Privacy-Preserving Deep Learning Over Graphs

Sina Sajadmanesh
sajadmanesh@idiap.ch

Idiap Research Institute
Swiss Federal Institute of Technology (EPFL)

Information Processing and Communications Lab
Imperial College London
Dec 9, 2020

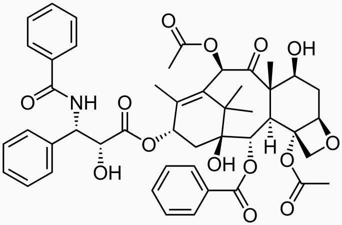
Outline

Introduction and Motivation

Graphs are ubiquitous

Social Networks

Molecules



Knowledge Graphs

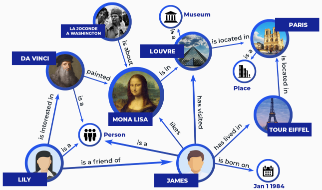


Image source (from left to right): https://yashuseth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kgqa/, https://en.wikipedia.org/wiki/Terpenoid, https://yashuseth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kgqa/

Node Classification / Regression

- Given a graph, which is the class label / value of a node?
- **Example:** face account detection

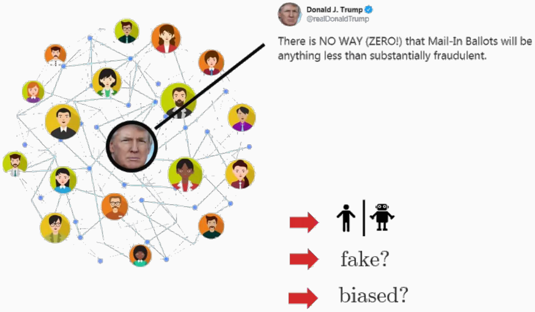
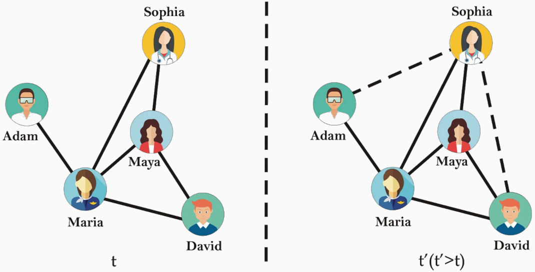


Image Source: "A gentle introduction to graph neural networks" by Andreas Loukas.

Link Prediction

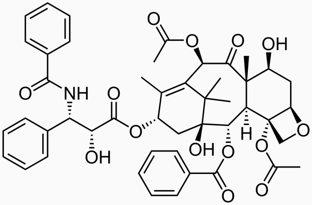
- Given a graph, which links are likely to form?
- **Example:** recommendation systems



[Ahmad et al., 2020]

Graph Classification

- Given a graph, predict its label
- **Example:** antibiotic discovery



Antibiotic? Or Not

Image source: https://en.wikipedia.org/wiki/Terpenoid

Graph Decoding to Structured Data

- Given a graph, what is the corresponding structures representation?
- **Example:** graph to text, graph to image, . . .

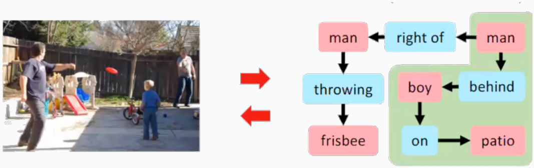
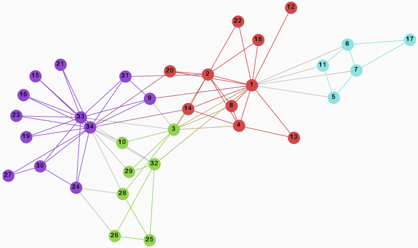


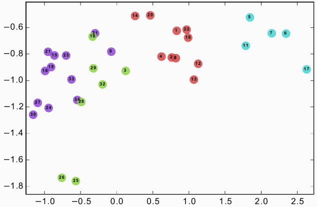
Image scene graph

[Johnson et al., 2018]

Graph Representation Learning

- A key step to applying machine learning algorithms over graphs
- Learn representation of nodes (or graphs) in a low-dimensional space
- **Graph embeddings algorithms:** learn node embeddings directly from topological structure
- **Graph neural networks:** learn how to compute node representation based on local network neighborhood





[Perozzi et al., 2014]

