Graph Learning in Financial Networks

Collaboration with Tianyu Du, Fan-yun Sun, Jure Leskovec

Jiaxuan You
Stanford University
Financial Networks

- **Financial Networks**: Describe financial entities and their connections

**International banking**
- **Nodes**: Countries
- **Edges**: Capital flows

**Bitcoin transactions**
- **Nodes**: BTC wallets
- **Edges**: Transactions

Image credit: The Political Economy of Global Finance: A Network Model

Image credit: https://dailyblockchain.github.io/
Graph Learning in Financial Networks

- **Goal**: A graph learning framework for financial networks
  - **Applications**: Fraud detection, Anti-money laundering, Anomaly detection
  - **Solution**: Graph representation learning!

**Input**: Financial networks

**Output**: Predictions

- **Node-level**: Fraudsters, ...
- **Subgraph-level**: Money laundering subnetworks, ...
- **Edge-level**: Fraudulent/anomalous transactions, ...
Why Graph Representation?

- **Transaction-based approach**
  - “On 01/03, Client $A$ sends Company $B$ $500$”
  - Build models based on transaction attributes
  - **Issues**: ignore the context of a transaction

- **Graph-based approach**
  - Represent transactions as a **dynamic graph**
  - Predictions are made based on the entire graph
  - **Benefits**:
    - Represents a transaction with a broader context
    - Requires fewer feature-engineering
Overview of This Talk

ROLAND: GNN for Financial Networks

Results & Insights from ROLAND

Application: Anomaly Detection

client  | bank  | company

NN NN NN NN NN
ROLAND: GNN for Financial Networks

- **ROLAND framework:**
  - Transform financial networks as GNN computational graphs
  - Learning from diverse signals

---

### Self-supervised (from raw data)
- Will there be a transaction? **Yes**
- What is the amount? **$500**
- When will it happen? **01/03**

### Supervised (from external sources)
- Is it a fraud? **No**
- Is it money laundering? **Yes**

---

**Jiaxuan You, Stanford University**

*ROLAND: Graph Neural Networks for Dynamic Graphs*, in submission
ROLAND Model: From Static to Dynamic GNN

**Question:** Can we adapt a SOTA static GNN to dynamic prediction tasks?
ROLAND Model: From Static to Dynamic GNN

- **Idea:** *Recursively* update node embeddings *at each layer*
- Introduce a new module to a static GNN:
  - **Input:**
    - Previous embeddings from the same layer
    - Current embeddings from the previous layer
  - **Output:** Updated embeddings

- **Benefits:**
  - Simple and effective
  - Benefit from the SOTA designs of a static GNN

**Static GNN**
- Graph $G$
- GNN Layer 1
- GNN Layer 2
- Pred $y$

**Dynamic GNN**
- Pred $y_t$
- GNN Layer 2
- Embedding update
- GNN Layer 1
- Embedding update
- Snapshot $G_{t-1}$
- Pred $y_{t+1}$
- GNN Layer 2
- Embedding update
- GNN Layer 1
- Embedding update
- Snapshot $G_t$
**ROLAND Training: Efficient Training**

- **Incremental training:**
  - Only keep these in GPU
    - GNN Model $GNN_t$
    - Historical node states $H_{t-1}$
    - Incoming new graph snapshot $G_t$
    - Efficient and work well in practice

- **Meta training:**
  - Train a meta-GNN that can quickly adapt to new data
  - Benefits: ROLAND does not need to be frequently retrained
Overview of This Talk

ROLAND: GNN for Financial Networks

Results & Insights from ROLAND

Application: Anomaly Detection
**ROLAND Implementation: GraphGym**

- **GraphGym** offers a standardized pipeline for graph learning
  - Design Space of Graph Neural Networks (NeurIPS 2020 spotlight)
  - **Now officially part of PyG 2.0! (Stay tuned for my next talk :)**

- We implement the financial prediction pipeline with GraphGym
  - We adopt many optimal design choices from GraphGym
An Example Financial Dataset

