Machine Unlearning on Graphs

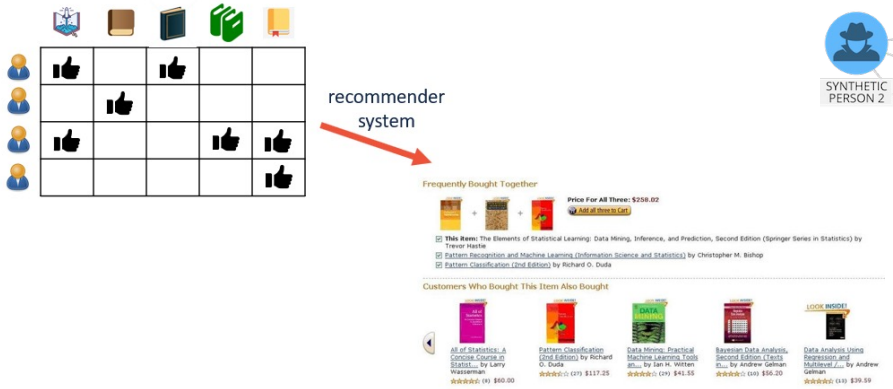
Jian Kang

09/01/2022

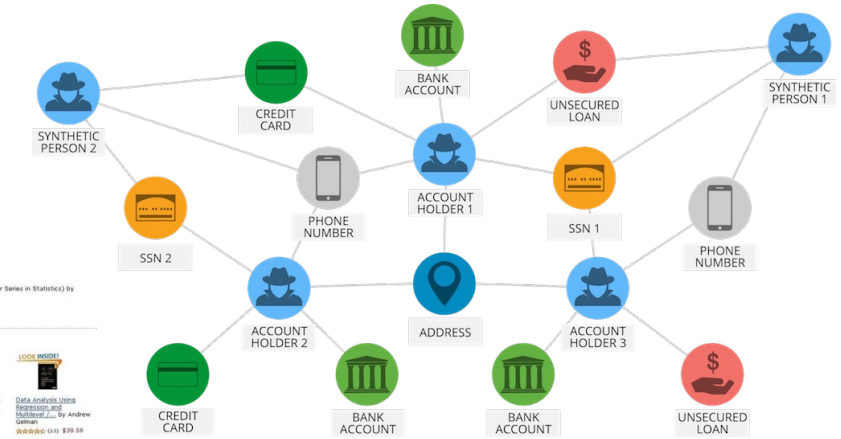
ILLINOIS

Graph Neural Networks: Applications

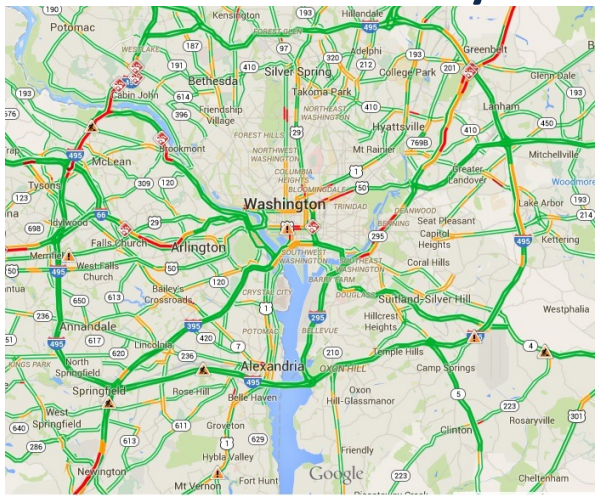
Social network analysis



E-commerce

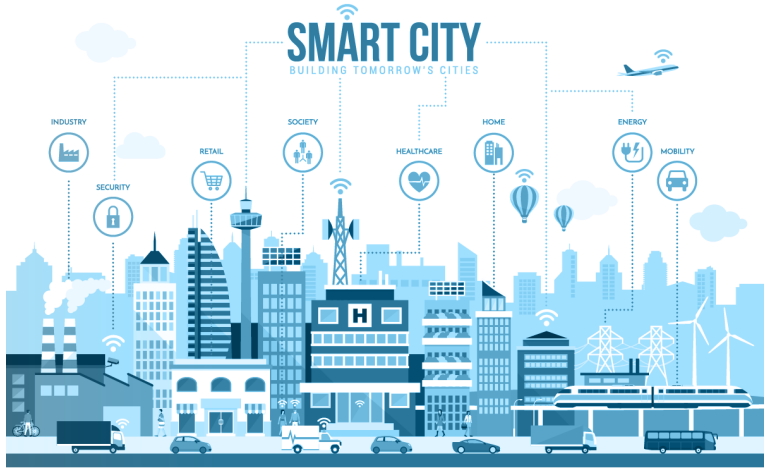


Financial fraud detection



Traffic prediction

Computational bioinformatics



Smart city

[1] Tsitsulin, A., Palowitch, J., Perozzi, B., & Müller, E. (2020). Graph Clustering with Graph Neural Networks. arXiv.
[2] Wang, X., He, X., Wang, M., Feng, F., & Chua, T. S. (2019). Neural Graph Collaborative Filtering. SIGIR 2019.
[3] Liu, Z., Dou, Y., Yu, P. S., Deng, Y., & Peng, H. (2020). Alleviating the Inconsistency Problem of Applying Graph Neural Network to Fraud Detection. SIGIR 2020.
[4] Derrow-Pinion, A., She, J., Wong, D., Lange, O., Hester, T., Perez, L., ... & Velickovic, P. (2021). ETA Prediction with Graph Neural Networks in Google Maps. CIKM 2021.
[5] Bongini, P., Bianchini, M., & Scarselli, F. (2021). Molecular Generative Graph Neural Networks for Drug Discovery. Neurpcomputing.
[6] Gama, F., Tolstaya, E., & Ribeiro, A. (2021). Graph Neural Networks for Decentralized Controllers. ICASSP 2021.