Graphs could be sensitive

- Users' personal attributes, financial transactions, medical/biological networks, ...
- Machine learning algorithms should preserve the privacy of individuals in graph data

Private Machine Learning on Graphs

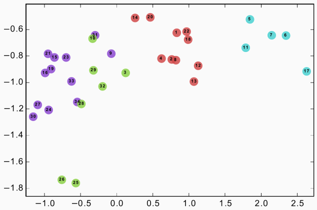
- Privacy-preserving ML methods are mostly designed for non-relational data
- Specific techniques need to be developed to address privacy issues in graphs
- Privacy-preserving graph representation learning tries to fill this gap

Graph Neural Networks

Input: Graph $G = (V, E)$, with node set V and link set E

Objective: Embed each node into a continuous vector space such that similarity in the embedding space approximates similarity in the graph:

$$similarity(u, v) \approx \mathbf{z}_u^T \mathbf{z}_v$$



[Perozzi et al., 2014]

Modern deep learning excels at exploiting grid-structured data

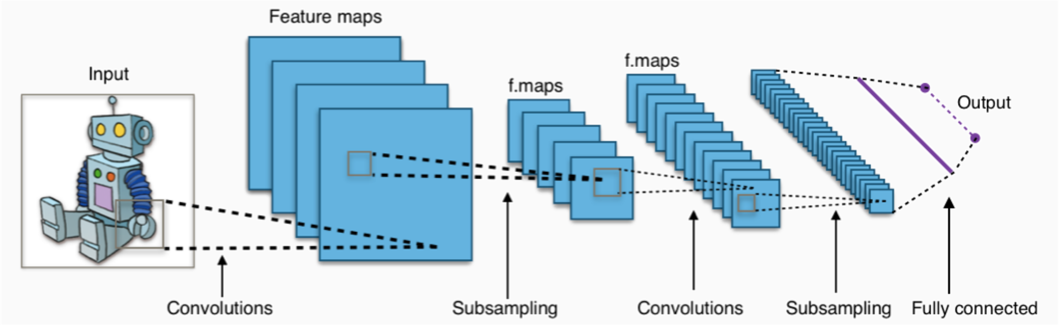
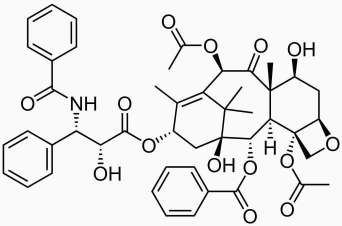


Image source: https://en.wikipedia.org/wiki/Convolutional_neural_network

But graphs are **combinatorial structures**, have arbitrary sizes, and contain multi-modal information

Social Networks

Molecules



Knowledge Graphs

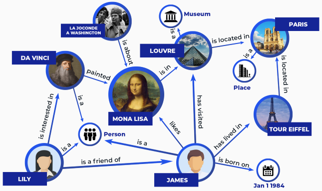
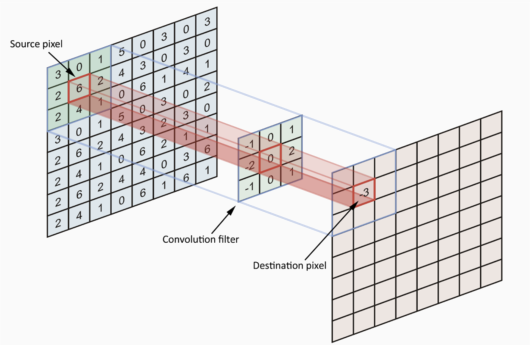


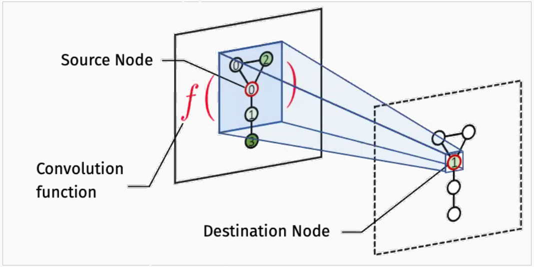
Image source (from left to right): https://yashuseth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kgqa/, https://en.wikipedia.org/wiki/Terpenoid, https://yashuseth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kgqa/

Graph Convolution

Image Convolution



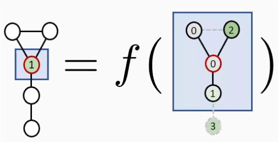
Graph Convolution



Slide adapted from ”A gentle introduction to graph neural networks” by Andreas Loukas.