- **Transactions between companies** captured by a bank clearing system
  - 336 million transactions, 6 million accounts, 6 years of daily data
Task 1: Classify Fraud Transactions

- **Task setting:** Supervised learning
  - *Unbalanced labels:* 2% of all the transactions are fraudulent
  - *Random dataset split:* 80% training, 10% validation, 10% testing
  - *Metric:* AUC on test set (random guess has AUC 0.5, higher is better)

- **Best non-deep learning baseline**
  - AUC: ~0.8

- **Simple GNN implementation using out-of-the-box PyG**
  - AUC: 0.81

- **ROLAND framework works much better!**
  - AUC: 0.94
Task 2: Forecast Future Transactions

- **Task setting**: Self-supervised/Unsupervised learning
  - **Rolling prediction**: On each day, use all the historical information to predict the transactions on the next day
  - **Metric**: Mean reciprocal ranking (MMR) of ground-truth future transactions

<table>
<thead>
<tr>
<th></th>
<th>Bitcoin-OTC</th>
<th>Bitcoin-Alpha</th>
<th>UCI-Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN [14]</td>
<td>0.0025</td>
<td>0.0031</td>
<td>0.1141</td>
</tr>
<tr>
<td>DynGEM [12]</td>
<td>0.0921</td>
<td>0.1287</td>
<td>0.1055</td>
</tr>
<tr>
<td>dyngraph2vecAE [8]</td>
<td>0.0916</td>
<td>0.1478</td>
<td>0.0540</td>
</tr>
<tr>
<td>dyngraph2vecAERNN [8]</td>
<td><strong>0.1268</strong></td>
<td><strong>0.1945</strong></td>
<td><strong>0.0713</strong></td>
</tr>
<tr>
<td>EvolveGCN-H [30]</td>
<td>0.0690</td>
<td>0.1104</td>
<td>0.0899</td>
</tr>
<tr>
<td>EvolveGCN-O [30]</td>
<td>0.0968</td>
<td>0.1185</td>
<td>0.1379</td>
</tr>
<tr>
<td>ROLAND Moving Average</td>
<td>0.0468 ± 0.0022</td>
<td>0.1399 ± 0.0107</td>
<td>0.0649 ± 0.0049</td>
</tr>
<tr>
<td>ROLAND MLP</td>
<td>0.0778 ± 0.0024</td>
<td>0.1561 ± 0.0114</td>
<td>0.0875 ± 0.0110</td>
</tr>
<tr>
<td>ROLAND GRU</td>
<td><strong>0.2203 ± 0.0167</strong></td>
<td><strong>0.2885 ± 0.0123</strong></td>
<td><strong>0.2289 ± 0.0618</strong></td>
</tr>
<tr>
<td>Improvement over best baseline</td>
<td>73.74%</td>
<td>43.33%</td>
<td>65.99%</td>
</tr>
</tbody>
</table>
Analysis: ROLAND’s Performance over Time

ROLAND maintains a robust performance over transaction pattern changes.
Question: What kind of transactions the model paid attention to?

Method: Inspect edges' attention scores of the last GNN layer.
- Select edges with low/high attention scores.
- Get the feature value distribution of these edges

Higher attention to large amount transactions
Overview of This Talk

ROLAND: GNN for Financial Networks

Results & Insights from ROLAND

Application: Anomaly Detection

Graph Neural Networks

client bank company
**Goal:** Detect anomalous transactions **without external supervision**

- Combine Graph Representation Learning with Outlier Detection

**Input:** Financial networks

**Application: Anomaly Detection**

- **node embeddings**
- **edge embeddings**

Outlier detection

- learned decision function
- true inliers
- true outliers
**Self-supervised objectives**

“Indicators and patterns of money laundering or terrorist financing”

<table>
<thead>
<tr>
<th>Real world Indicators (from <a href="http://baselgovernance.org">baselgovernance.org</a>)</th>
<th>Objectives for GNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>investment funds sent to countries of concern</td>
<td>Predict country</td>
</tr>
<tr>
<td>Large cash deposits into (company) accounts</td>
<td>Predict amount</td>
</tr>
<tr>
<td>unusual subject of transaction</td>
<td>Predict subject of transaction (link prediction)</td>
</tr>
<tr>
<td>high volume of transactions within a short period</td>
<td>predict accumulated amount</td>
</tr>
<tr>
<td>multiple individuals sending funds to the one beneficiary</td>
<td>predict # transactions</td>
</tr>
</tbody>
</table>
Objectives for Anomaly Detection

- **Solution:** ROLAND framework + diverse objectives

**Self-supervised Objectives**
- Objective 1: predict amount
- Objective 2: predict recipient
- Objective 3: predict country
- ...