Regulations on Data Protection

- **Legislation**
  - Call for transparency and clarity on the usage of data
  - Empower user to remove his/her data from entities that store it → right to be forgotten
- **Examples**

GDPR

CCPA

PIPEDA

[1] GDPR: https://gdpr-info.eu/
[2] CCPA: https://oag.ca.gov/privacy/ccpa
[3] PIPEDA: https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/

Right to be Forgotten

- **What is it?**
  - The right to ask service provider to delete an individual's personal data
- **Why is data deletion not enough?**
  - GNNs might leak information of training data (membership inference attack)
  - Unintentional memorization the training data in neural networks

The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks

Nicholas Carlini[1,2] Chang Liu[2] Úlfar Erlingsson[1] Jernej Kos[3] Dawn Song[2]

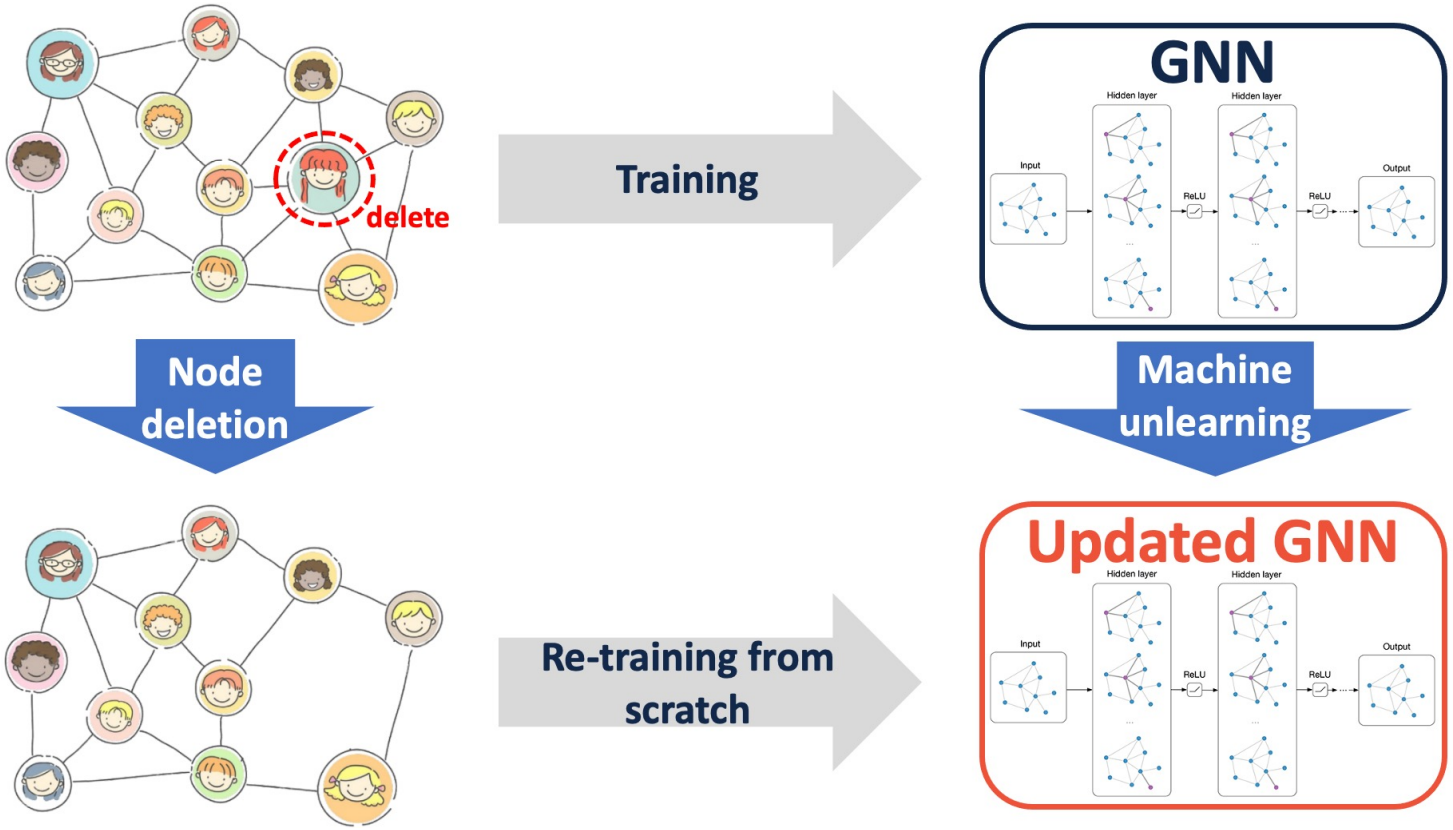
[1]*Google Brain* [2]*University of California, Berkeley* [3]*National University of Singapore*

- **Naïve solution:** re-training from scratch
  - High computational cost
  - Potential unavailability to retrieve all training data
- **Question:** how to remove the effect of a data without re-training GNNs?

[1] Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., & Song, D. (2019). The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. USENIX Security 2019.

Machine Unlearning on Graphs

- **Goal:** reproduce the GNNs after forgetting some training data without re-training from scratch
- **Challenges**
  - Efficient unlearning
  - Comparable model utility
  - Node dependency
    - Removing one node would affect the representation of other nodes



[1] Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., & Song, D. (2019). The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. USENIX Security 2019.

