Locally Private Graph Neural Networks

Sina Sajadmanesh   Daniel Gatica-Perez

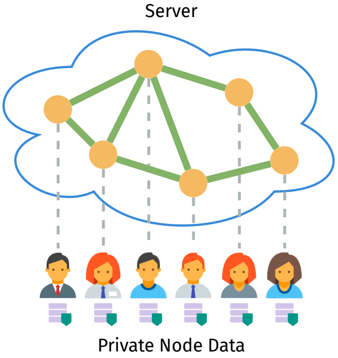
Idiap Research Institute
EPFL

Graph Neural Networks User Group Meetup – July 29, 2021

Graph learning with node data privacy

Assumptions:

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



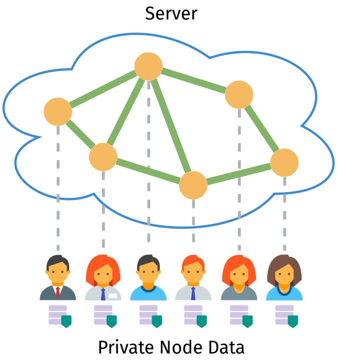
Graph learning with node data privacy

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?



Graph learning with node data privacy

Assumptions:

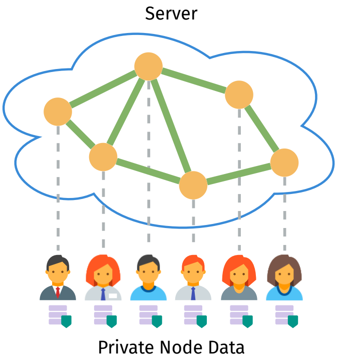
- Graph topology is public to the server
- Node data (features/labels) are private to nodes

Problem:

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

Solution:

- Preserve the privacy of nodes using Local Differential Privacy



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

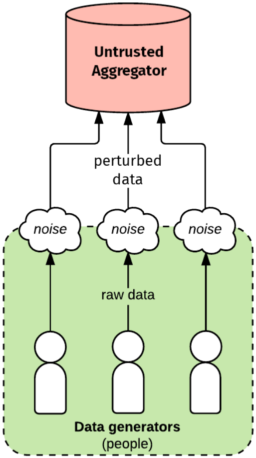


Image Credit: Bennett Cyphers

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:

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

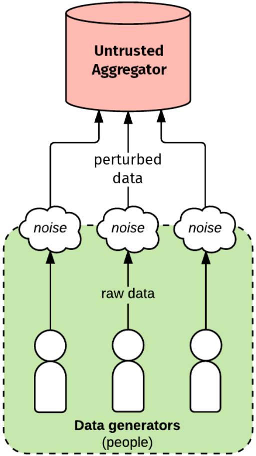
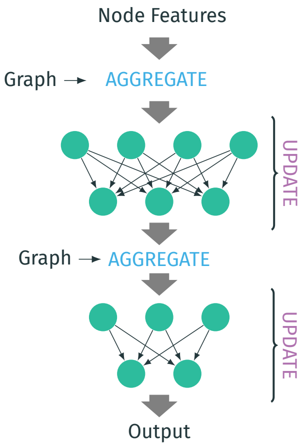


Image Credit: Bennett Cyphers

GNNs are message-passing neural networks

AGGREGATE: nodes aggregate their neighbors' representation vector

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



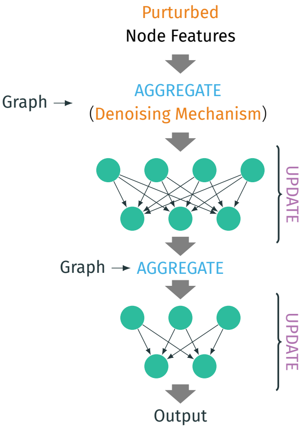
GNNs are message-passing neural networks

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 $\longrightarrow$ **Too much noise!**

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!