**Outlier Detection Objectives**

\[
\min_{w,V,r} \frac{1}{2} \|w\|^2 + \frac{1}{2} \|V\|^2_F + \frac{1}{\nu} \cdot \frac{1}{N} \sum_{n=1}^{N} \max(0, r - \langle w, g(VX_n) \rangle) - r
\]
Quantitative Evaluation for Anomaly Detection

- **Task:** Detect anomalous transactions **without supervision**
- **Goal:** Simulate real-world anomaly detection use cases
  - Corrupt the dataset to see if ROLAND can extract them
    - Report *precision/recall* of corrupted edges detected by ROLAND
  - Ways of corrupting the dataset:
    - **Amount corruption** (~1% transactions)
    - **Country corruption** (~1% transactions)
    - **Recipient corruption** (~1% transactions) → Rewire!
## Quantitative Evaluation for Anomaly Detection

### Deep learning baseline:

<table>
<thead>
<tr>
<th>Amount corruption (~1%)</th>
<th>Naive Baseline</th>
<th>Link Prediction + One Class SVM</th>
<th>Multi-head Self-supervised + One Class SVM</th>
<th>Multi-head Self-supervised End to end</th>
</tr>
</thead>
<tbody>
<tr>
<td>- 20% corruption by x2</td>
<td>0.34</td>
<td>1.06</td>
<td>0.94</td>
<td>1.32</td>
</tr>
<tr>
<td>- 20% corruption by x5</td>
<td>0.47</td>
<td>1.00</td>
<td>0.95</td>
<td>1.09</td>
</tr>
<tr>
<td>- 20% corruption by x10</td>
<td>0.52</td>
<td>1.03</td>
<td>1.05</td>
<td>1.13</td>
</tr>
<tr>
<td>- 20% corruption by x100</td>
<td>0.37</td>
<td>1.07</td>
<td>1.05</td>
<td>1.09</td>
</tr>
<tr>
<td>- 20% corruption by x1000</td>
<td>0.32</td>
<td>1.10</td>
<td>1.09</td>
<td>1.12</td>
</tr>
</tbody>
</table>

### Rule-based baseline:

<table>
<thead>
<tr>
<th>Amount corruption (~1%)</th>
<th>Naive Baseline</th>
<th>Link Prediction + One Class SVM</th>
<th>Multi-head Self-supervised + One Class SVM</th>
<th>Multi-head Self-supervised End to end</th>
</tr>
</thead>
<tbody>
<tr>
<td>- 20% corruption by x2</td>
<td>1.97</td>
<td>4.71</td>
<td>4.74</td>
<td>5.24</td>
</tr>
<tr>
<td>- 20% corruption by x5</td>
<td>0.64</td>
<td>2.27</td>
<td>2.41</td>
<td>2.65</td>
</tr>
<tr>
<td>- 20% corruption by x10</td>
<td>0.63</td>
<td>1.37</td>
<td>1.55</td>
<td>1.65</td>
</tr>
<tr>
<td>- 20% corruption by x100</td>
<td>3.15</td>
<td>7.48</td>
<td>7.92</td>
<td>8.58</td>
</tr>
</tbody>
</table>

### +14.7% Precision

### +14.2% Recall
Normal accounts
Potential anomalous accounts
Overview of This Talk

ROLAND: GNN for Financial Networks

Results & Insights from ROLAND

Application: Anomaly Detection

Resources: ROLAND (in submission)
Design Space for Graph Neural Networks
Learn more about GNN: PyG.org