Input: an input representation vector $\mathbf{h}_v$ for each node v

Output: a new representation vector $\mathbf{h}'_v$ for each node v



$$\mathbf{h}'_v = f\left(\{\mathbf{h}_u : u \in \mathcal{N}(v)\}\right) = \mathrm{UPDATE}\Big(\mathrm{AGGREGATE}\left(\{\mathbf{h}_u : u \in \mathcal{N}(v)\}\right)\Big)$$

- AGGREGATE is a permutation invariant function (e.g., sum, mean, max)
- UPDATE is a neural network (e.g., MLP)

Graph Convolutional Network (GCN) [Kipf and Welling, 2017]

$$\text{AGGREGATE}\left(\{\mathbf{h}_u : u \in \mathcal{N}(v)\}\right) = \sum_{u \in \mathcal{N}(v)} \frac{\mathbf{h}_u}{\sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(v)|}}$$

Graph Sample and Aggregate (GraphSAGE) [Hamilton et al., 2017]

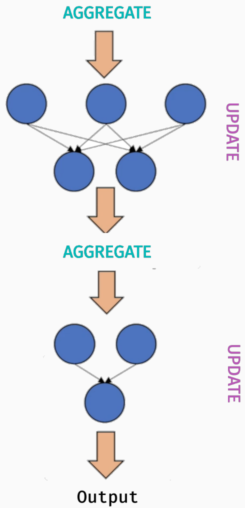
$$\text{AGGREGATE}\left(\{\mathbf{h}_u : u \in \mathcal{N}(v)\}\right) = CONCAT\left(\mathbf{h}_v\,,\; MEAN\left(\{\mathbf{h}_u : u \in \mathcal{N}(v)\}\right)\right)$$

Graph Isomorphism Network (GIN) [Xu et al., 2018]

$$\text{AGGREGATE}\left(\{\mathbf{h}_u : u \in \mathcal{N}(v)\}\right) = (1+\epsilon)\cdot\mathbf{h}_v + SUM\left(\{\mathbf{h}_u : u \in \mathcal{N}(v)\}\right)$$

Input : Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; Feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V} \times d|}$
Output: Embedding vector $\mathbf{z}_v$ for all $v \in \mathcal{V}$
Initialization: $\mathbf{h}_v^0 = \mathbf{x}_v$ for all $v \in \mathcal{V}$
for $l = 1$ *to* L **do**
  foreach *node* $v \in \mathcal{V}$ **do**
    $\mathbf{h}^l_{\mathcal{N}(v)} = AGGREGATE_l\left(\{\mathbf{h}_u^{l-1} : u \in \mathcal{N}(v)\}\right)$
    $\mathbf{h}_v^l = UPDATE_l\left(\mathbf{h}^l_{\mathcal{N}(v)}\right)$
  end
end
return $\mathbf{z}_v = \mathbf{h}_v^L$: *for all* $v \in \mathcal{V}$

You can feed these embeddings into any loss function to train the network parameters



Privacy Attacks on Graph Neural Networks

[Duddu et al., 2020] **Quantifying Privacy Leakage in Graph Embedding**, NeurIPS PPML
[He et al., 2021] **Stealing Links from Graph Neural Networks**, USENIX Security

Potential Attacks on GNNs

Membership Inference Attack

- Infer whether a given node is part of the target graph

Attribute Inference Attack

- Infer sensitive attributes of a node in the target graph

Link Inference Attack

- Infer whether a given pair of nodes are connected in the target graph

Adversary have back-box access to a trained GNN

- The GNN is trained for node classification
- The GNN can be queried to retrieve embeddings or predictions

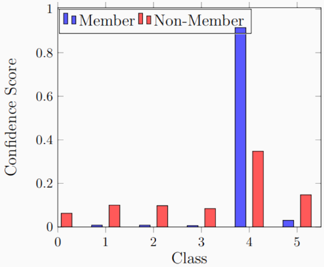
Example

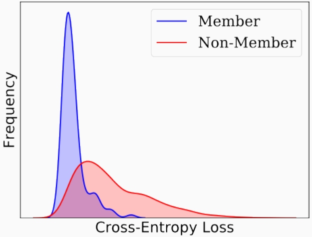
- GNN-based fake account detection service
- Machine Learning as a Service
- Publishing graph embeddings for research purposes

Different attacks may need extra background knowledge

Exploits the statistical difference between the prediction confidence on training and testing data

