Locally Private Graph Neural Networks

Sina Sajadmanesh Daniel Gatica-Perez

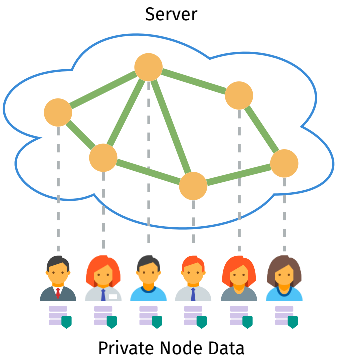
Idiap Research Institute
EPFL

ACM SIGSAC Conference on Computer and Communications Security – November 2021

Learning Graph Neural Networks (GNNs) with node data privacy

Motivating Example:

- A social network server (e.g., Facebook) wishes to train a GNN over the graph of users
- Need to access user's personal data, such as mobile sensor data for training a better model



Learning Graph Neural Networks (GNNs) with node data privacy

Motivating Example:

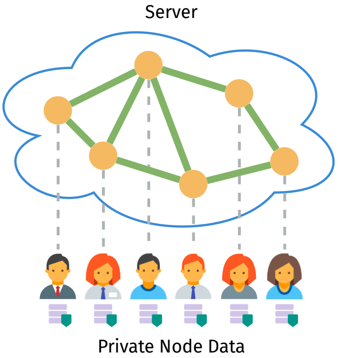
- A social network server (e.g., Facebook) wishes to train a GNN over the graph of users
- Need to access user's personal data, such as mobile sensor data for training a better model

Assumptions:

- Graph topology is public to the server
- Node data (features/labels) are private to nodes

Problem:

- How to let the server train a GNN without giving up private node data?



Our Approach: Preserve the privacy of nodes using
Local Differential Privacy

- Multi-bit mechanism for high-dimensional feature perturbation
- KProp layer for better feature and label estimation
- Drop algorithm for learning with privatized labels

Procedure

- Data holders perturb their data using a randomized mechanism
- The aggregator estimates the target statistics by aggregating perturbed data
  - The noise cancels out through aggregation

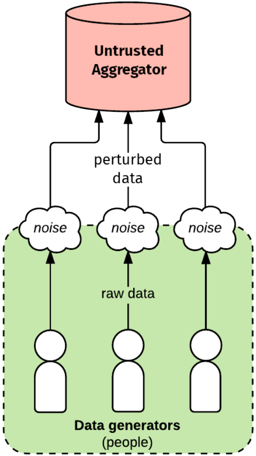


Image Credit: Bennett Cyphers

Local Differential Privacy (LDP)

Procedure

- Data holders perturb their data using a **randomized mechanism**
- The aggregator **estimates** the target statistics by **aggregating** perturbed data
  - The noise cancels out through aggregation

Definition

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

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

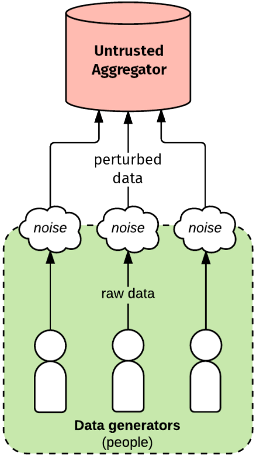


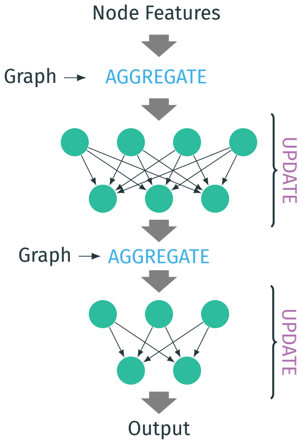
Image Credit: Bennett Cyphers

WHY LOCAL DP?

