LOCALLY PRIVATE GRAPH NEURAL NETWORKS

Sina Sajadmanesh
Daniel Gatica-Perez

AI4Media Workshop on Explainability, Robustness and Privacy in AI
June 2, 2021
Graph learning with node data privacy

Setting:
- Graph topology is public to the server
- Node data (features and possibly labels) are private to nodes

Problem:
- How to learn a GNN without exposing private node data?
Local Differential Privacy

- Every data holder perturbs their data using a randomized mechanism
- The aggregator collects and aggregates perturbed data to estimate the target statistics

**Definition**

A randomized mechanism $M$ satisfies $\epsilon$-LDP if for all pairs of private data $x_1$ and $x_2$, and for all outputs $x'$ of $M$, we have:

$$\Pr[M(x_1) = x'] \leq e^\epsilon \Pr[M(x_2) = x']$$

Image Credit: Bennett Cyphers
**Local Differential Privacy**

- Every data holder perturbs their data using a **randomized mechanism**
- The aggregator collects and **aggregates** perturbed data to **estimate** the target statistics

**Definition**

A randomized mechanism $\mathcal{M}$ satisfies $\epsilon$-LDP if for all pairs of private data $x_1$ and $x_2$, and for all outputs $x'$ of $\mathcal{M}$, we have:

$$\Pr[\mathcal{M}(x_1) = x'] \leq e^\epsilon \Pr[\mathcal{M}(x_2) = x']$$
GNNs are **message-passing neural networks**

**AGGREGATE**: nodes aggregate their neighbors’ representation vector

**UPDATE**: a neural network generates new node representation from aggregated vectors

Private neighborhood aggregation with LDP

- Node features are perturbed by injecting noise
- The neighborhood aggregation cancels out the noise
**WHY LOCAL DP?**

GNNs are *message-passing* neural networks

**AGGREGATE:** nodes aggregate their neighbors’ representation vector
**UPDATE:** a neural network generates new node representation from aggregated vectors

Private neighborhood aggregation with LDP
- Node features are perturbed by *injecting noise*
- The neighborhood aggregation *cancels out* the noise
High-dimensional features

- The total privacy budget of a node scales with the number of features
  - Keeping the total privacy budget small → Too much noise!
CHALLENGES

High-dimensional features

- The total privacy budget of a node scales with the number of features
  - Keeping the total privacy budget small → Too much noise!

Our solution: Multi-bit mechanism for multidimensional perturbation

- Multi-bit Encoder: perturb a random subset of node features and compress the output
- Multi-bit Rectifier: uncompress and de-bias encoded features
Small neighborhoods

- Lots of the nodes have **too few neighbors**
  - Noise won’t cancel out if the neighborhood size is small
Small neighborhoods

- Lots of the nodes have too few neighbors
  - Noise won’t cancel out if the neighborhood size is small

Our solution: KProp linear convolution

- Expands the neighborhood to the nodes that are up to K-hops away
- Applies $K$ consecutive AGGREGATE
- Can be prepended to any GNN architecture as a feature denoising mechanism
User-Side:
1. Perturb node features using MB encoder
2. Send encoded features to server

Server-Side:
3. De-bias encoded features with MB rectifier
4. De-noise rectifier’s output using KProp
5. Train GNN on denoised features
Randomized Response for label differential privacy

- True label $y$
- Perturbed label $y'$
- Number of classes $c$
- DP privacy budget $\epsilon$

\[
p(y' \mid y) = \begin{cases} 
\frac{e^{\epsilon}}{e^{\epsilon} + c - 1}, & \text{if } y' = y \\
\frac{1}{e^{\epsilon} + c - 1}, & \text{otherwise}
\end{cases}
\]
Trivial method: directly train GNN with noisy labels

- GNN severely overfits the noisy labels
- Poor generalization performance
Trivial method: directly train GNN with noisy labels

- GNN severely overfits the noisy labels
- Poor generalization performance

**Key Idea:** use KProp to denoise labels!

- Apply KProp on one-hot encoded noisy labels
- Pick the label with highest value
Effect of KProp on label accuracy

- Accuracy between true label $y$ and recovered label $\tilde{y}$

Facebook (4 classes)

Cora (7 classes)

LastFM (10 classes)
Effect of KProp on label accuracy

- Accuracy between true label $y$ and recovered label $\tilde{y}$

How to find best performing $K$ without clean validation data?
Prevent absorbing noise in $\tilde{y}$

- $y$ is perturbed by RR and is given KProp to get $\tilde{y}$
- Apply the same process on $\hat{p}(y \mid x)$ to obtain $\hat{p}(\tilde{y} \mid x)$
- Train $\hat{p}(\tilde{y} \mid x)$ with $\tilde{y}$
LABEL DENOISING WITH PROPAGATION

Prevent absorbing noise in $\tilde{y}$

- $y$ is perturbed by RR and is given KProp to get $\tilde{y}$
- Apply the same process on $\hat{p}(y \mid x)$ to obtain $\hat{p}(\tilde{y} \mid x)$
- Train $\hat{p}(\tilde{y} \mid x)$ with $\tilde{y}$