Roadmap

- Background
- GraphEraser
  - Published at CCS 2022

Graph Unlearning

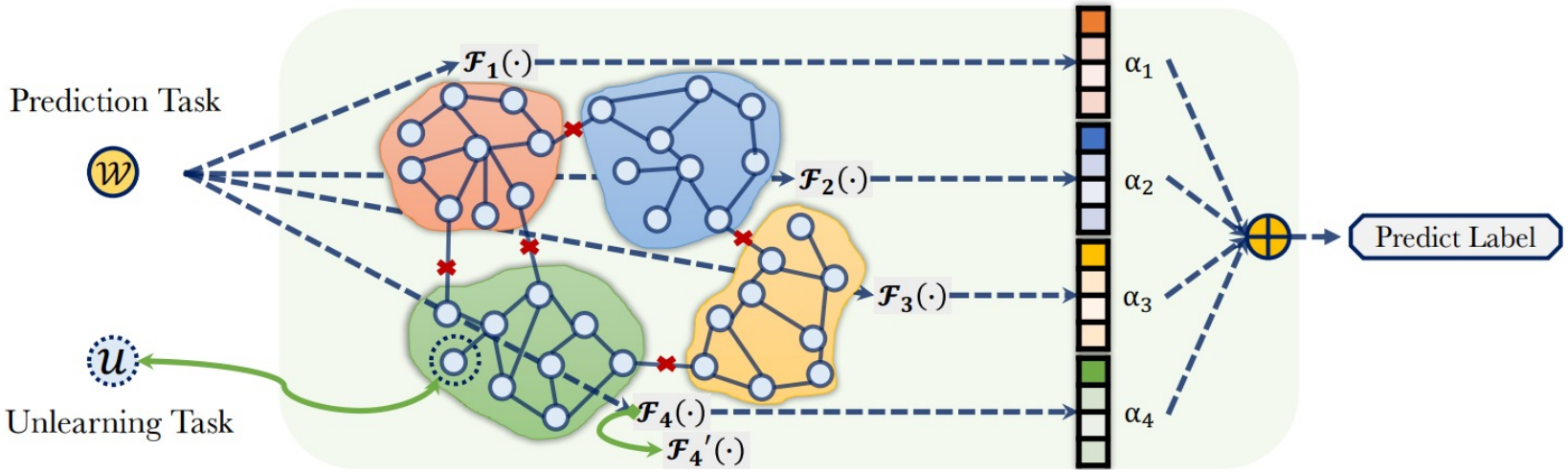
Min Chen[1*] Zhikun Zhang[1*] Tianhao Wang[2] Michael Backes[1]
Mathias Humbert[3] Yang Zhang[1]

[1]*CISPA Helmholtz Center for Information Security*
[2]*Purdue University* [3]*Cyber-Defence Campus, armasuisse S+T*

- Projector
- Possible future directions

GraphEraser

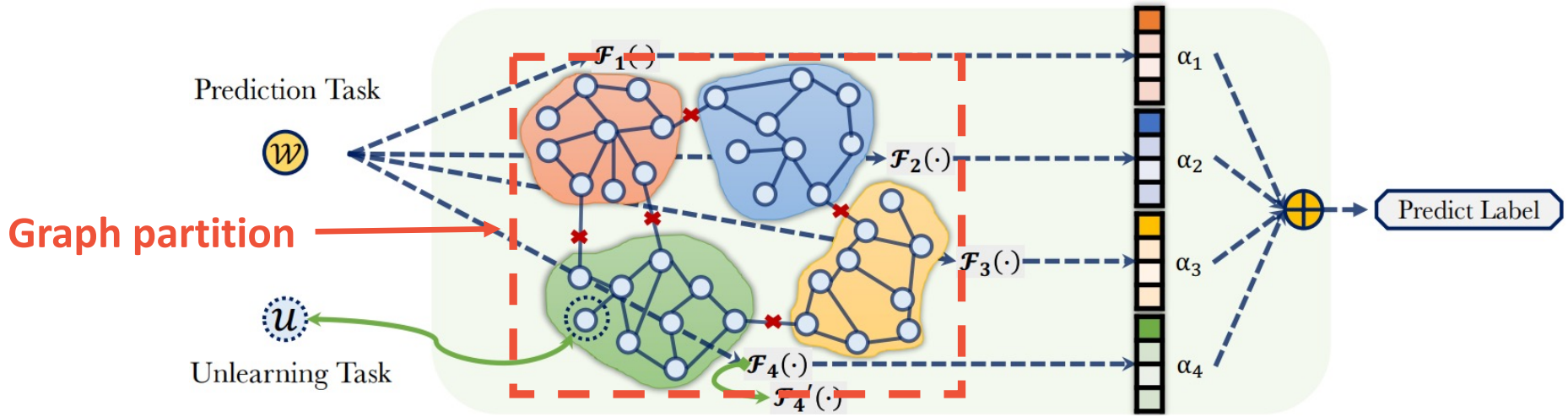
- **Observation:** computationally prohibitive to re-train a GNN on large graphs
- **Question:** can we re-train on a relatively smaller (sub-)graph?
- **Key idea:** Sharded, Isolated, Sliced, and Aggregated training (SISA)
  - **Step 1:** graph partition
  - **Step 2:** shard model training
  - **Step 3:** shard model aggregation



[1] Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C. A., Jia, H., Travers, A., Zhang, B., ... & Papernot, N. (2021). Machine Unlearning. SP 2021.
[2] Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., & Zhang, Y. (2022). Graph unlearning. ACM CCS 2022.

Step 1: Graph Partition

- **Goal:** split the graph into several clusters (shards) so that
  - Each shard should preserve the structural properties of the training graph
  - Clusters have similar size ← not satisfied by existing clustering methods (e.g., Louvain, Infomap)
- **Partition strategies**
  - **Strategy 1:** balanced label propagation (BLPA)
  - **Strategy 2:** balanced embedding k-means (BEKM)



[1] Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., & Zhang, Y. (2022). Graph unlearning. ACM CCS 2022.

