Federated Graph Representation Learning using Self-Supervision

arXiv preprint arXiv:2210.15120 (2022)

Suresh, Susheel and Godbout, Danny and Mukherjee, Arko and Shrivastava, Mayank and Neville, Jennifer and Li, Pan

Reporter: Fengjiao Gong

February 16, 2023

Outline

1. Problem Setting
2. Novel Formulation
3. Method
4. Experiments

Problem Setting — Topic

Federated graph representation learning (FedGRL)

- **Federated learning (FL)**
  1. no sharing of raw data
  2. privacy-preserving model learning
  3. improve performance from others

Problem Setting — Topic

Federated graph representation learning (FedGRL)

- Federated learning (FL)
- **Graph representation learning (GRL)**
  1. node attributes
  2. structure information

Problem Setting — Challenges

Real-world graph data

- Label deficiency
- Downstream task heterogeneity

Problem Setting

Problem Setting for Clients

- Shared space of graph-structured data, though different distributions.
- Have access to vast amounts of unlabeled data.
- Different local downstream tasks with few private labeled data.

Novel Formulation — Notations

Consider graph $G = (V, E)$

- Node set $V = \{v_1, \ldots v_N\}$
- Edge set $E \subseteq V \times V$
- Node feature matrix $\boldsymbol{X} \in \mathbb{R}^{N \times F}$
- Adjacency matrix $\boldsymbol{A} \in \{0, 1\}^{N \times N}$

For graph encoder $g(G) = g(\boldsymbol{X}, \boldsymbol{A})$

- Node representations $\boldsymbol{H} \in \mathbb{R}^{N \times F'}, F' << F$

Novel Formulation — Notations

For client $c \in \mathcal{C}$

1. graph data $G_c = (V_c, E_c)$
   - node set $V_c = V_c^l \cup V_c^u$, labeled and unlabeled
   - data distribution $\mathcal{D}_c$
   - number of samples n_c
2. downstream task T_c
   - node label space $\mathcal{Y}_c$ with size m_c
   - infer labels for unlabeled nodes in V_c^u

Novel Formulation

Notes:

- Label scarcity
  1. small $\left|V_c^l\right|$
  2. much less unlabeled data $\sum_c \left|V_c^l\right| \ll \sum_c |V_c^u|$
  3. Poor data quality $\left|V_c^l\right| + |V_c^u|$ also small
- Downstream task heterogeneity
  1. different class label domain $\mathcal{Y}_c$

Novel Formulation — Federated Optimization

For client $c \in \mathcal{C}$

- Learn $f_c : \mathcal{X}_c \to \mathcal{Y}_c$
- Expected loss

$$\mathcal{L}_c\left(f_c\right) = \mathbb{E}_{(x_c, y_c) \sim \mathcal{D}_c}\left[\ell_c\left(f_c, x_c, y_c\right)\right]$$

Overall distributed optimization

$$\min_{\{f_c\}} \sum_{c \in \mathcal{C}} \frac{n_c}{n} \mathcal{L}_c\left(f_c\right)$$

where, $n = \cup_{c \in C} n_c$ is total amount of data.

Novel Formulation — General Formulation

Interpolated model $f_c = p_c \circ f$

- $p_c \Rightarrow$ client model <local>
- $f \Rightarrow$ global model <shared>

Distributed Objective

$$\min_{\{p_c\},f} \sum_{c \in C} \left[\mathcal{L}_c \left(p_c, f, G_c, Y_c \right) \mathbf{1}_{T_c \text{ exists}} + \lambda_c \tilde{\mathcal{L}}_c \left(f, G_c \right) \right] \tag{1}$$

where

- first term $\Rightarrow$ label-supervised objective
- second term $\Rightarrow$ self-supervised objective
- λ_c controls the amount of model interpolation

Novel Formulation — Learning protocols

Two steps:

1. *Federation* $<$ *ignore first term*, $\lambda_c = 1>$
   self-supervised learning to train f
2. *Client Local*
   specific task to train p_c by freezing or finetuning on top of f