- GNNs are more confident when predicting labels for the training data
- Nodes with high output confidence are likely members of the training graph





[Duddu et al., 2020]

Confidence Attack (unsupervised)

- Compare the highest prediction confidence of the given node to a **threshold**
- If above the threshold, then member

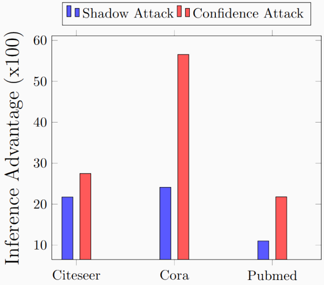
Shadow Attack (supervised)

- Uses an **auxiliary graph** sampled from the training graph
- Train a **similar GNN** over the auxiliary graph and get predictions
- Train a **binary classifier** with prediction scores as features to predict the membership status in the auxiliary graph
- Predict the membership of nodes in the original graph using the learned classifier

Adversary advantage metric

- Estimates model information leakage compared to the random guess

$$I_{adv} = 2 \times (Acc - 0.5)$$



[Duddu et al., 2020]

Exploits the fact that similar users have similar attributes

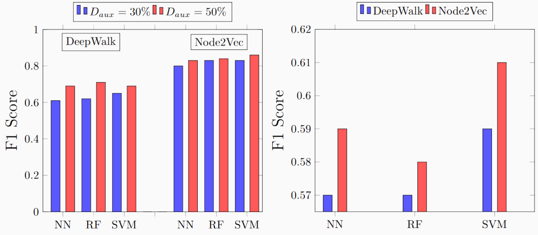
- Similar users are connected
- The connectivity of users is captured by embeddings

Requirements

- Needs embeddings, not predictions
- Requires an subset of nodes with sensitive attributes revealed

Attack Methodology

- Train a classifier with the embedding of the auxiliary graph's nodes as features and their sensitive attribute as label
- Use the trained classifier to predict the sensitive attribute of any given node in the original graph



(a) LastFM (b) Facebook

[Duddu et al., 2020]

Exploits the similarity of prediction posteriors for connected nodes

- If two nodes are connected, then their prediction scores are likely similar

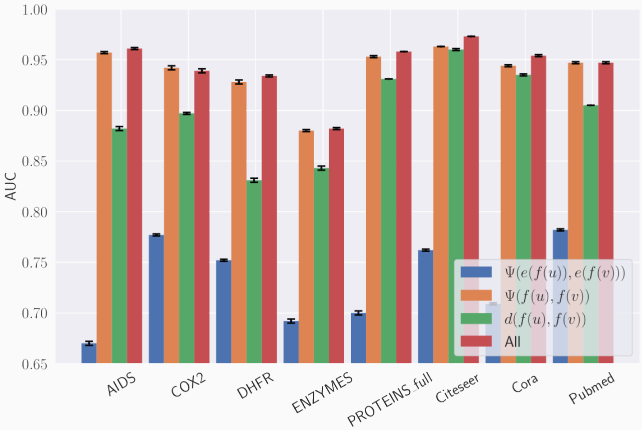
Requirements

- Requires access to an auxiliary subgraph of the original graph

Attack Methodology

- Obtain the prediction scores from the target GNN for every node pair in the auxiliary graph
- Extract features from the obtained scores for each node pair
    - features based on **distance metrics** (cosine, euclidean, etc), **vector operations** (average, hadamard product, etc), and **entropy**
- Train an MLP using the extracted features and the link state in the auxiliary graph
- Use the trained MLP to infer the link between any node pair in the original graph

Link Inference Attack

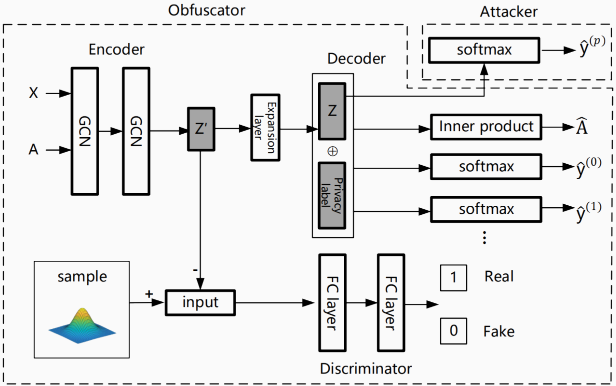


[He et al., 2021]