Strategy 1: Balanced Label Propagation (BLPA)

- **Step 1:** initialization
  - Randomly assign each node to one of the k shard
- **Step 2:** reassignment calculation
  - Propagate a node's current cluster assignment C_{src} to its neighbors
  - Calculate its new cluster assignment C_{dst} = the majority cluster in its neighbors
  - Calculate a preference score ξ = #neighbors belonging to C_{dst}
- **Step 3:** preference sorting
  - Sort all nodes by the preference score ξ in descending order
- **Step 4:** label propagation
  - Enumerate all nodes based on the sorted list in step 3
  - If the size of destination cluster C_{dst} does not exceed a threshold δ, add the node into C_{dst} and remove it from the current shard

[1] Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., & Zhang, Y. (2022). Graph unlearning. ACM CCS 2022.

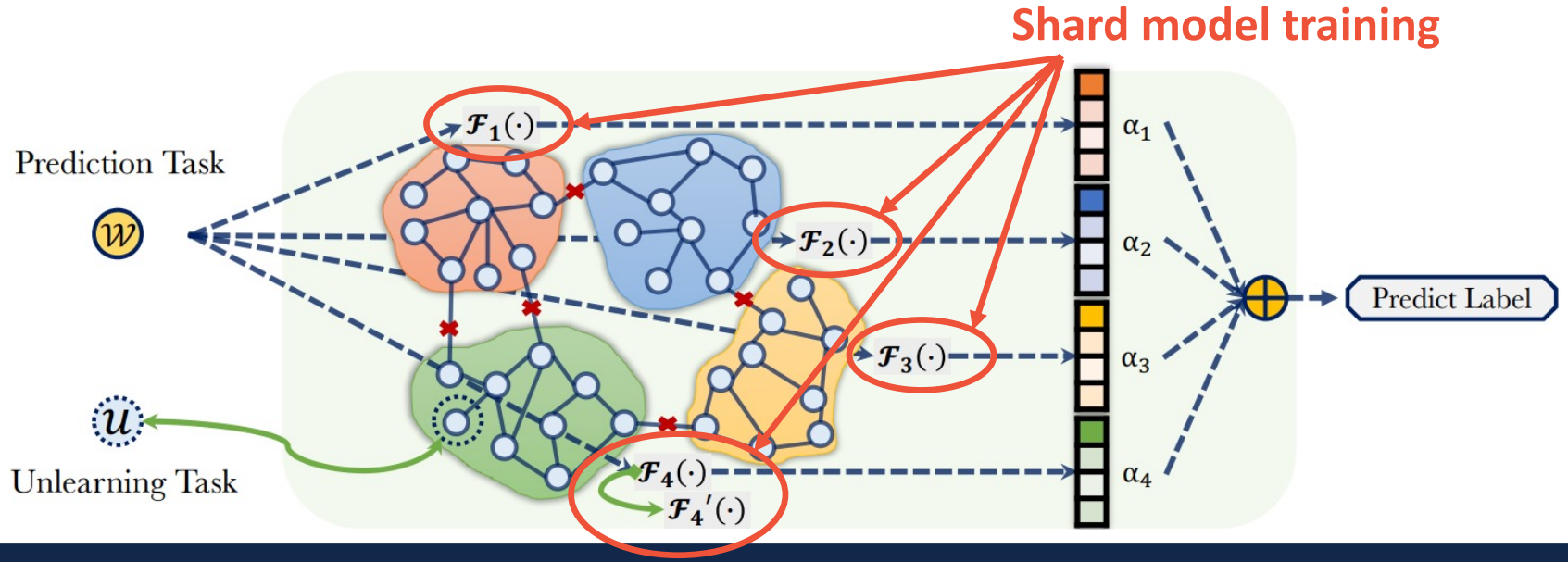
Strategy 2: Balanced Embedding K–Means (BEKM)

- **Step 1:** initialization
  - Extract node embeddings from a pre-trained GNN
  - Randomly select k centroids from the nodes
- **Step 2:** reassignment calculation
  - For each centroid, calculate a node's preference score ξ = Euclidean distance between its embedding and the centroid
- **Step 3:** preference sorting
  - Sort all (node, centroid) pairs by the preference score ξ in ascending order
- **Step 4:** label propagation
  - Enumerate all nodes based on the sorted list in step 3
  - If the size of a cluster does not exceed a threshold δ, add the node into C_{dst} and remove it from the current shard
  - Calculate new centroids as the mean of all nodes in the cluster

[1] Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., & Zhang, Y. (2022). Graph unlearning. ACM CCS 2022.

Step 2: Shard Model Training

- **Goals**
  - Train a shard model (i.e., a GNN) for each cluster
  - When receiving a deletion request, re-train the corresponding shard model
- **Remarks**
  - Each shard model should have the same architecture
  - Different shard models can be trained in parallel