GNN objective: learn a representation vector for every node

AGGREGATE: nodes aggregate their neighbors' representation vector using a permutation invariant function (e.g., mean, sum, or max)

UPDATE: a neural network generates new node representations from aggregated vectors

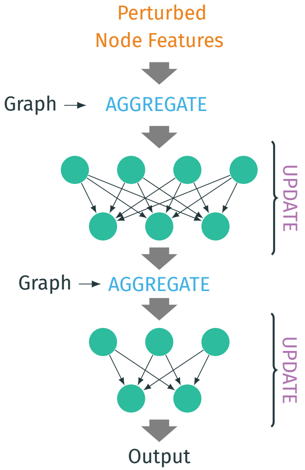


Why Local DP?

GNN objective: learn a representation vector for every node

AGGREGATE: nodes aggregate their neighbors' representation vector using a permutation invariant function (e.g., mean, sum, or max)

UPDATE: a neural network generates new node representations from aggregated vectors



Private neighborhood aggregation with LDP

- Node features are perturbed by injecting noise
- The neighborhood aggregation cancels out the noise

Challenges

High-dimensional features

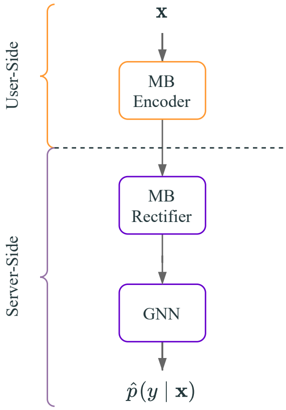
- The total privacy budget of a node scales with the number of features
  - Keeping the total privacy budget small $\longrightarrow$ **Too much noise!**

Multi-bit Encoder

- Runs at user-side
- Performs **randomized feature selection**, **perturbation**, and **compression**
- Introduces bias into the features

Multi-bit Rectifier

- Runs at server-side
- Performs feature **decompression** and **debiasing**

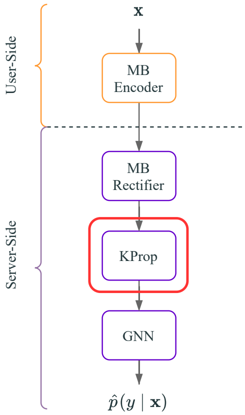


Small neighborhoods

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

Our solution: KProp denoising layer

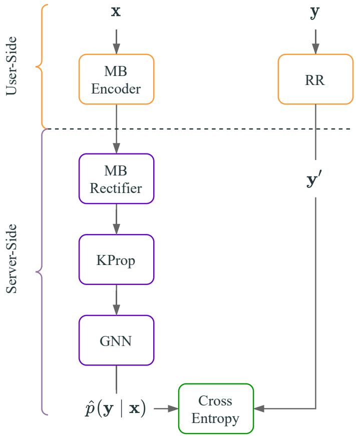
- Expands the neighborhood to the nodes that are up to K-hops away
- Applies K consecutive linear AGGREGATE functions
  - No non-linearity in between
- Can be prepended to any GNN architecture
  - Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), GraphSAGE, . . .



Randomized Response for label perturbation

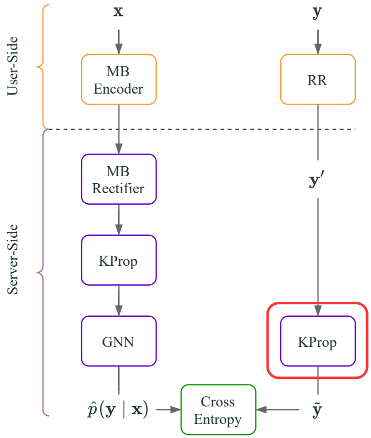
- True label $\mathbf{y}$
- Perturbed label $\mathbf{y}'$
- Number of classes c
- DP privacy budget ϵ

$$y' = \begin{cases} y & \text{w.p. } \frac{e^{\epsilon}}{e^{\epsilon}+c-1} \\ \text{another random label} & \text{o.w.} \end{cases}$$