Privacy-Preserving Graph Neural Network Models

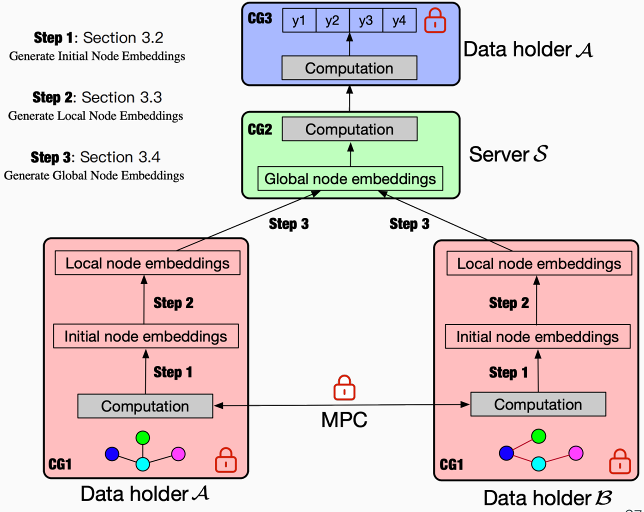
Adversarial Privacy-Preserving Graph Embedding Against Inference Attack [Li et al., 2020]

- **Setting:** graph embeddings are released publicly
- **Goal:** preserving information about the graph structure and utility node attributes
- **Privacy**: mitigating the inference of sensitive node attributes
- **Approach**: adversarial learning



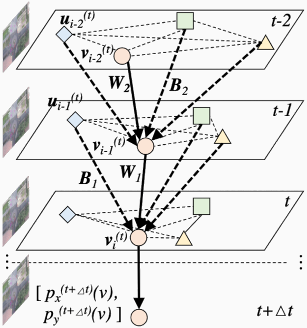
Privacy-Preserving GNN for Node Classification [Zhou et al., 2020]

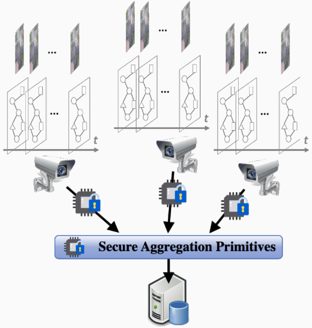
- **Setting:** the graph is partitioned vertically across multiple parties
- **Goal:** learning a global GNN collaboratively
- **Privacy**: keep node features and link information local to each party
- **Approach**: split learning + secure multiparty computation



Federated Dynamic GNN with Secure Aggregation [Jiang et al., 2020]

- **Setting:** each camera has its own graph sequence
- **Goal:** learning a global GNN collaboratively to predict future object positions
- **Privacy**: keep node features and link information local to each camera
- **Approach**: federated learning + secure multiparty computation





Locally Private Graph Neural Networks

[Sajadmanesh and Gatica-Perez, 2020]

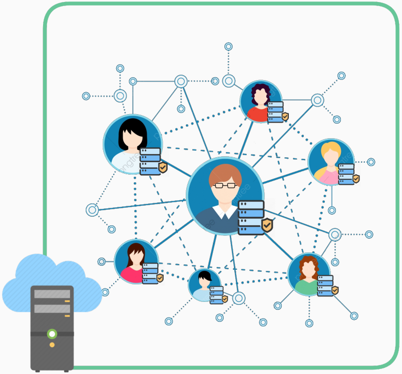
Privacy-Preserving GNN learning with node-level privacy

Setting:

- The server has access to a graph
- Each node has a private feature vector
- Node features are inaccessible by the server

Problem:

- How to learn a GNN without letting the private features leave the nodes?



Infeasible Solution

Why not federated learning?

- Message passing must be done at node side
- Each node requires the private features of its adjacent nodes
    - If sent in plain text $\rightarrow$ privacy violation!
    - If sent using SMC $\rightarrow$ FL + SMC = massive communication!
- **Result:** message passing at node side is not feasible

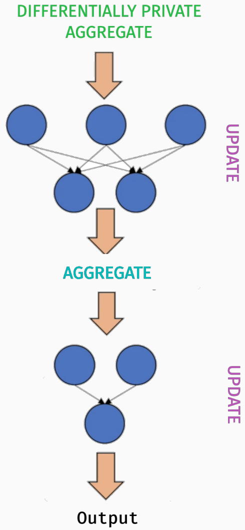
Our Approach

Let's keep the model on the server

- We only need to calculate the first layer's AGGREGATE function privately!
- SMC? It's vulnerable to differential attack!