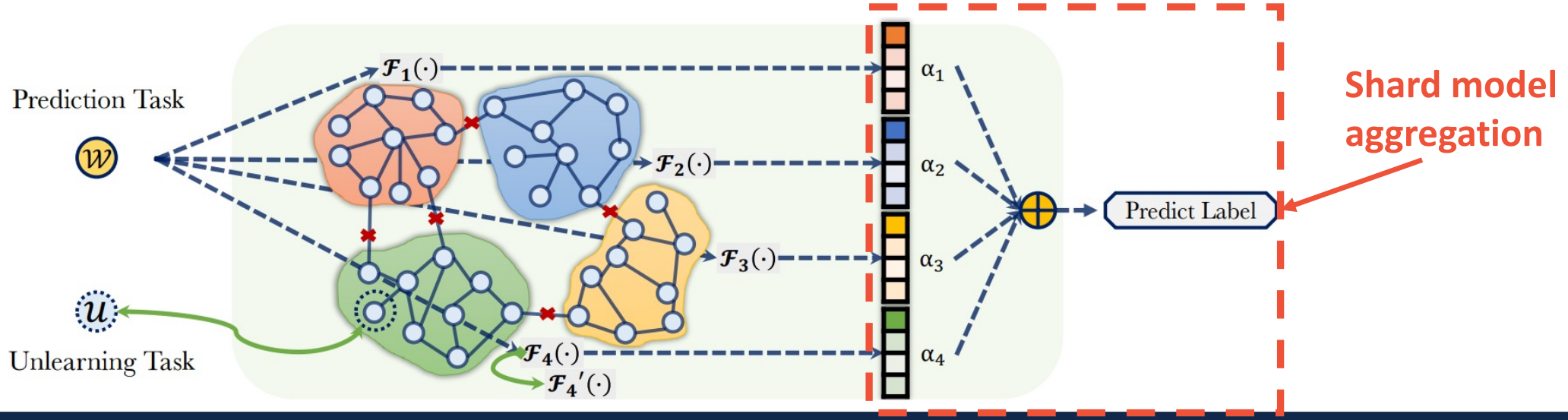


[1] Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., & Zhang, Y. (2022). Graph unlearning. ACM CCS 2022.

Step 3: Shard Model Aggregation

- **Goal:** aggregate predictions from all models as final prediction
- **OptAggr:** learning-based aggregation method
  - $\boldsymbol{\alpha}$: importance score of shard model; $\boldsymbol{\alpha}[i]$: importance score of i-th shard model
  - y_x: ground truth of node x; $\tilde{y}_{x,i}$: prediction of node x from i-th shard model
  - L: cross entropy; λ: regularization hyperparameter

$$\min_{\boldsymbol{\alpha}} \mathbb{E}_x \left[L\left(\sum_i \boldsymbol{\alpha}[i] \tilde{y}_{x,i}, y_x \right) \right] + \lambda \sum_i \|\boldsymbol{\alpha}[i]\|$$



[1] Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., & Zhang, Y. (2022). Graph unlearning. ACM CCS 2022.

GraphEraser: Effectiveness Results

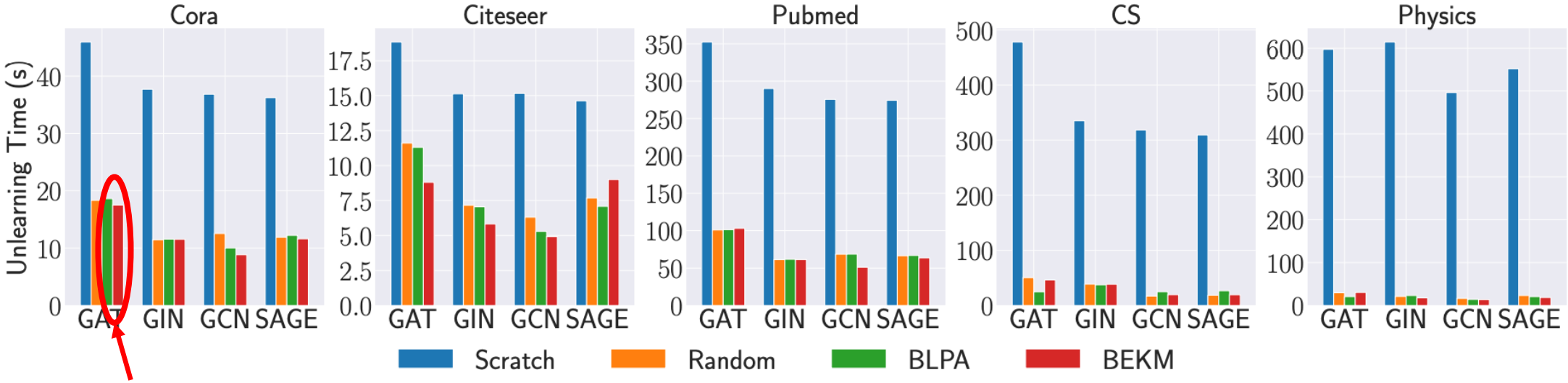
- **Tasks:** node classification
- **BLPA:** GraphEraser + balanced label propagation
- **BEKM:** GraphEraser + balanced embedding k-means
- **Observation:** GraphEraser (last two columns) achieves comparable classification accuracy