KProp for label denoising

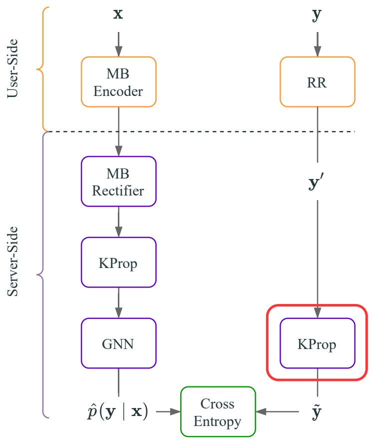
- Neighboring nodes tend to have **similar labels**
- Node label can be estimated by aggregating neighboring labels
- KProp can help overcoming small neighborhoods



KProp for label denoising

- Neighboring nodes tend to have **similar labels**
- Node label can be estimated by aggregating neighboring labels
- KProp can help overcoming small neighborhoods

How to find best KProp iteration (and other hyper-parameters) without clean labels?

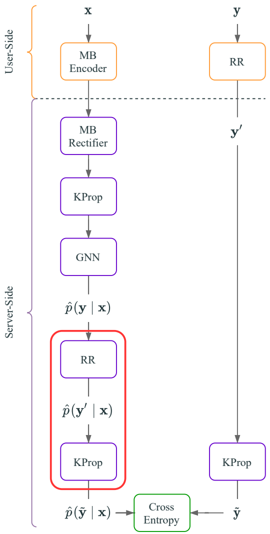


Learning without Clean Data

- Validation data is used for **model selection** and **early stopping**
- **No clean data** for validation in our problem due to privacy
  - More realistic assumption in real-world scenarios
- Need alternative methods to **prevent overfitting** and **selecting hyper-parameters**

Prevent overfitting the recovered labels

- We want the GNN to be the **predictor of true labels**, not the recovered ones
- The predictions should go through **the same process** as the labels
  - We apply RR and KProp on predictions as well

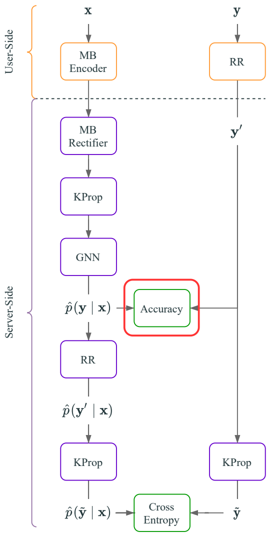


Prevent overfitting the noisy labels

- Overfitting still happens if KProp step is not enough for efficient denoising
- RR gives us an expected label accuracy:

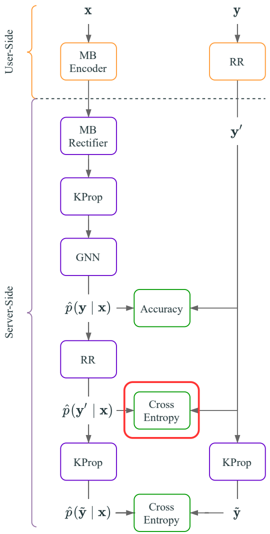
$$p(\mathbf{y}' = \mathbf{y}) = \frac{e^{\epsilon}}{e^{\epsilon} + c - 1}$$

- We stop training when GNN's accuracy for predicting noisy labels **goes beyond this value**
  - Achieving a higher accuracy is a signal of overfitting



Model selection using Forward Correction loss

- Calculated between noisy labels and noisy predictions
- An unbiased estimator for the true loss