Idea: make AGGREGATE differentially private by input perturbation!

- Individual features are not necessary, **only aggregated features** are needed
- Node features can be privatized by injecting noise using **Local Differential Privacy** (LDP)
- The neighborhood aggregation will **cancel out the injected noise** in the features



Local Differential Privacy [Kasiviswanathan et al., 2011]

- De facto standard for computing aggregated statistics over private data
- **Key idea:** data holders send perturbed data to the aggregator that are meaningless individually, but can approximate the target statistic when aggregated.
- Composed of two steps:
    1. **Data collection:** each data holder perturbs his data x using a randomized mechanism $\mathcal{M}$, returning $y = \mathcal{M}(x)$ to the aggregator.
    2. **Estimation:** the aggregator computes the target statistic (e.g. mean)
- Randomization in $\mathcal{M}$ provides plausible deniability to data holders
- However, the aggregator must not be able to infer initial data x by observing y and having arbitrary background knowledge

Local Differential Privacy

Given $\epsilon > 0$, a randomized mechanism $\mathcal{M}$ satisfies **ϵ-local differential privacy** if for all possible pairs of user's private data x and x', and for all possible outputs y of $\mathcal{M}$, we have:

$$\Pr[\mathcal{M}(x) = y] \leq e^{\epsilon} \Pr[\mathcal{M}(x') = y]$$

Interpretation

- Any input value x is almost as likely (depending on ϵ) to produce the same output y
- An adversary cannot distinguish between x and x' by observing y

The outline of our locally private GNN (LPGNN) algorithm

1. Each node perturbs its private feature vector using an LDP mechanism and sends it to the server
2. The server uses the received perturbed feature vectors and estimates the first layer's AGGREGATE function
3. The training proceeds with forward and backward propagation as usual
4. Return to step 2 if stopping criteria has not met

But it's not that easy! there are challenges...

Challenge #1: High-dimensional features

- The total privacy budget for a single node scales with the number of features
  $\rightarrow$Too much privacy leakage!
- **Trivial solution:** perturb individual features with ϵ/d privacy budget
  $\rightarrow$Too much noise!

Addressing Challenge #1

Multi-bit mechanism: multidimensional feature perturbation

- Uniformly sample m features out of d ones without replacement
- Perturb each sampled feature with ϵ/m privacy budget
- Map the output of the 1-bit mechanism to either -1 or 1
- For the rest of the features (not sampled) return 0

Algorithm 1: Multi-Bit Mechanism

Input : feature vector $\mathbf{x} \in [\alpha, \beta]^d$; privacy budget $\epsilon > 0$; range parameters α and β; sampling parameter $m \in \{1, 2, \ldots, d\}$.

Output: perturbed vector $\mathbf{x}^* \in \{-1, 0, 1\}^d$.

```
Let S be a set of m values drawn uniformly at
  random without replacement from {1, 2, ..., d}
for i ∈ {1, 2, ..., d} do
    s_i = 1 if i ∈ S otherwise s_i = 0
    t_i ~ Bernoulli( 1/(e^{ε/m}+1) + (x_i−α)/(β−α) · (e^{ε/m}−1)/(e^{ε/m}+1) )
    x*_i = s_i · (2t_i − 1)
end
return x* = [x*_1, ..., x*_d]^T
```

Let $\mathcal{S}$ be a set of m values drawn uniformly at random without replacement from $\{1, 2, \ldots, d\}$

for $i \in \{1, 2, \ldots, d\}$ **do**

$s_i = 1$ if $i \in \mathcal{S}$ otherwise $s_i = 0$

$$t_i \sim \text{Bernoulli}\left(\frac{1}{e^{\epsilon/m}+1} + \frac{x_i-\alpha}{\beta-\alpha} \cdot \frac{e^{\epsilon/m}-1}{e^{\epsilon/m}+1}\right)$$

$x_i^* = s_i \cdot (2t_i - 1)$

end

return $\mathbf{x}^* = [x_1^*, \ldots, x_d^*]^T$

Approximation of Graph Convolution

- The server can estimate the first layer's graph convolution by:

$$\mathbf{x}'_u = \frac{d(\beta-\alpha)}{2m} \cdot \frac{e^{\epsilon/m}+1}{e^{\epsilon/m}-1} \cdot \mathbf{x}^*_u + \frac{\alpha+\beta}{2}$$

$$\widehat{\mathbf{h}}_{\mathcal{N}(v)} = \text{AGGREGATE}\left(\{\mathbf{x}'_u, \forall u \in \mathcal{N}(v)\}\right)$$

$$\mathbf{h}_v = \text{UPDATE}\left(\widehat{\mathbf{h}}_{\mathcal{N}(v)}\right)$$

Theorem 3.1

The multi-bit mechanism satisfies ϵ-LDP for each node.

