Alleviating the Inconsistency Problem of Applying Graph Neural Network to Fraud Detection

SIGIR’20, Virtual Event, China

Zhiwei Liu, Yingtong Dou, Philip S. Yu (University of Illinois at Chicago)
Yutong Deng (Beijing University of Posts and Telecommunications)
Hao Peng (Beihang University)

Code: https://github.com/safe-graph/DG Fraud.git
Background
Graph Neural Network (GNN)

GCN\(^{[1]}\)

- Directly aggregate neighbors using Laplacian adjacency matrix

GraphSAGE\(^{[2]}\)

- Sample and aggregate neighbors

GAT\(^{[3]}\)

- Attentively aggregate neighbors

---

GNN-based Fraud Detectors

FdGars\(^1\) (GCN-based)  
GAS\(^2\) (GAT-based)  
Player2Vec\(^3\) (Hybrid)

\[^3\] Zhang, Y., Fan, Y., Ye, Y., Zhao, L. and Shi, C., 2019, November. Key Player Identification in Underground Forums over Attributed Heterogeneous Information Network Embedding Framework. CIKM 2019
Motivation
Motivation

• Assumption of GNN: neighbors share similar features, context, and labels (smoothness\textsuperscript{(1)})

• This assumption can no longer hold in fraud detection task, i.e., inconsistency problem

Motivation

- **Context inconsistency**: $v_1$ staying with three begin nodes (4,6,7)
- **Feature inconsistency**: $v_1$ having features of great difference to others
- **Relation inconsistency**: under relation I (solid), $v_1$ only connecting to other fraudsters; while under relation II (dash), $v_1$ connecting to three benign nodes.

Direct aggregation results in the loss of information! We should design a new GNN model to handle these inconsistencies.
Proposed Model (GraphConsis)
GraphConsis

• Context Embedding: trainable context embedding
• Embedding consistency measurement
  • Ignoring inconsistent neighbors
  • Generating consistent sampling probabilities
• Relation Attention: specially dealing with various relations
GraphConsis

• General GNN structure

\[ h^{(l)}_{v} = h^{(l-1)}_{v} \oplus AGG^{(l)} \left( \left\{ h^{(l-1)}_{v'} : v' \in \mathcal{N}_{v} \right\} \right) \]
GraphConsis

- Context Embedding
  \[ h^{(1)}_{\nu} = \{x_{\nu} \| c_{\nu} \} \oplus \text{AGG}^{(1)} \left( \{x_{\nu'} \| c_{\nu'} : \nu' \in \mathcal{N}_\nu \} \right) \]

- Sampling Probability
  \[ p^{(l)}(u; \nu) = s^{(l)}(u, \nu) / \sum_{u \in \mathcal{N}_\nu} s^{(l)}(u, \nu) \]

- Relation Attention
  \[ \text{AGG}^{(l)} \left( \left\{ h^{(l-1)}_q \right\}_{q=1}^Q \right) = \sum_{q=1}^Q \alpha_q^{(l)} h_q^{(l)} \]
  \[ \alpha_q^{(l)} = \exp \left( \sigma \left( \{h_q^{(l)} \| t_{rq} \} a^T \right) \right) / \sum_{q=1}^Q \exp \left( \sigma \left( \{h_q^{(l)} \| t_{rq} \} a^T \right) \right) \]
  \[ t_{rq} : \text{the relation vector of relation } r \text{ (of } q \text{ sample); } a^T : \text{the weight of attention layer} \]

Q: # of samples; \( \alpha_q \) : a scalar denoting the attention weights of q-th sample
Experiment
Inconsistency problem

Context Characteristic Score (Label smoothness)

\[ y_r^{(c)} = \sum_{(u, v) \in E_r} \frac{1 - \mathbb{1}(u \sim v)}{|E_r|} \]

Indication of whether node \( u, v \) have the same label

Feature Characteristic Score (Feature smoothness)

\[ y_r^{(f)} = \sum_{(u, v) \in E_r} \exp \left( -\frac{\|x_u - x_v\|_2^2}{|E_r| \cdot d} \right) \]

Table 1: The statistics of different graphs.