Small neighborhoods

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

Challenge: High-Dimensional Features

Multi-bit mechanism for multidimensional perturbation

- **Multi-bit Encoder:** for feature selection, perturbation, and compression
  - Runs at user-side

- **Multi-bit Rectifier:** for decompression and de-biasing
  - Runs at server-side

Challenge: High-Dimensional Features

Multi-bit mechanism for multidimensional perturbation

- **Multi-bit Encoder:** for feature selection, perturbation, and compression
  - Runs at user-side
  - Randomly perturb m out of d features with ϵ/m privacy budget using 1-bit mechanism:

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

- **Multi-bit Rectifier:** for decompression and de-biasing
  - Runs at server-side

Multi-bit mechanism for multidimensional perturbation

- **Multi-bit Encoder:** for feature selection, perturbation, and compression
  - Runs at user-side
  - Randomly perturb m out of d features with ϵ/m privacy budget using 1-bit mechanism:

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

  - Map 1-bit output to either -1 or 1, return 0 for non-perturbed
- **Multi-bit Rectifier:** for decompression and de-biasing
  - Runs at server-side

Challenge: High-Dimensional Features

Multi-bit mechanism for multidimensional perturbation

- **Multi-bit Encoder:** for feature selection, perturbation, and compression
  - Runs at user-side
  - Randomly perturb m out of d features with ϵ/m privacy budget using 1-bit mechanism:

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

  - Map 1-bit output to either -1 or 1, return 0 for non-perturbed
- **Multi-bit Rectifier:** for decompression and de-biasing
  - Runs at server-side
  - Encoder's output is biased

Multi-bit mechanism for multidimensional perturbation

- **Multi-bit Encoder:** for feature selection, perturbation, and compression
  - Runs at user-side
  - Randomly perturb m out of d features with ϵ/m privacy budget using 1-bit mechanism:

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

  - Map 1-bit output to either -1 or 1, return 0 for non-perturbed
- **Multi-bit Rectifier:** for decompression and de-biasing
  - Runs at server-side
  - Encoder's output is biased
  - De-biases encoded features by reversing the encoder's mapping:

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

Challenge: Small neighborhoods

Our solution: KProp denoising layer

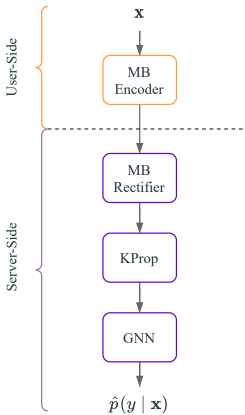
- Expands the neighborhood to the nodes that are up to K-hops away
- Applies K consecutive linear AGGREGATE functions
  - No non-linearity in between
- Can be prepended to any GNN architecture

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



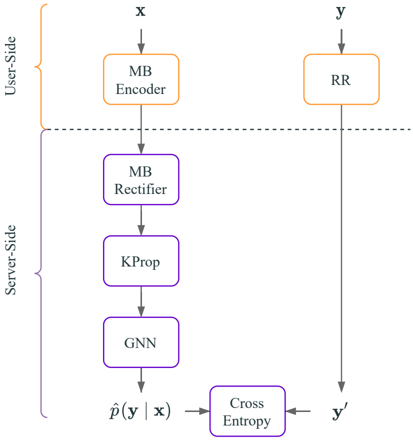
Randomized Response for label differential privacy

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

$$p(\mathbf{y}' \mid \mathbf{y}) = \begin{cases} \frac{e^{\epsilon}}{e^{\epsilon}+c-1}, & \text{if } \mathbf{y}' = \mathbf{y} \\ \frac{1}{e^{\epsilon}+c-1}, & \text{otherwise} \end{cases}$$