Proposition 3.5

The optimal value of the sampling parameter m in the multi-bit mechanism is: $m^\star = \max(1, \min(d, \lfloor \frac{\epsilon}{2.18} \rfloor))$

Challenge #2: Small-size neighborhood

- Lots of the nodes have too few neighbors (remember Power-Law degree distribution?)
- The neighborhood aggregator cannot cancel out the noise if the neighborhood size is small

KProp convolution layer: neighborhood expansion method

- Expands the neighborhood to the nodes that are up to K-hops away
- Applies K consecutive AGGREGATE function
- Applies the UPDATE function after the K-th AGGREGATE
- Trade-off between the aggregation estimation error and the GNN expressive power

Algorithm 2: KProp Convolution Layer

Input : Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$; linear aggregator function AGGREGATE; Non-linear update function UPDATE; step parameter $K \geq 1$;

Output: embedding vector $\mathbf{h}_v, \forall v \in \mathcal{V}$

```
for all v ∈ V do in parallel
    h^0_{N(v)} = x_v
    for k = 1 to K do
        h^k_{N(v)} =
            AGGREGATE({h^{k-1}_{N(u)}, ∀u ∈ N(v) − {v}})
    end
    h_v = UPDATE(h^K_{N(v)})
endfor
return {h_v, ∀v ∈ V}
```

$$\mathbf{h}^0_{\mathcal{N}(v)} = \mathbf{x}_v$$

$$\mathbf{h}^k_{\mathcal{N}(v)} = \text{AGGREGATE}\left(\{\mathbf{h}^{k-1}_{\mathcal{N}(u)}, \forall u \in \mathcal{N}(v) - \{v\}\}\right)$$

$$\mathbf{h}_v = \text{UPDATE}\left(\mathbf{h}^K_{\mathcal{N}(v)}\right)$$

Statistics of Datasets

Dataset	#Classes	#Nodes	#Edges	#Features	Avg. Deg.
Cora	7	2,708	5,278	1,433	3.90
Citeseer	6	3,327	4,552	3,703	2.74
Pubmed	3	19,717	44,324	500	4.50
Facebook	4	22,470	170,912	4,714	15.21
Github	2	37,700	289,003	4,005	15.33
LastFM	18	7,624	27,806	7,842	7.29

Comparison methods

- GCN+Raw: A standard two-layer GCN trained on raw features **(non-private)**
- LPGNN: A two layer GNN (KProp as the first, GCN as the second layer) trained on perturbed features using the multi-bit mechanism **(locally-private)**
- GCN+Rnd: Similar to GCN+Raw, but trained on random features **(fully-private)**
- GCN+Ohd: Similar to GCN+Raw, but trained on “one-hot degree” features **(fully-private)**

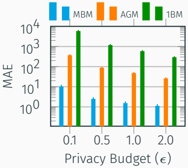
Analysis of Accuracy-Privacy Trade-Off

Micro-F1 score of different methods for node classification under varying privacy budget (ϵ)

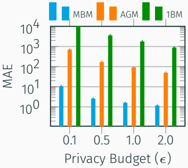
Dataset	GCN +Raw	LPGNN $\epsilon = 0.1$	LPGNN $\epsilon = 0.5$	LPGNN $\epsilon = 1.0$	LPGNN $\epsilon = 2.0$	GCN +Rnd	GCN +Ohd
Cora	87.5 ± 0.2	81.4 ± 4.8	83.3 ± 1.5	83.6 ± 1.1	83.6 ± 0.7	58.1 ± 7.8	29.3 ± 0.2
Citeseer	74.1 ± 0.3	64.5 ± 1.1	66.0 ± 1.0	66.5 ± 0.9	66.8 ± 0.8	29.6 ± 6.5	27.2 ± 0.1
Pubmed	87.6 ± 0.1	81.9 ± 0.3	82.0 ± 0.3	82.2 ± 0.3	82.2 ± 0.3	53.5 ± 1.1	50.3 ± 0.1
Facebook	94.9 ± 0.1	92.4 ± 0.5	93.2 ± 0.3	93.4 ± 0.3	93.4 ± 0.3	31.8 ± 2.1	63.8 ± 0.3
Github	87.1 ± 0.1	84.1 ± 3.4	85.7 ± 0.8	86.1 ± 0.3	86.2 ± 0.1	74.3 ± 0.0	83.7 ± 0.0
LastFM	88.2 ± 0.3	76.3 ± 1.5	82.7 ± 1.6	84.3 ± 0.8	84.8 ± 0.7	21.8 ± 1.2	45.3 ± 0.7

