Efficient Learning of Linear Graph Neural Networks via Node Subsampling

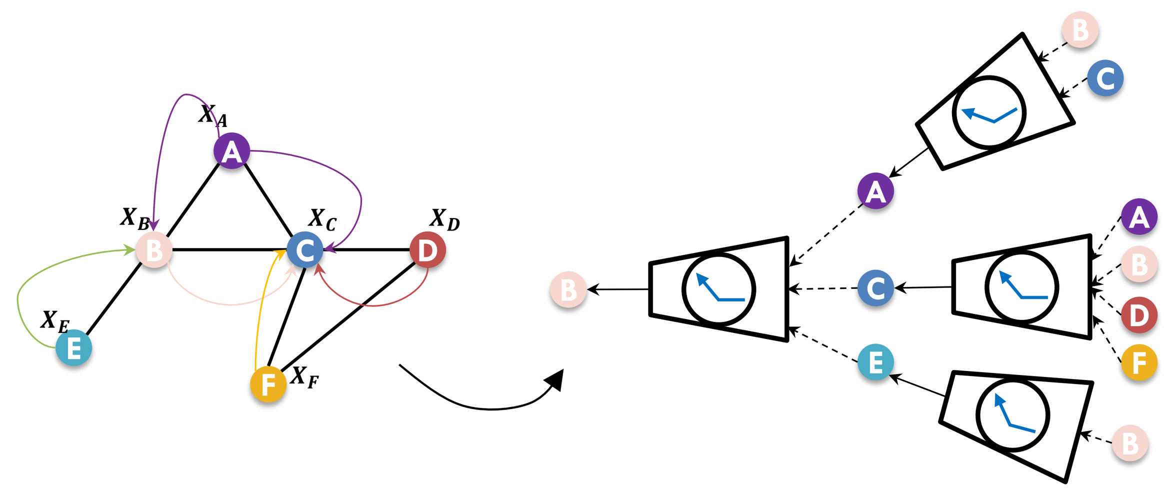
Seiyun Shin, Ilan Shomorony, and Han Zhao
University of Illinois at Urbana-Champaign

NEURAL INFORMATION PROCESSING SYSTEMS

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Graph Neural Networks (GNNs)

- Powerful models for graph-structured data



- Many applications:
  - Recommender systems, social network, drug discovery etc.

Being able to train GNNs *efficiently* is an important task!

Challenge: Large-scale Graphs in GNNs

- Graph Convolutional Networks (GCNs):
  Layer-wise propagation rule: For $A \in \mathbb{R}^{n \times n}$

$$H^{(\ell+1)} = \sigma(AH^{(\ell)}W^{(\ell)}), H^{(0)} = X \in \mathbb{R}^{n \times d}$$

- Storage cost:
  - Matrix A could be large and stored in distributed manner
  - Fully observing A may be costly or infeasible
- Computational cost:
  - The matrix multiplication AX requires $O(n^2 d)$ time
  - This can be prohibitive in big data settings

Can GNNs avoid quadratic complexity scaling with *n* via sampling?

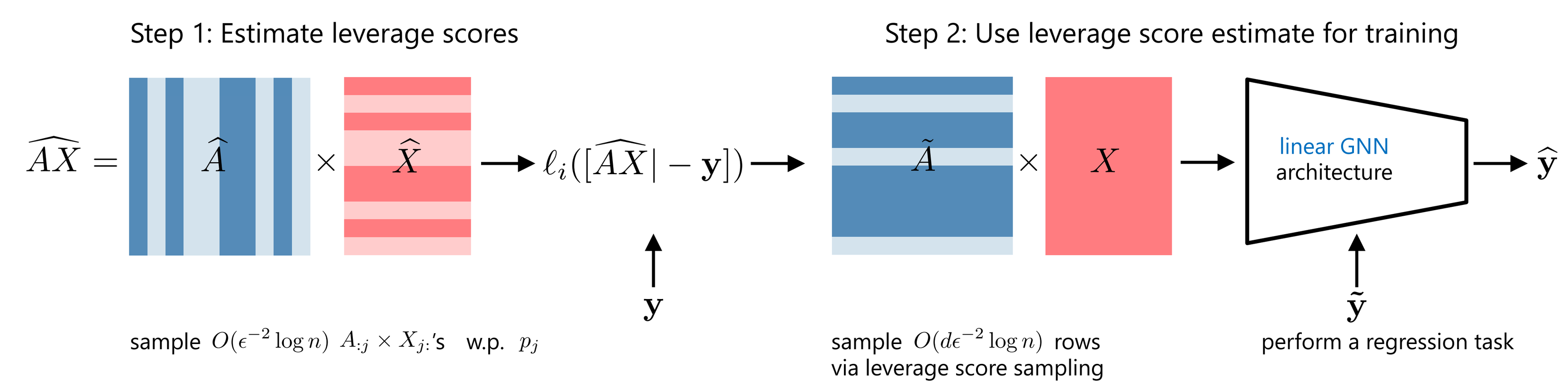
Questions:
- How many samples need to be observed?
- What graph subsampling strategies are amenable?