<table>
<thead>
<tr>
<th>Graph</th>
<th>#Nodes</th>
<th>#Edges</th>
<th>( y^{(f)} )</th>
<th>( y^{(c)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Others</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cora</td>
<td>2,708</td>
<td>5,278</td>
<td>0.72</td>
<td>0.81</td>
</tr>
<tr>
<td>PPI</td>
<td>14,755</td>
<td>225,270</td>
<td>0.48</td>
<td>0.98</td>
</tr>
<tr>
<td>Reddit</td>
<td>232,965</td>
<td>11,606,919</td>
<td>0.70</td>
<td>0.63</td>
</tr>
<tr>
<td>Ours</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R-U-R</td>
<td>45,954</td>
<td>98,630</td>
<td>0.83</td>
<td>0.90</td>
</tr>
<tr>
<td>R-T-R</td>
<td>45,954</td>
<td>1,147,232</td>
<td>0.79</td>
<td>0.05</td>
</tr>
<tr>
<td>R-S-R</td>
<td>45,954</td>
<td>6,805,486</td>
<td>0.77</td>
<td>0.05</td>
</tr>
<tr>
<td>Yelp-ALL</td>
<td>45,954</td>
<td>7,693,958</td>
<td>0.77</td>
<td>0.07</td>
</tr>
</tbody>
</table>

Yelp data
29431 users, 182 products, and 45954 reviews
Overall Comparison

**Observations**

- LR is better than other GNNs
- GraphConsis performs better than other baselines
- Increasing training data improves GraphConsis a lot

<table>
<thead>
<tr>
<th>Method</th>
<th>40% (F1)</th>
<th>40% (AUC)</th>
<th>60% (F1)</th>
<th>60% (AUC)</th>
<th>80% (F1)</th>
<th>80% (AUC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>0.4647</td>
<td><strong>0.6140</strong></td>
<td>0.4640</td>
<td>0.6239</td>
<td>0.4644</td>
<td>0.6746</td>
</tr>
<tr>
<td>GraphSAGE</td>
<td>0.4956</td>
<td>0.5081</td>
<td>0.5127</td>
<td>0.5165</td>
<td>0.5158</td>
<td>0.5169</td>
</tr>
<tr>
<td>FdGars</td>
<td>0.4603</td>
<td>0.5505</td>
<td>0.4600</td>
<td>0.5468</td>
<td>0.4603</td>
<td>0.5470</td>
</tr>
<tr>
<td>Player2Vec</td>
<td>0.4608</td>
<td>0.5426</td>
<td>0.4608</td>
<td>0.5697</td>
<td>0.4608</td>
<td>0.5403</td>
</tr>
<tr>
<td>GraphConsis</td>
<td><strong>0.5656</strong></td>
<td>0.5911</td>
<td><strong>0.5888</strong></td>
<td><strong>0.6613</strong></td>
<td><strong>0.5776</strong></td>
<td><strong>0.7428</strong></td>
</tr>
</tbody>
</table>
Implementations

• Code: https://github.com/safe-graph/DGFraud.git
• We also reproduced some GNN-based fraud detector

A Deep Graph-based Toolbox for Fraud Detection

**DG Fraud** is a Graph Neural Network (GNN) based toolbox for fraud detection. It integrates the implementation & comparison of state-of-the-art GNN-based fraud detection models. It also includes several utility functions such as graph preprocessing, graph sampling, and performance evaluation. The introduction of implemented models can be found here.

We welcome contributions on adding new fraud detectors and extending the features of the toolbox. Some of the planned features are listed in TODO list.

If you use the toolbox in your project, please cite the paper below and the algorithms you used:

**Implemented Models**

<table>
<thead>
<tr>
<th>Model</th>
<th>Paper</th>
<th>Venue</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>SemiGNN</td>
<td>A Semi-supervised Graph Attentive Network for Financial Fraud Detection</td>
<td>ICDM 2019</td>
<td>BibTex</td>
</tr>
<tr>
<td>Player2Vec</td>
<td>Key Player Identification in Underground Forums over Attributed Heterogeneous Information Network Embedding Framework</td>
<td>CIKM 2019</td>
<td>BibTex</td>
</tr>
<tr>
<td>GAS</td>
<td>Spam Review Detection with Graph Convolutional Networks</td>
<td>CIKM 2019</td>
<td>BibTex</td>
</tr>
<tr>
<td>FdGars</td>
<td>FdGars: Fraudster Detection via Graph Convolutional Networks in Online App Review System</td>
<td>WWW 2019</td>
<td>BibTex</td>
</tr>
<tr>
<td>GeniePath</td>
<td>GeniePath: Graph Neural Networks with Adaptive Receptive Paths</td>
<td>AAAI 2019</td>
<td>BibTex</td>
</tr>
<tr>
<td>GEM</td>
<td>Heterogeneous Graph Neural Networks for Malicious Account Detection</td>
<td>CIKM 2018</td>
<td>BibTex</td>
</tr>
</tbody>
</table>
Discussion
Conclusion and Future Work

• Conclusion
  • Investigate three inconsistencies (context, feature, and relation)
  • Design three mechanisms in GraphConsis

• Future work
  • General inconsistencies?
  • Adaptive sampling?
  • Other consistency metrics?
Thanks!