	Model	Scratch	Random	BLPA	BEKM
Cora	GAT	0.823 ± 0.006	0.726 ± 0.004	0.727 ± 0.009	**0.754 ± 0.009**
	GCN	0.739 ± 0.003	0.509 ± 0.009	**0.676 ± 0.004**	0.493 ± 0.006
	GIN	0.787 ± 0.013	0.766 ± 0.021	0.753 ± 0.015	**0.801 ± 0.018**
	SAGE	0.824 ± 0.004	0.682 ± 0.013	0.684 ± 0.014	**0.740 ± 0.013**
Citeseer	GAT	0.691 ± 0.015	0.631 ± 0.015	0.676 ± 0.004	**0.746 ± 0.006**
	GCN	0.493 ± 0.006	0.277 ± 0.009	**0.450 ± 0.006**	0.332 ± 0.006
	GIN	0.587 ± 0.031	0.626 ± 0.022	0.612 ± 0.026	**0.739 ± 0.020**
	SAGE	0.668 ± 0.013	0.623 ± 0.014	0.657 ± 0.012	**0.716 ± 0.007**
Pubmed	GAT	0.851 ± 0.004	0.857 ± 0.002	0.858 ± 0.003	**0.860 ± 0.003**
	GCN	0.748 ± 0.017	0.551 ± 0.005	**0.718 ± 0.010**	0.482 ± 0.003
	GIN	0.837 ± 0.015	0.856 ± 0.003	0.855 ± 0.004	**0.859 ± 0.003**
	SAGE	0.874 ± 0.003	0.857 ± 0.002	**0.863 ± 0.002**	0.862 ± 0.002
CS	GAT	0.919 ± 0.004	0.882 ± 0.000	0.858 ± 0.004	**0.886 ± 0.002**
	GCN	0.903 ± 0.006	0.706 ± 0.008	**0.750 ± 0.023**	0.671 ± 0.012
	GIN	0.867 ± 0.005	0.858 ± 0.005	0.789 ± 0.013	**0.861 ± 0.002**
	SAGE	0.932 ± 0.004	0.905 ± 0.004	0.886 ± 0.010	**0.913 ± 0.002**
Physics	GAT	0.955 ± 0.005	0.920 ± 0.002	0.921 ± 0.004	**0.925 ± 0.001**
	GCN	0.947 ± 0.002	0.747 ± 0.010	**0.858 ± 0.008**	0.750 ± 0.001
	GIN	0.934 ± 0.003	0.921 ± 0.002	0.907 ± 0.003	**0.926 ± 0.001**
	SAGE	0.956 ± 0.005	0.823 ± 0.011	0.922 ± 0.001	**0.933 ± 0.001**

[1] Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., & Zhang, Y. (2022). Graph unlearning. ACM CCS 2022.

GraphEraser: Efficiency Results

- **Tasks:** node classification
- **BLPA:** GraphEraser + balanced label propagation
- **BEKM:** GraphEraser + balanced embedding k-means
- **Observation:** GraphEraser (green bar + red bar) is much more efficient than re-training from scratch



Green bar + red bar = proposed methods

[1] Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., & Zhang, Y. (2022). Graph unlearning. ACM CCS 2022.

Roadmap

- Background
- GraphEraser
- Projector
  - In submission to NeurIPS 2022

Privacy Matters! Efficient Graph Representation Unlearning with Data Removal Guarantee

Weilin Cong
Penn State
weilin@psu.edu

Mehrdad Mahdavi
Penn State
mzm616@psu.edu

- Possible future directions

Recap: GraphEraser

- **Key idea:** Sharded, Isolated, Sliced, and Aggregated training (SISA)
  - **Step 1:** graph partition
  - **Step 2:** shard model training
  - **Step 3:** shard model aggregation
- **Limitations**
  - Hard to determine a good number of partitions
    - Too many clusters → lack of training data for each shard model, might hurt overall performance
    - Too few clusters → re-training on a relatively large subgraph, might be computationally expensive
  - Removal of cross-shard links → might hurt overall performance
- **Question:** how to unlearn a node without (re-)training multiple shard model and performance degradation?

[1] Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., & Zhang, Y. (2022). Graph unlearning. ACM CCS 2022.
[2] Cong, W., & Mahdavi, M. Privacy Matters! Efficient Graph Representation Unlearning with Data Removal Guarantee. In submission to NeurIPS 2022.

Projector: Problem Settings

- **Task:** binary node classification
- **Given**
  - Adjacency matrix $\mathbf{A}$
    - Propagation matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ with $\mathbf{D}$ as the degree matrix of $\mathbf{A}$
  - Node feature matrix $\mathbf{X}$
  - Node label vector $\mathbf{y}$
  - Weight vector $\mathbf{w}$
  - An L-layer linear GNN $\mathbf{H} = \mathbf{P}^L\mathbf{X}\mathbf{w}$
- **Minimize**

$$L = \frac{1}{|\mathcal{V}_{\text{train}}|} \sum_{i \in \mathcal{V}_{\text{train}}} \log(1 + \exp(-\mathbf{y}[i]\mathbf{w}\mathbf{H}[i,:])) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

[1] Cong, W., & Mahdavi, M. Privacy Matters! Efficient Graph Representation Unlearning with Data Removal Guarantee. In submission to NeurIPS 2022.

Projector: Warm Up

- **Proposition**
  - Suppose
    - Initial weight is in the linear span of node features $\mathbf{w}^{(0)} \in \text{span}\{\mathbf{X}[1,:], \dots, \mathbf{X}[n,:]\}$
    - Linear GNN is trained with mini-batch stochastic gradient descent (SGD)
  - The weight after T steps of mini-batch SGD is in the linear span of node features

$$\mathbf{w}^{(T)} \in \text{span}\{\mathbf{X}[1,:], \dots, \mathbf{X}[n,:]\}$$

- **Proof sketch**
  - Gradient is in the linear span of node features

$$\nabla L = \lambda w + \frac{1}{|\mathcal{V}_{\text{train}}|} \sum_{j \in \mathcal{V}_{\text{train}}} \alpha_j \mathbf{X}[j,:]$$