Effect of the Multi-Bit Mechanism

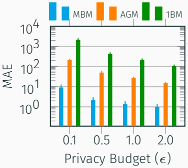
Mean absolute error of the multi-bit (MBM), 1-bit (1BM), and the Analytic Gaussian (AGM) mechanisms in estimating the neighborhood aggregation



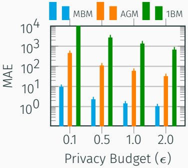
Cora



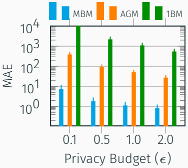
Citeseer



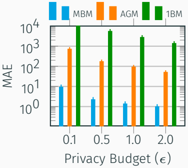
Pubmed



Facebook



Github



LastFM

Research Directions and Conclusion

Differentially Private GNNs

- How to build privacy-preserving GNNs satisfying graph-based notions of differential privacy?
- Edge-DP and Node-DP?

Privacy-Preserving Distributed GNN Learning

- How to remove the trusted server in multi-party GNN training?
- Multi-layer networks?

Privacy-Preserving Deep Graph Learning

- How to learn a latent graph from private data?
- Privacy-preserving graph-based classifier?

Conclusion

Graphs are likely to be sensitive

- social connections, financial transactions, disease outbreak, ...

Graph representation algorithms are vulnerable to privacy attacks

- Simple but effective attacks has already been proposed

Common privacy-preserving ML methods cannot trivially be applied on graphs

- e.g., the exhaustive communication cost of federated learning

Privacy-preserving graph representation learning aims to address privacy issue of applying deep learning over graphs

- This is a new-born field of research with lots of opportunities and open questions

If you are interested, please get in touch!

Thank You!

Questions?

@sajadmanesh
sajadmanesh@idiap.ch

References I

Ahmad, I., Akhtar, M. U., Noor, S., and Shahnaz, A. (2020).
Missing link prediction using common neighbor and centrality based parameterized algorithm.
Scientific Reports, 10(1):1–9.

Duddu, V., Boutet, A., and Shejwalkar, V. (2020).
Quantifying privacy leakage in graph embedding.
arXiv preprint arXiv:2010.00906.

Hamilton, W., Ying, Z., and Leskovec, J. (2017).
Inductive representation learning on large graphs.
In *Advances in neural information processing systems*, pages 1024–1034.

References II

He, X., Jia, J., Backes, M., Gong, N. Z., and Zhang, Y. (2021).
Stealing links from graph neural networks.
In *30th USENIX Security Symposium (USENIX Security 21)*, Vancouver, B.C. USENIX Association.

Jiang, M., Jung, T., Karl, R., and Zhao, T. (2020).
Federated dynamic gnn with secure aggregation.
arXiv preprint arXiv:2009.07351.

Johnson, J., Gupta, A., and Fei-Fei, L. (2018).
Image generation from scene graphs.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1219–1228.

Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. (2011).
What can we learn privately?
SIAM Journal on Computing, 40(3):793–826.

Kipf, T. N. and Welling, M. (2017).
Semi-supervised classification with graph convolutional networks.
In *International Conference on Learning Representations (ICLR)*.

Li, K., Luo, G., Ye, Y., Li, W., Ji, S., and Cai, Z. (2020).
Adversarial privacy preserving graph embedding against inference attack.
IEEE Internet of Things Journal.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014).
Deepwalk: Online learning of social representations.
In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710.

Sajadmanesh, S. and Gatica-Perez, D. (2020).
Locally private graph neural networks.
arXiv preprint arXiv:2006.05535.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018).
How powerful are graph neural networks?
arXiv preprint arXiv:1810.00826.

Zhou, J., Chen, C., Zheng, L., Zheng, X., Wu, B., Liu, Z., and Wang, L. (2020).
Privacy-preserving graph neural network for node classification.
arXiv preprint arXiv:2005.11903.