Experiment Setting

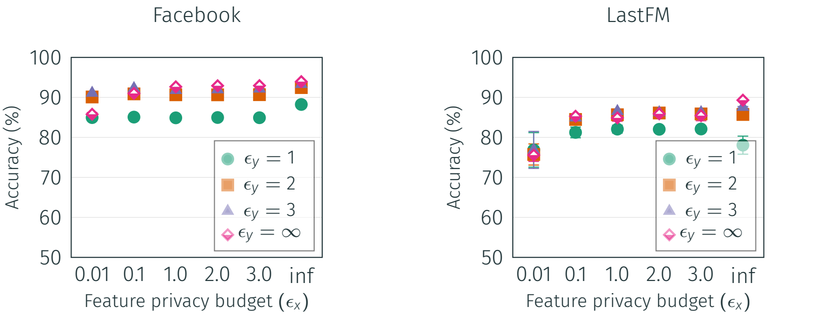
- Learning Task: **Node Classification**
- Backbone Model: **2-layer GNN**
  - Default: GraphSAGE
- KProp Aggregation: **GCN**

Datasets

Dataset	Classes	Nodes	Edges	Features	Avg. Degree
Cora	7 categories	2,708 documents	5,278 citations	1,433	3.90
PubMed	3 categories	19,717 documents	44,324 citations	500	4.50
Facebook	4 categories	22,470 pages	170,912 likes	4,714	15.21
LastFM	10 countries	7,083 users	25,814 friendships	7,842	7.29

Privacy vs. Accuracy Trade-Off

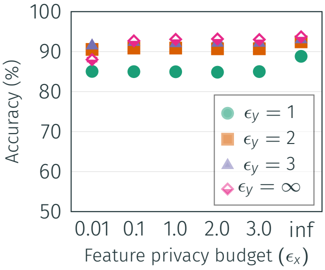
- Base GNN: GraphSAGE



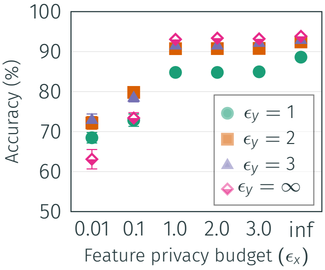
LPGNN is very robust to noisy features!

- Dataset: Facebook

Graph Convolutional Network (GCN)



Graph Attention Network (GAT)



Feature-dependent models suffer at high-privacy regimes!

Comparison Against Privacy-Free Features

- Base GNN: GraphSAGE
- Label privacy budget $\epsilon_y = 1$

Feature	Cora	PubMed	Facebook	LastFm
All Ones	22.6 ± 5.0	38.9 ± 0.4	29.0 ± 1.4	19.6 ± 1.8
One-Hot Degree	44.4 ± 3.5	52.5 ± 5.7	77.2 ± 0.3	66.4 ± 1.6
Random	26.4 ± 3.0	56.0 ± 1.3	35.2 ± 5.6	32.3 ± 6.3
Multi-Bit ($\epsilon_x = 1$)	**69.3 ± 1.2**	**74.9 ± 0.3**	**84.9 ± 0.2**	**82.1 ± 1.0**

Meaningful node features, even privatized, are very helpful!

Comparison of LDP Mechanisms

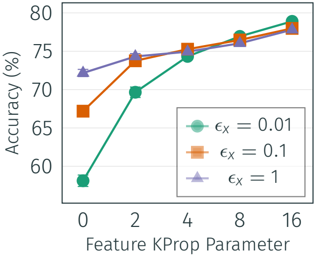
- Base GNN: GraphSAGE
- Label privacy budget $\epsilon_y = \infty$

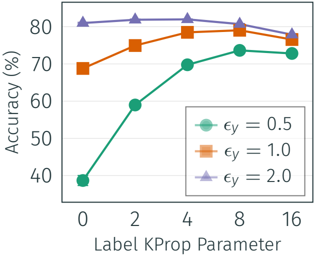
Dataset	Mechanism	$\epsilon_x = 0.1$	$\epsilon_x = 1$	$\epsilon_x = 2$
Cora	Laplace	57.8 ± 2.3	61.9 ± 3.1	58.1 ± 2.1
	Gaussian	62.7 ± 2.8	67.5 ± 3.0	77.2 ± 1.9
	Multi-Bit	**64.6 ± 3.2**	**83.9 ± 0.4**	**84.0 ± 0.3**
Facebook	Laplace	72.5 ± 2.1	85.4 ± 0.4	84.8 ± 1.6
	Gaussian	85.6 ± 0.7	92.0 ± 0.1	92.4 ± 0.2
	Multi-Bit	**91.0 ± 0.4**	**92.7 ± 0.1**	**92.9 ± 0.1**

Perturbing fewer features but with higher privacy budget is better!