Prevent absorbing noise in $y'$

- RR gives an upperbound on label accuracy:
  \[
  Acc^* = p(y' = y) = \frac{e^\epsilon}{e^\epsilon + c - 1}
  \]
- Stop training when GNN's accuracy on $y'$ goes beyond $Acc^*$
Experimental Results

LPNNG’s performance under varying feature and label privacy budgets

- Base GNN: GraphSAGE

Facebook

LastFM
Comparison of base GNN architectures

Dataset: Facebook
**Experimental Results**

Comparison of different LDP mechanisms

- Base GNN: GraphSAGE
- $\epsilon_y = \infty$

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Mechanism</th>
<th>$\epsilon_x = 0.01$</th>
<th>$\epsilon_x = 0.1$</th>
<th>$\epsilon_x = 1$</th>
<th>$\epsilon_x = 2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORA</td>
<td>1B</td>
<td>45.8 ± 3.3</td>
<td>62.3 ± 1.5</td>
<td>59.9 ± 2.7</td>
<td>58.5 ± 2.9</td>
</tr>
<tr>
<td></td>
<td>LP</td>
<td>43.2 ± 3.1</td>
<td>57.8 ± 2.3</td>
<td>61.9 ± 3.1</td>
<td>58.1 ± 2.1</td>
</tr>
<tr>
<td></td>
<td>AG</td>
<td>59.7 ± 2.3</td>
<td>62.7 ± 2.8</td>
<td>67.5 ± 3.0</td>
<td>77.2 ± 1.9</td>
</tr>
<tr>
<td></td>
<td>MB</td>
<td><strong>68.0 ± 2.9</strong></td>
<td><strong>64.6 ± 3.2</strong></td>
<td><strong>83.9 ± 0.4</strong></td>
<td><strong>84.0 ± 0.3</strong></td>
</tr>
<tr>
<td>FACEBOOK</td>
<td>1B</td>
<td>57.0 ± 3.4</td>
<td>76.3 ± 1.6</td>
<td>86.1 ± 0.6</td>
<td>84.0 ± 1.3</td>
</tr>
<tr>
<td></td>
<td>LP</td>
<td>54.2 ± 2.9</td>
<td>72.5 ± 2.1</td>
<td>85.4 ± 0.4</td>
<td>84.8 ± 1.6</td>
</tr>
<tr>
<td></td>
<td>AG</td>
<td>78.2 ± 1.4</td>
<td>85.6 ± 0.7</td>
<td>92.0 ± 0.1</td>
<td>92.4 ± 0.2</td>
</tr>
<tr>
<td></td>
<td>MB</td>
<td><strong>85.8 ± 0.4</strong></td>
<td><strong>91.0 ± 0.4</strong></td>
<td><strong>92.7 ± 0.1</strong></td>
<td><strong>92.9 ± 0.1</strong></td>
</tr>
</tbody>
</table>
Experimental Results

Comparison of different learning algorithms

- Base GNN: GraphSAGE
- $\epsilon_x = 1$

<table>
<thead>
<tr>
<th>DATASET</th>
<th>$\epsilon_y$</th>
<th>CROSS ENTROPY</th>
<th>FORWARD CORRECTION</th>
<th>DROP</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORA</td>
<td>0.5</td>
<td>18.6 ± 1.3</td>
<td>18.6 ± 2.5</td>
<td>42.9 ± 1.5</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>25.5 ± 1.7</td>
<td>37.1 ± 2.5</td>
<td>69.3 ± 1.2</td>
</tr>
<tr>
<td></td>
<td>2.0</td>
<td>52.9 ± 2.1</td>
<td>75.1 ± 1.0</td>
<td>78.4 ± 0.7</td>
</tr>
<tr>
<td>FACEBOOK</td>
<td>0.5</td>
<td>50.9 ± 4.2</td>
<td>68.9 ± 1.3</td>
<td>75.1 ± 0.6</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>55.2 ± 1.3</td>
<td>73.8 ± 1.1</td>
<td>84.9 ± 0.2</td>
</tr>
<tr>
<td></td>
<td>2.0</td>
<td>81.6 ± 1.2</td>
<td>88.9 ± 0.2</td>
<td>90.7 ± 0.1</td>
</tr>
<tr>
<td>LASTFM</td>
<td>0.5</td>
<td>21.1 ± 4.6</td>
<td>44.9 ± 5.3</td>
<td>70.0 ± 3.0</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>28.4 ± 2.5</td>
<td>58.5 ± 3.6</td>
<td>82.1 ± 1.0</td>
</tr>
<tr>
<td></td>
<td>2.0</td>
<td>56.8 ± 2.8</td>
<td>79.2 ± 1.3</td>
<td>85.7 ± 0.7</td>
</tr>
</tbody>
</table>
CONCLUSION

Summary

▶ Proposed a privacy-preserving GNN based on local differential privacy
  • Multi-bit mechanism for high-dimensional feature perturbation
  • KProp for feature and label denoising
  • Drop algorithm for learning with noisy labels

▶ Demonstrated promising results in terms of accuracy-privacy trade-off

Future Work

▶ Protect privacy of graph topology
THANK YOU!

@sajadmanesh
sajadmanesh@idiap.ch