level and Graph-Level Representations: GCNs can learn both node-level and graph-level representations. This is beneficial for tasks where understanding the local and global structure of the data is important, such as identifying patterns in elliptic datasets.
- Spatial and Spectral Convolution: GCNs perform convolution operations in both spatial and spectral domains on the graph, capturing local and global patterns in the data. This flexibility can be advantageous for tasks where features of interest exhibit spatial dependencies.
- Handling Missing Data and Irregular Graphs:GCNs are capable of handling irregular graph structures and missing data, which can be common challenges in real-world datasets, including elliptic datasets.

WHY NOT OTHER MODELS FOR THE ELLIPTIC DATASET

While Graph Convolutional Networks (GCNs) have demonstrated success in handling graph-structured data, including various applications, it doesn't mean that they are always the best choice for every dataset or prediction task. The choice of a model depends on the characteristics of the data, the nature of the problem, and various other factors.

Some reasons

- Nature of the Data:

If the elliptic dataset does not exhibit clear graph structures or relationships between data points, traditional machine learning models such as support vector machines (SVMs), decision trees, or random forests might be more appropriate.

- Data Size and Complexity:

  For smaller datasets or simpler problems, complex models like GCNs might be unnecessary and prone to overfitting. Simpler models with fewer parameters may generalize better in such cases.

1 WORK DONE

I have gone through 4 research papers to understand the behaviour of the elliptic dataset and GCN model.

1.1 DATASET

The Elliptic Data Set is a sub-graph of the bitcoin graph, made of 203,769 nodes and 234,355 edges. Together with the graph information, it also categorizes the nodes into three classes: "licit", "illicit" or "unkown". A node is deemed "licit" / "illicit" if the corresponding transaction has been created by an entity that belongs to a licit (exchanges, wallet providers, miners, financial service providers, etc.) or illicit (scams, malware, terrorist organizations, ransomware, Ponzi schemes, etc.) category respectively.

The task on the dataset is to classify the illicit and licit nodes, given a set of features and the graph topology. Since not all the nodes are labeled, the problem can be approached in a semi-supervised setting that includes information carried by the unlabeled nodes.

Nodes and edges
Two percent (4,545) of the nodes are labelled class1 (illicit); twenty-one percent (42,019) are labelled class2 (licit). No information is given on the other nodes, which are classified as "unknown".

Features
There are 167 features associated with each node. The temporal information is encoded by a time step running from 1 to 49, a measure of the actual trans-

action time stamp. The time steps are evenly spaced with an interval of about two weeks; each one of them contains a single connected component of transactions that appeared on the blockchain within less than three hours between each other. There are no edges connecting the different time steps. Out of the 167 features, the first 94 represent local information about the transaction — including the time step described above, number of inputs/outputs, transaction fee, output volume and aggregated figures such as average BTC received (spent) by the inputs/outputs and average number of incoming (outgoing) transactions associated with the inputs/outputs. The remaining 72 features represent nonlocal (graph) information in the form of aggregated features, obtained using transaction information one-hop backward/forward from the center node — giving the maximum, minimum, standard deviation and correlation coefficients of the neighbour transactions for the same information data (number of inputs/outputs, transaction fee, etc.).

1.2 Work done related to GCN

- I did the Save and load part. We faced an issue while loading the data after saving the model. after so much research we got to know that the latest version of TensorFlow doesn't support for the predictions to do on the loaded model. The only way is to save the weights of the model and load the weights of the model for the future predictions.
- I have done some predictions using some random transactions from the dataset and output is to print weather they are licit or illicit transactions.

```
1/1 [==============================] - 0s 26ms/step
Sample Predictions:
   Transaction ID Predicted Class
0           26637           licit
1           23946           licit
2           43215           licit
3           20436           licit
4           21404           licit
5           13723           licit
6           34543           licit
7           22166           licit
8           32916         illicit
9           10582           licit
```

Figure 1.1: Taking input as randomly selected 10 transactions from the data to get output

1.3 WORK DONE RELATED TO FEDERATED LEARNING

Federated learning workflow

- Edge devices receives a copy of a global model from a central server.
- The model is being trained locally on the data residing on the edge devices.
- The global model weights are updated during training on each worker.
- A local copy is sent back to the central server.
- The server receives various updated model and aggregate the updates thereby improving the global model and also preserving privacy of data in which it was being trained on.

I have developed a Federated Learning environment using the FLOWER framework. I have taken a model from tensorflow keras (open source neural networks library) and implemented the same in a federated learning setup. The MNIST dataset has been used to train this model. The task selected is Image Classification. Federated system has to have minimum of 2 clients. These clients train local models on their local machines non-IID subsets of the MNIST dataset. (Non-IID data distribution implies that the clients have different subsets of the data, ensuring a more realistic and challenging Federated Learning scenario) . Then send them back to the server. Upon receiving all the models from the clients , server coordinates the training process by aggregating model updates and broadcasting the global model updates.