Two-stage Training Algorithm

For regression tasks, lev. score sampling has a nice guarantee

However, computing lev. scores requires the computation of AX

- Workaround:



Main Result

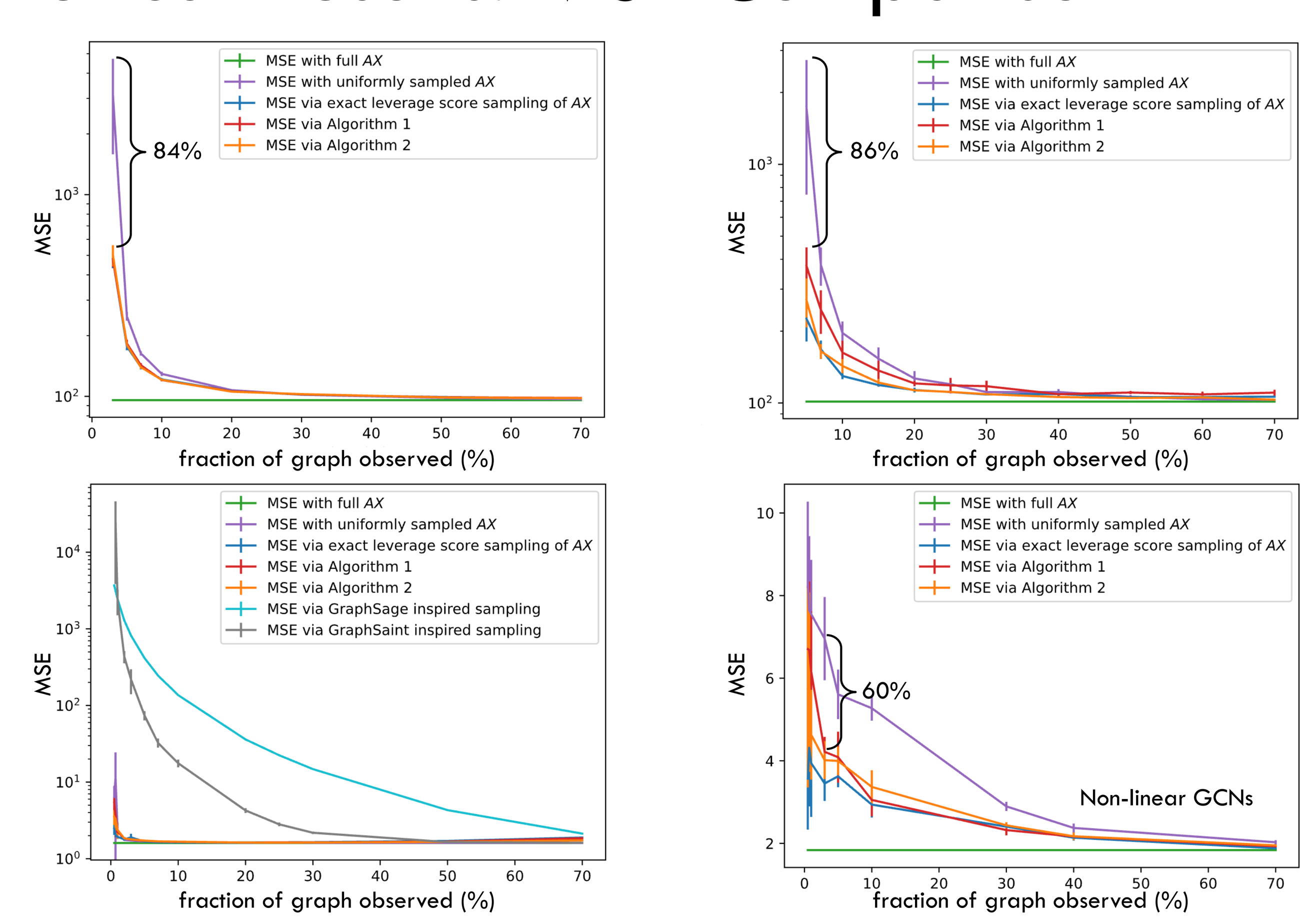
Theorem (informal): With $O\left(nd\epsilon^{-2}\log n\right)$ observations,