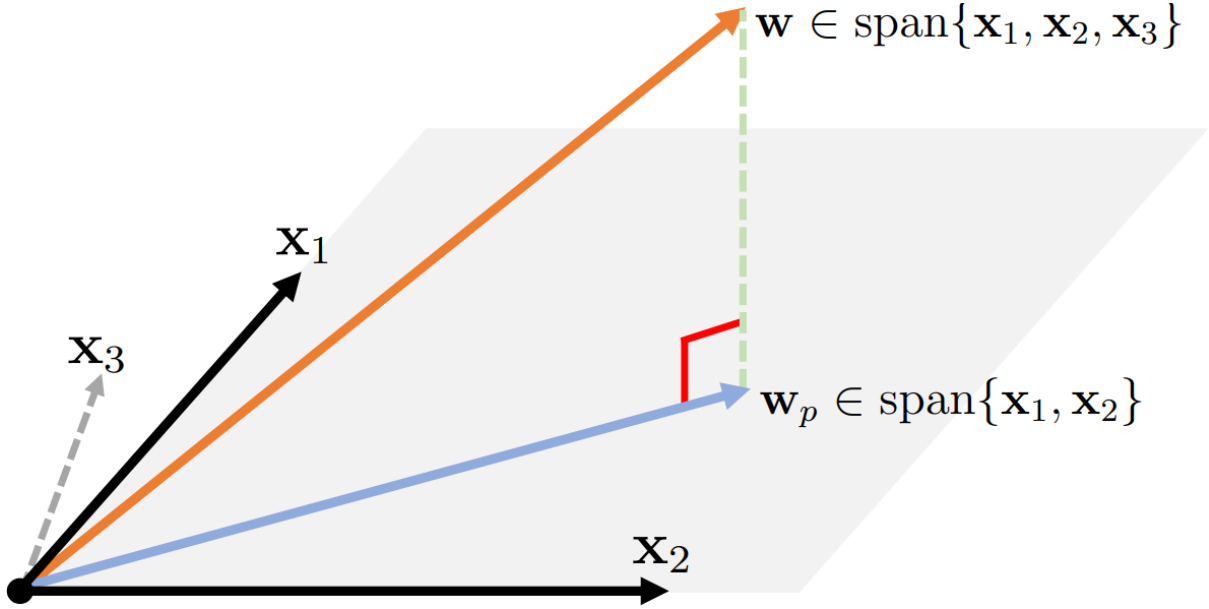
where $\alpha_j = \sum_{i \in \mathcal{V}_{\text{train}}} -\mathbf{y}[i]\text{sigmoid}(-\mathbf{y}[i]\mathbf{w}\mathbf{H}[i,:])\mathbf{P}^L[i,j]$

  - After SGD, the updated weights still stay in the linear span of node features

[1] Cong, W., & Mahdavi, M. Privacy Matters! Efficient Graph Representation Unlearning with Data Removal Guarantee. In submission to NeurIPS 2022.

Projector: Key Idea

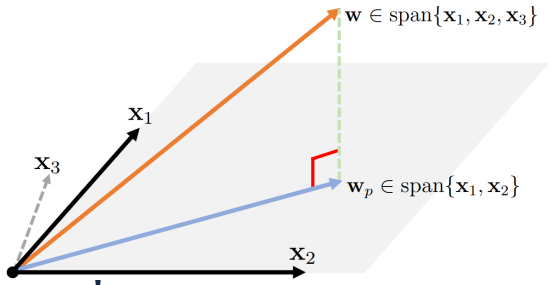
- **Proposition:** weights are in the linear span of node features
- **Key idea:** project the weights into a subspace that is spanned by the features without the deleted nodes



- **Question:** how to project the weights to the new subspace?

[1] Cong, W., & Mahdavi, M. Privacy Matters! Efficient Graph Representation Unlearning with Data Removal Guarantee. In submission to NeurIPS 2022.

Projector: Direction of Projection



- **Main result**

$$\mathbf{w}_p = \mathbf{X}_{\text{remain}}{}^T \mathbf{X}_{\text{remain}} \left(\mathbf{X}_{\text{remain}}{}^T \mathbf{X}_{\text{remain}} \right)^\dagger \mathbf{w}$$

  - $\mathbf{X}_{\text{remain}}$: features of remaining nodes
  - $\mathbf{w}$: original model weights
  - $\mathbf{w}_p$: new weights after projection

- **Proof sketch**
  - New subspace is the linear span of remaining node features $\mathbf{w}_p = \mathbf{X}_{\text{remain}}^T \boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ is the weight vector for linear combination
  - Remaining node features are orthogonal to the projection direction

$$\mathbf{X}_{\text{remain}} \left(\mathbf{w} - \mathbf{X}_{\text{remain}}^T \boldsymbol{\alpha} \right) = \mathbf{0} \Rightarrow \boldsymbol{\alpha} = \left(\mathbf{X}_{\text{remain}} \mathbf{X}_{\text{remain}}^T \right)^\dagger \mathbf{X}_{\text{remain}} \mathbf{w}$$

$$= \lim_{\epsilon \to 0} \left(\epsilon \mathbf{I} + \mathbf{X}_{\text{remain}} \mathbf{X}_{\text{remain}}^T \right)^\dagger \mathbf{X}_{\text{remain}} \mathbf{w}$$

  - Apply Woodbury identity

$$\left(\epsilon \mathbf{I} + \mathbf{X}_{\text{remain}} \mathbf{X}_{\text{remain}}^T \right)^\dagger \mathbf{X}_{\text{remain}} \mathbf{w} = \mathbf{X}_{\text{remain}} \left(\epsilon \mathbf{I} + \mathbf{X}_{\text{remain}}^T \mathbf{X}_{\text{remain}} \right)^\dagger \mathbf{w}$$

[1] Cong, W., & Mahdavi, M. Privacy Matters! Efficient Graph Representation Unlearning with Data Removal Guarantee. In submission to NeurIPS 2022.

Projector: Algorithm

- **Given**
  - Node feature matrix $\mathbf{X}$
  - The pre-trained weights of linear GNN $\mathbf{w}$
- **Option 1:** if we know the remaining node features $\mathbf{X}_{\text{remain}}$

$$\mathbf{M}_{\text{remain}} = \mathbf{X}_{\text{remain}}{}^{T}\mathbf{X}_{\text{remain}}$$

- **Option 2:** if we know the deleted node features $\mathbf{X}_{\text{delete}}$ and $\mathbf{M} = \mathbf{X}^{T}\mathbf{X}$

$$\mathbf{M}_{\text{remain}} = \mathbf{M} - \mathbf{X}_{\text{delete}}^{T}\mathbf{X}_{\text{delete}}$$

- The projected weights after unlearning is

$$\mathbf{w}_{p} = \mathbf{M}_{\text{remain}}\mathbf{M}_{\text{remain}}^{\dagger}\mathbf{w}$$

[1] Cong, W., & Mahdavi, M. Privacy Matters! Efficient Graph Representation Unlearning with Data Removal Guarantee. In submission to NeurIPS 2022.

