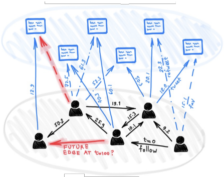
Learn Locally, Correct Globally: A Distributed Algorithm for Training Graph Neural Networks

Morteza Ramezani, Weilin Cong, Mehrdad Mahdavi, Mahmut Kandemir, Anand Sivasubramaniam

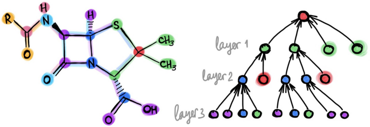
March 17, 2022

Motivation

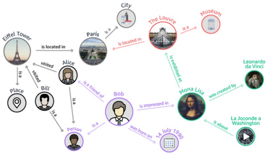
Graph neural networks (GNNs) has received massive attention and achieve great success in real-world applications.



Social networks



Drug discovery



Knowledge graph

Figure: Real-world applications of GNNs.

Motivation

Training GNNs on large graphs remains challenging, due to

- The limited resource (e.g., memory/computation power) of the existing servers
- The privacy concern due to the centralized storage and model learning

One potential solution to tackle these limitations is employing distributed training with data parallelism.

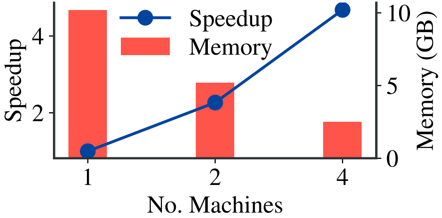


Figure: Comparison of the speedup and the memory consumption of distributed multi-machine training and centralized single machine training on the `Reddit` dataset.

Motivation

Main challenge. Employing distributed training on graph needs to partition graphs into subgraph, which results in edges spanning subgraphs (*cut-edges*)

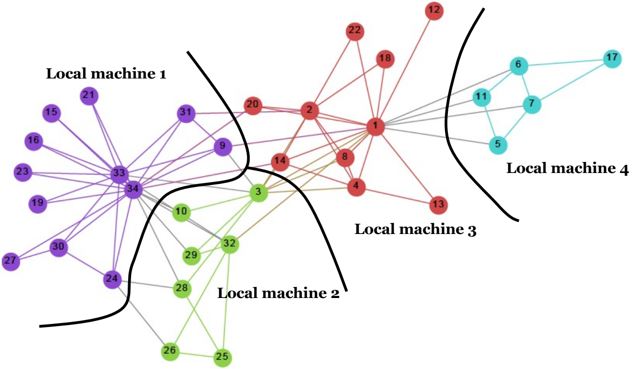


Figure: An illustration of distributed GNN training on Karate graph and cut-edges (edges that have nodes with different colors).

Motivation

Ignoring the cut-edges will hurt performance. Considering the cut-edges will results in high communication cost.

- *Parallel SGD with Periodic Averaging* (PSGD-PA): ignore cut-edges
- *Global Graph Sampling* (GGS): consider cut-edges

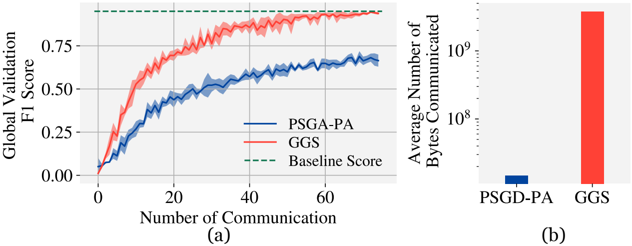


Figure: Comparison of (a) the validation F1-score and (b) the average data communicated per round (in bytes and log-scale) for two different distributed GNN training settings.

Method

To reduce the communication overhead, we propose Local Training with Periodic Averaging (i.e., PSGD-PA with carefully chosen #iters between local-server communication). Each local machine ...

- locally trains a GNN model by ignoring the cut-edges
- sends the trained model to the server for periodic model averaging
- receives the averaged model from server to continue the training

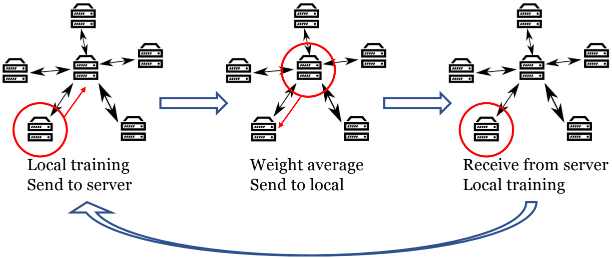


Figure: Local Training with Periodic Averaging.

Method

By doing so

- We eliminate the features exchange phase between server and local machines,
- BUT it can result in a significant performance degradation due to the lack of the global graph structure and the dependency between nodes among different machines.

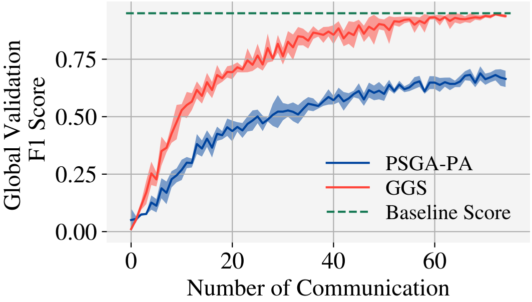


Figure: Ignore cut-edges will result in performance degradation.

Method

To compensate for this error, we propose a Global Server Correction scheme to

- take advantage of the available global graph structure on the server
- refine the averaged locally learned models before sending it back to each local machine.