Trivial method: directly train GNN with noisy labels

- GNN severely overfits the noisy labels
- Poor generalization performance

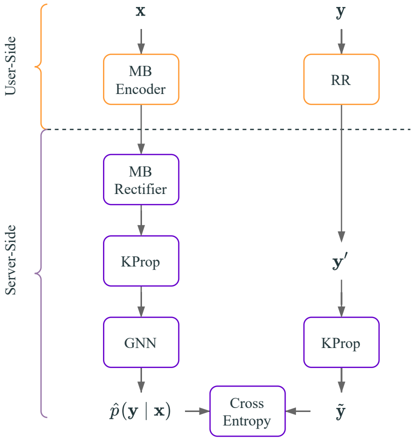


Trivial method: directly train GNN with noisy labels

- GNN severely overfits the noisy labels
- Poor generalization performance

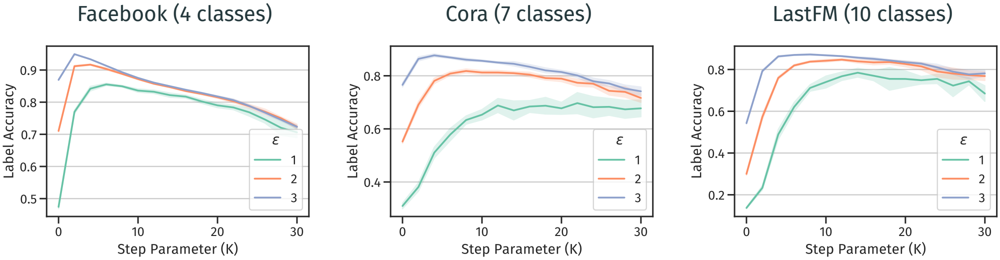
Idea: use KProp for label denoising

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



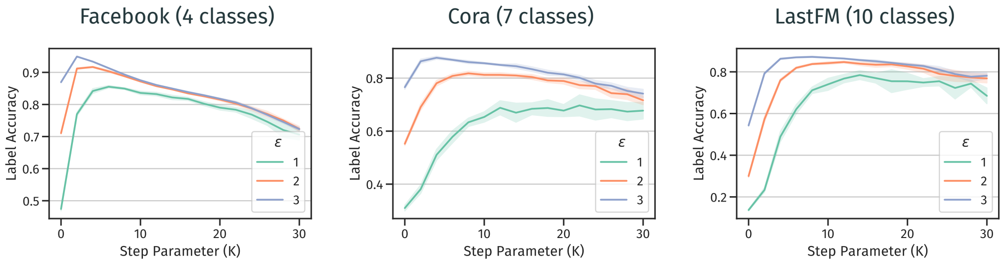
Effect of KProp on label accuracy

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



Effect of KProp on label accuracy

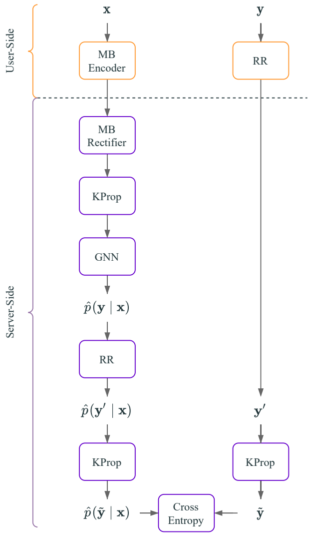
- Accuracy between true label **y** and recovered label $\tilde{\mathbf{y}}$



How to find best performing K without clean validation data?

Prevent absorbing noise in $\tilde{\mathbf{y}}$

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

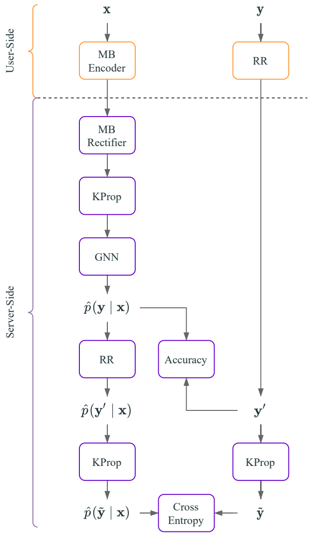


Prevent absorbing noise in $\tilde{\mathbf{y}}$

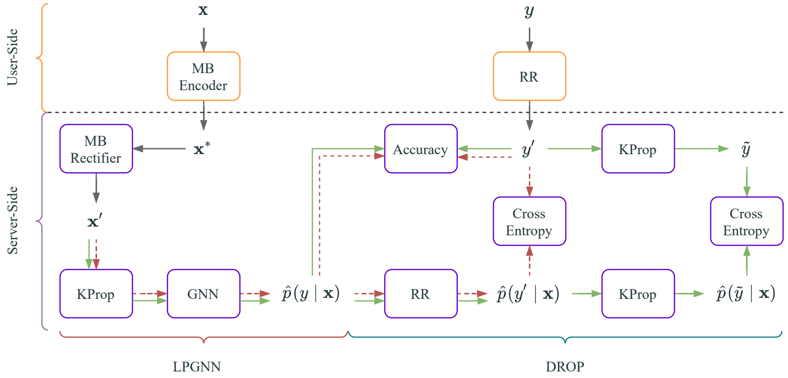
- $\mathbf{y}$ is perturbed by RR and is given to KProp to yield $\tilde{\mathbf{y}}$
- Apply the **same process** on $\hat{p}\,(\mathbf{y} \mid \mathbf{x})$ to obtain $\hat{p}\,(\tilde{\mathbf{y}} \mid \mathbf{x})$
- Train $\hat{p}\,(\tilde{\mathbf{y}} \mid \mathbf{x})$ with $\tilde{\mathbf{y}}$