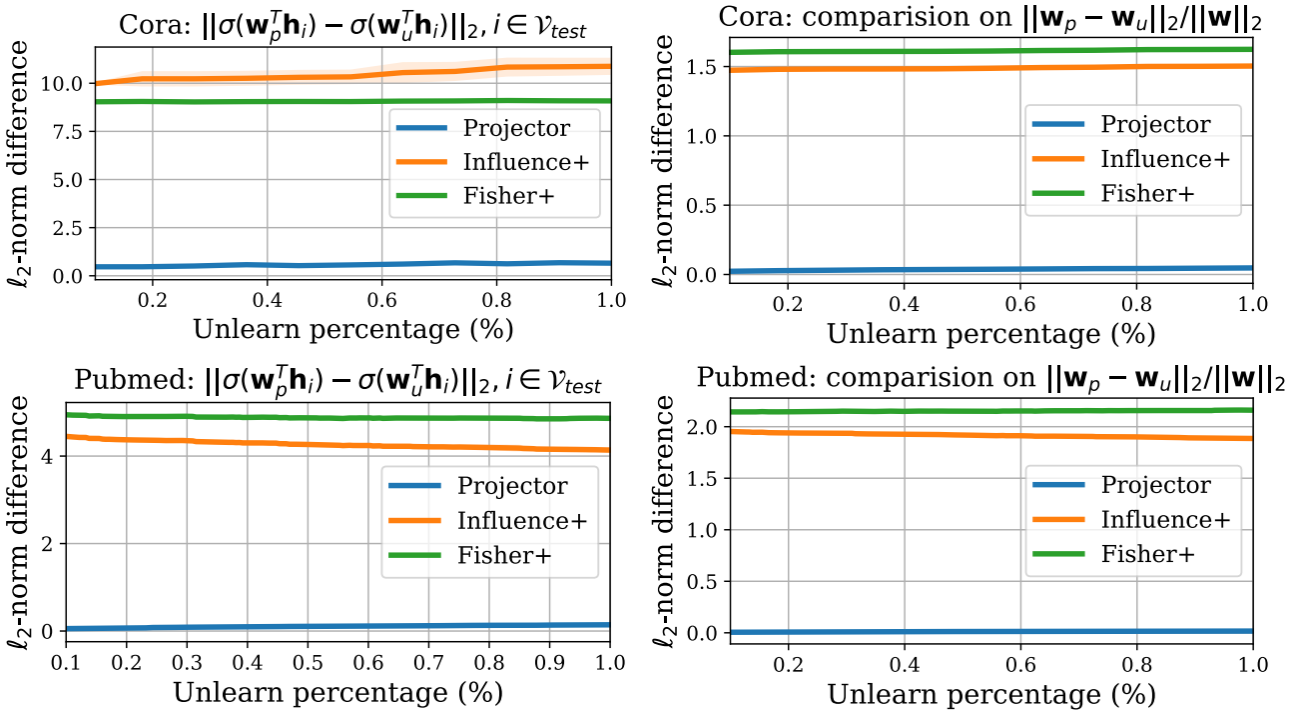
Projector: Effectiveness

- **Assumptions**
  - P_s: upper bound of the rows of propagation matrices before and after unlearning
  - P_d: upper bound of the difference between the propagation matrices before and after unlearning
  - B_x and B_w: upper bounds of the norm of node feature and weights, respectively
  - δ: error of approximating the node feature of any deleted node by linear combination of remaining nodes
  - G: upper bound of the variance of gradient on the remaining nodes
- The difference between the weights $\mathbf{w}_p$ from projector and the weights $\mathbf{w}_u$ by re-training from scratch are upper bounded by Δ
  - Δ can be mainly controlled by P_d, δ and G

[1] Cong, W., & Mahdavi, M. Privacy Matters! Efficient Graph Representation Unlearning with Data Removal Guarantee. In submission to NeurIPS 2022.

Projector: Experimental Results

- $\mathbf{w}_u$: weight parameters by re-training from scratch
- $\mathbf{w}_p$: weight parameters by running Projector
- **Observation:** Project has the closest performance w.r.t. re-training from scratch



[1] Cong, W., & Mahdavi, M. Privacy Matters! Efficient Graph Representation Unlearning with Data Removal Guarantee. In submission to NeurIPS 2022.

Roadmap

- Background
- GraphEraser
- Projector
- Possible future directions

Possible Future Directions

- **What to unlearn?**
  - **Existing works:** unlearn edge (GraphEraser) and node (GraphEraser, Projector, etc.)
  - **Question:** can we unlearn features and labels?
- **Who to unlearn?**
  - **Existing works:** unlearn a node based on user query
  - **Questions:** can we boost the model performance or debias model by unlearning certain edges/nodes in the graph?
    - Might be related to fast GNN model editing
- **How to unlearn?**
  - **Existing works:** either have poor performance or only work on linear GNNs
  - **Questions:** can we achieve unlearning on nonlinear GNNs?
    - If not, how to achieve better effective and/or efficiency in approximating the unlearned nonlinear GNNs?

References

- Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., & Zhang, Y. (2022). Graph unlearning. ACM CCS 2022.
- Cong, W., & Mahdavi, M. Privacy Matters! Efficient Graph Representation Unlearning with Data Removal Guarantee. In submission.