$$\|\tilde{\mathbf{y}} - \tilde{A}X\tilde{\mathbf{w}}\|_2^2 \le (1+\epsilon) \cdot \min_{\mathbf{w}} \|\mathbf{y} - AX\mathbf{w}\|_2^2$$

in time $O\left(nd^2\epsilon^{-2}\log n\right)$, with probability $1 - n^{-\Omega(1)}$.

- Implications: When $d \ll n$,
  - Speed-up: $nd^2\epsilon^{-2}\log n \ll n^2 d$
  - Query complexity gains: $nd\epsilon^{-2}\log n \ll n^2$

Numerical Results: MSE Comparison



Datasets: ogbl-ddi (from OGB), ego-Facebook (from SNAP), House dataset

Numerical Results: Run-time Comparison

Dataset	# Nodes	# Edges	# Features	Wall-clock time (sec)	
				full AX	Our scheme and lev. score sampling
ogbl-ddi	4.3K	1.3M	100	1.49	1.39
ogbn-arxiv	169.3K	1.2M	128	299.48	7.40
Synthetic data (Gaussian)	50.0K	625.0M	500	27.28	5.77
Synthetic data (Gaussian)	100.0K	2.5B	500	107/10	8.97
Synthetic data (Gaussian)	150.0K	5.6B	500	247.70	9.96

- Orders of maginitude less wall-clock time:
  - 40x acceleration on our scheme for ogbn-arxiv datasets

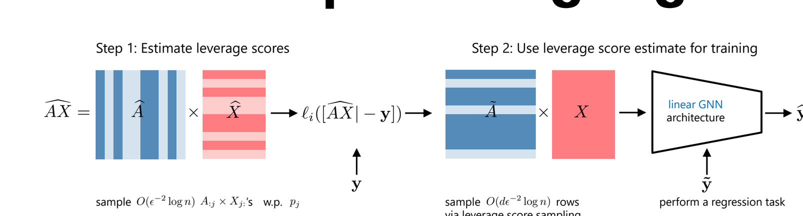
Numerical Results: Peak Memory Usage

Dataset	# Nodes	# Edges	# Features	Wall-clock time (sec)	
				full AX	Our scheme and lev. score sampling
ogbl-ddi	4.3K	1.3M	100	1.49	1.39
ogbn-arxiv	169.3K	1.2M	128	299.48	7.40
Synthetic data (Gaussian)	50.0K	625.0M	500	27.28	5.77
Synthetic data (Gaussian)	100.0K	2.5B	500	107/10	8.97
Synthetic data (Gaussian)	150.0K	5.6B	500	247.70	9.96

- Improvements on the memory usage:
  - In the best case, 1414x less memory requirement

Concluding Remark

- **Two-step training algorithms:**



Budget: $O\left(nd\epsilon^{-2}\log n\right)$
Run-time: $O\left(nd^2\epsilon^{-2}\log n\right)$
Error: $\|\tilde{\mathbf{y}} - \tilde{A}X\tilde{\mathbf{w}}\|_2^2 \le (1+\epsilon) \cdot \min_{\mathbf{w}} \|\mathbf{y} - AX\mathbf{w}\|_2^2$

- Extensions:
  - Nonlinear GCNs, classification or link prediction tasks?
- Generalization guarantee?
- Adaptive alglorithms at each gradient descent step?