Effect of KProp in Performance Improvement

- Dataset: Pubmed
- Base GNN: GraphSAGE





KProp significantly boosts accuracy, especially at larger noises!

Comparison of Different Learning Algorithms

- Base GNN: GraphSAGE
- Feature privacy budget $\epsilon_x = 1$

Dataset	ϵ_y	Cross Entropy	Forward Correction	Drop
Cora	0.5	18.6 ± 1.3	18.6 ± 2.5	**42.9 ± 1.5**
	1.0	25.5 ± 1.7	37.1 ± 2.5	**69.3 ± 1.2**
	2.0	52.9 ± 2.1	75.1 ± 1.0	**78.4 ± 0.7**
LastFm	0.5	21.1 ± 4.6	44.9 ± 5.3	**70.0 ± 3.0**
	1.0	28.4 ± 2.5	58.5 ± 3.6	**82.1 ± 1.0**
	2.0	56.8 ± 2.8	79.2 ± 1.3	**85.7 ± 0.7**

DROP significantly outperforms baseline methods!

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
- GNN models demonstrate graceful accuracy-privacy trade-off
  - Feature privacy almost comes for free in simpler models
  - Label privacy with low privacy budget gives acceptable accuracy

Thank You!

sajadmanesh@idiap.ch
sajadmanesh

Multi-bit mechanism for high-dimensional perturbation

Multi-bit Encoder (user-side):

- **Feature selection:** pick m **out of** d **dimensions** uniformly at random
- **Perturbation:** perturb selected features using **1-bit mechanism** with ϵ/m privacy budget per feature:

$$x^{\star}_{v,i} \sim \text{Bernoulli}\left(\frac{1}{e^{\epsilon/m}+1} + \frac{x_{v,i}-\alpha}{\beta-\alpha}\cdot\frac{e^{\epsilon/m}-1}{e^{\epsilon/m}+1}\right)$$

- **Compression:** Map 1-bit output to either -1 or 1, return 0 for non-selected

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

Multi-bit mechanism for high-dimensional perturbation

Multi-bit Encoder (user-side):

- **Feature selection:** pick m **out of** d **dimensions** uniformly at random
- **Perturbation:** perturb selected features using **1-bit mechanism** with ϵ/m privacy budget per feature:

$$x^{\star}_{v,i} \sim \text{Bernoulli}\left(\frac{1}{e^{\epsilon/m}+1} + \frac{x_{v,i}-\alpha}{\beta-\alpha}\cdot\frac{e^{\epsilon/m}-1}{e^{\epsilon/m}+1}\right)$$

- **Compression:** Map 1-bit output to either -1 or 1, return 0 for non-selected

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

This process introduces bias into the features

Multi-bit mechanism for high-dimensional perturbation

Multi-bit Rectifier (server-side):

- **Decompression and de-biasing:** reverse the encoder's mapping:

$$x'_{v,i} = \frac{d(\beta - \alpha)}{2m} \cdot \frac{e^{\epsilon/m} + 1}{e^{\epsilon/m} - 1} \cdot x^{\star}_{v,i} + \frac{\alpha + \beta}{2}$$

- Optimal m is found by minimizing the rectifier's variance:

$$m^{\star} = \max(1, \min(d, \left\lfloor \frac{\epsilon}{2.18} \right\rfloor))$$