Prevent absorbing noise in $\mathbf{y}'$

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

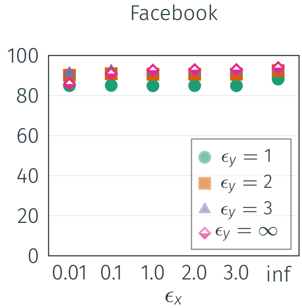


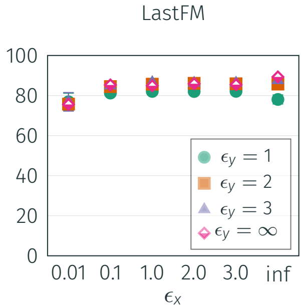
Label **D**enoising with **Prop**agation (**Drop**)



LPGNN's performance under varying feature and label privacy budgets

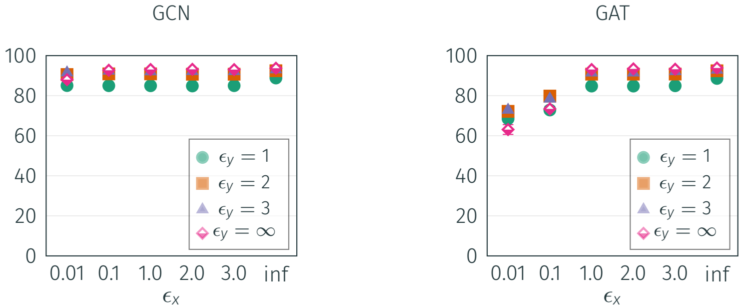
- Base GNN: GraphSAGE





Comparison of base GNN architectures

- Dataset: Facebook



Comparison against ad-hoc features

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

Feature	Cora	Pubmed	Facebook	LastFm
Ones	22.6 ± 5.0	38.9 ± 0.4	29.0 ± 1.4	19.6 ± 1.8
Ohd	44.4 ± 3.5	52.5 ± 5.7	77.2 ± 0.3	66.4 ± 1.6
Rnd	26.4 ± 3.0	56.0 ± 1.3	35.2 ± 5.6	32.3 ± 6.3
Mbm ($\epsilon_x = 1$)	$\mathbf{69.3 \pm 1.2}$	$\mathbf{74.9 \pm 0.3}$	$\mathbf{84.9 \pm 0.2}$	$\mathbf{82.1 \pm 1.0}$

Comparison of different learning algorithms

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

DATASET	ϵ_y	CROSS ENTROPY	FORWARD CORRECTION	DROP
CORA	0.5	18.6 ± 1.3	18.6 ± 2.5	$\mathbf{42.9 \pm 1.5}$
	1.0	25.5 ± 1.7	37.1 ± 2.5	$\mathbf{69.3 \pm 1.2}$
	2.0	52.9 ± 2.1	75.1 ± 1.0	$\mathbf{78.4 \pm 0.7}$
FACEBOOK	0.5	50.9 ± 4.2	68.9 ± 1.3	$\mathbf{75.1 \pm 0.6}$
	1.0	55.2 ± 1.3	73.8 ± 1.1	$\mathbf{84.9 \pm 0.2}$
	2.0	81.6 ± 1.2	88.9 ± 0.2	$\mathbf{90.7 \pm 0.1}$
LASTFM	0.5	21.1 ± 4.6	44.9 ± 5.3	$\mathbf{70.0 \pm 3.0}$
	1.0	28.4 ± 2.5	58.5 ± 3.6	$\mathbf{82.1 \pm 1.0}$
	2.0	56.8 ± 2.8	79.2 ± 1.3	$\mathbf{85.7 \pm 0.7}$

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 demostrate reasonable accuracy-privacy trade-off
  - Feature privacy almost comes for free in simpler models
  - Label privacy with low to mederate privacy budget gives acceptable accuracy

Thank You!

sajadmanesh

sajadmanesh@idiap.ch