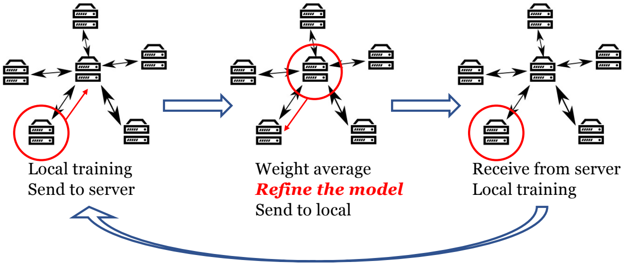


Figure: Our proposal: Local Learning Correct Globally.

Theoretical analysis

We provide the first theoretical analysis on the convergence of distributed training for GNNs with periodic averaging:

- We show that solely averaging the local machine models and ignoring the global graph structure will suffer from an irreducible residual error.

Theorem (Distributed GCN via Parameter Averaging)

Consider applying model averaging for GNN training under assumptions on stochasitc gradient variance. If we choose learning rate $\eta = \frac{\sqrt{P}}{\sqrt{T}}$ and the local step size $K \leq \frac{\sqrt{2}T^{1/4}}{8LP^{3/4}}$, then for any $T \geq L^2 P$ steps of gradient updates we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\boldsymbol{\theta}}^t)\|^2] = \mathcal{O}\left(\frac{1}{\sqrt{PT}}\right) + \mathcal{O}(\kappa^2 + \sigma_{bias}^2).$$

Theoretical analysis

- Then, we show that LLCG enjoys the convergence rate that matches the rate of FedAvg on a general (not specific for GNN training) non-convex optimization setting.

Theorem (Local Learning Correct Globally)

If we choose learning rate $\eta = \frac{\sqrt{P}}{\sqrt{T}}$, *the local step size* K, ρ *such that* $\sum_{r=1}^{R} K^2 \rho^{2r} \leq \frac{RT^{1/2}}{32L^2P^{3/2}}$, *and server correction step size* $S = \max_{r\in[R]} \left(\frac{\kappa^2+2\sigma_{bias}^2}{1-L(\sqrt{P/T})} - G_{local}^r\right)\frac{K\rho^r}{G_{local}^r}$, *then for any* $T \geq L^2P$ *steps of gradient updates we have:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\bar{\theta}^t)\|^2] = \mathcal{O}\Big(\frac{1}{\sqrt{PT}}\Big).$$

Experiments

Table: Comparison of performance and the average Megabytes of node representation/feature communicated per round on various datasets.

	Method	No. Comm.	GCN / SAGE Performance	GCN / SAGE Avg. MB	GAT Performance	GAT Avg. MB	APPNP Performance	APPNP Avg. MB
Flickr (F1-score)	PSGD-PA	50	$49.08_{\pm 0.27}$	12.57	$51.56_{\pm 0.28}$	4.24	$50.81_{\pm 0.48}$	8.40
	GGS		$51.22_{\pm 0.13}$	1849.32	$52.41_{\pm 0.29}$	1895.61	$51.33_{\pm 0.33}$	1897.82
	LLCG		$50.38_{\pm 0.20}$	12.57	$52.01_{\pm 0.33}$	4.24	$51.15_{\pm 0.25}$	8.40
OGB-Proteins (ROC-AUC)	PSGD-PA	100	$72.85_{\pm 0.70}$	6.20	$64.95_{\pm 1.01}$	3.14	$71.10_{\pm 0.79}$	7.31
	GGS		$74.78_{\pm 0.36}$	922.42	$68.11_{\pm 0.60}$	912.79	$71.29_{\pm 0.31}$	917.20
	LLCG		$73.92_{\pm 0.45}$	6.20	$67.62_{\pm 0.58}$	3.14	$71.18_{\pm 0.43}$	7.31
OGB-Arxiv (F1-score)	PSGD-PA	100	$69.43_{\pm 0.21}$	3.55	$69.88_{\pm 0.18}$	3.59	$68.48_{\pm 0.17}$	7.71
	GGS		$70.51_{\pm 0.26}$	3391.03	$70.82_{\pm 0.23}$	3396.79	$69.01_{\pm 0.10}$	3394.33
	LLCG		$70.21_{\pm 0.13}$	3.55	$70.58_{\pm 0.37}$	3.59	$68.73_{\pm 0.29}$	7.71
Reddit (F1-score)	PSGD-PA	75	$71.17_{\pm 1.06}$	14.83	$70.57_{\pm 1.24}$	7.48	$83.48_{\pm 0.81}$	11.63
	GGS		$94.77_{\pm 0.20}$	3798.81	$95.03_{\pm 0.48}$	3805.28	$95.23_{\pm 0.22}$	3770.46
	LLCG		$94.67_{\pm 0.15}$	14.83	$94.73_{\pm 0.23}$	7.48	$94.64_{\pm 0.17}$	11.63

Experiments

- Fully-sync vs LLCG: save accuracy but less time
- PSGD-PA vs LLCG: similar time but better accuracy

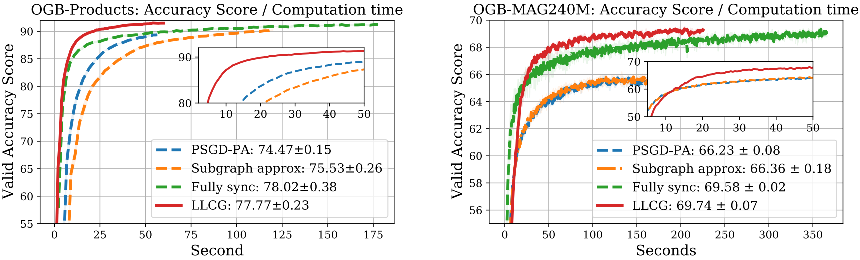


Figure: Compare validation accuracy and computation time.