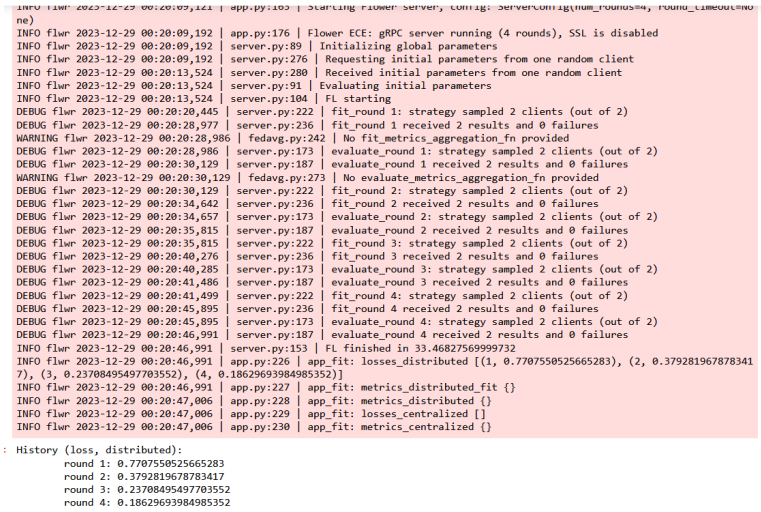

Figure 1.2: server output

```
Fit History :
 {'loss': [0.2025355100631714], 'accuracy': [0.9427821636199951], 'val_loss': [2.608057975769043], 'val_accuracy': [0.497500
0023841858]}


GLOBAL Model Evaluation accuracy :  0.8476999998092651


Fit History :
 {'loss': [0.11063624173402786], 'accuracy': [0.9685039520263672], 'val_loss': [1.7765408754348755], 'val_accuracy': [0.5569
000244140625]}


GLOBAL Model Evaluation accuracy :  0.8867999911308289


Fit History :
 {'loss': [0.075461745262146], 'accuracy': [0.977637767791748], 'val_loss': [1.1508959531784058], 'val_accuracy': [0.6951000
094413757]}


GLOBAL Model Evaluation accuracy :  0.9280999898910522


Fit History :
 {'loss': [0.05905267596244812], 'accuracy': [0.9810498952865601], 'val_loss': [1.3368622064590454], 'val_accuracy': [0.6704
000234603882]}

DEBUG flwr 2023-12-29 00:20:47,101 | connection.py:141 | gRPC channel closed
INFO flwr 2023-12-29 00:20:47,101 | app.py:304 | Disconnect and shut down

GLOBAL Model Evaluation accuracy :  0.9423999786376953
```

Figure 1.3: client1 output

```
Fit History :
 {'loss': [0.21164193749427795], 'accuracy': [0.9343719482421875], 'val_loss': [2.883103132247925], 'val_accuracy': [0.47330
00099658966]}


GLOBAL Model Evaluation accuracy :  0.8476999998092651


Fit History :
 {'loss': [0.10224951803684235], 'accuracy': [0.9688413143157959], 'val_loss': [2.3707377910614014], 'val_accuracy': [0.5261
99996471405]}


GLOBAL Model Evaluation accuracy :  0.8867999911308289


Fit History :
 {'loss': [0.07140635699033737], 'accuracy': [0.9767283201217651], 'val_loss': [2.1168456077575684], 'val_accuracy': [0.5985
999703407288]}


GLOBAL Model Evaluation accuracy :  0.9280999898910522


Fit History :
 {'loss': [0.05616910383105278], 'accuracy': [0.9814994931221008], 'val_loss': [1.9067076444625854], 'val_accuracy': [0.6593
000292778015]}

DEBUG flwr 2023-12-29 00:20:47,069 | connection.py:141 | gRPC channel closed
INFO flwr 2023-12-29 00:20:47,084 | app.py:304 | Disconnect and shut down

GLOBAL Model Evaluation accuracy :  0.9423999786376953
```

Figure 1.4: client2 output

Federated learning using elliptic dataset with GCN model

Taking example as the above model we have implemented as the same using GCN model. we are getting parameters but the model was not fitting into the frame work. Because as i mentioned earlier the latest tenserflow version was not allowing to create pickle file so, we are unable to fit the model into the frame work.

2 References

Here is the link to the GitHub repository that contains my codebase :
https://github.com/venkataPallavi/Crop_recommendation
Data Set
https://www.kaggle.com/datasets/ellipticco/elliptic-data-set/data