Future work

1. Label-supervision $\Rightarrow$ both f and p_c
2. Self-supervision $\Rightarrow$ train f

Overview

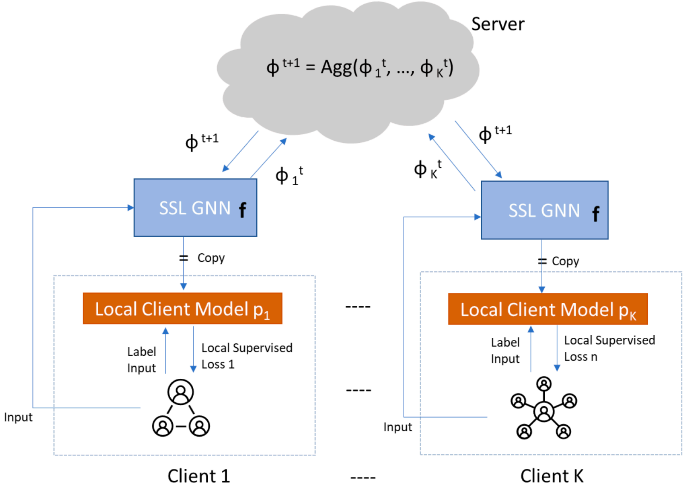

Figure 1: Overview of the proposed general formulation for self-supervised FedGRL. Here, we learn a shared global GNN model f using a self-supervised (SSL) objective, collaboratively based on federated learning. With the copy gate, f is only trained on unlabeled data through SSL and local client models p_c are further trained individually on top of f with label supervision.

Method

Distributed Objective

$$\min_{\{p_c\},f} \sum_{c\in C} \left[\mathcal{L}_c \left(p_c, f, G_c, Y_c \right) \mathbf{1}_{T_c \text{ exists}} + \lambda_c \tilde{\mathcal{L}}_c \left(f, G_c \right) \right]$$

Next

- **Self-Supervised Objective**
- Task Specific Supervision

Method — Self-Supervised Objective

Bootstrapped Graph Latents (BGRL)

- a graph representation learning method that learns by predicting alternative augmentations of the input.
- a scalable and SoTA for node level self-supervised learning of GNNs
- it does not require contrasting negative node pairs

[reference] Bootstrapped representation learning on graphs. In ICLR 2021 Workshop on Geometrical and Topological Representation Learning, 2021.

Self-Supervised Objective — BGRL

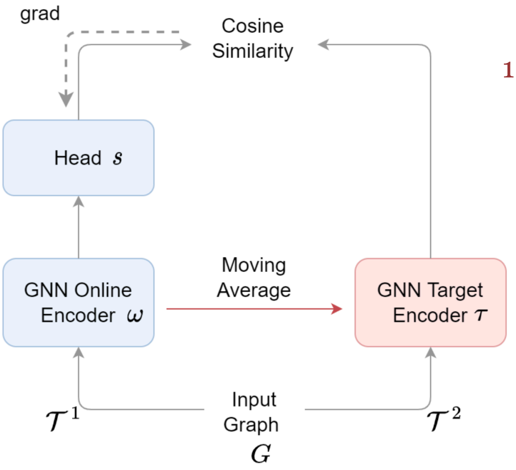

Figure 2: Overview of BGRL

1. produce two alternate views of G

$$G^1 = \mathcal{T}^1(G), \text{ and } G^2 = \mathcal{T}^2(G)$$

Graph Augmentations

- Node feature masking
- Edge masking

[reference] Deep Graph Contrastive Representation Learning. arXiv, 2020.

Self-Supervised Objective — BGRL

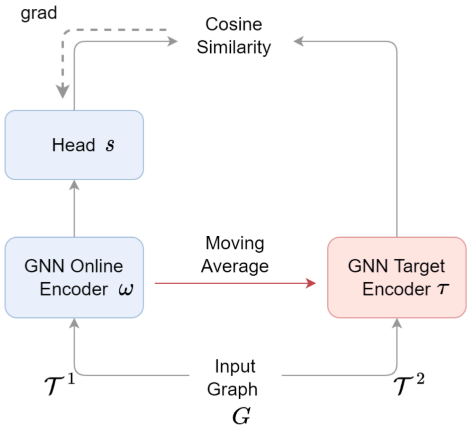

Figure 3: Overview of BGRL

1. produce two alternate views of G

$$G^1 = \mathcal{T}^1(G), \text{ and } G^2 = \mathcal{T}^2(G)$$

2. encode with two GNN encoders

$$\boldsymbol{H}^1 = \omega\left(G^1\right), \text{ and } \boldsymbol{H}^2 = \tau\left(G^2\right)$$

3. feed $\boldsymbol{H}^1$ to a node-level head

$$\boldsymbol{Z}^1 = s\left(\boldsymbol{H}^1\right)$$

4. predict $\boldsymbol{H}^2$ with $\boldsymbol{Z}^1$

$$\tilde{\mathcal{L}}(\omega, s, \tau) = -\frac{2}{N}\sum_{i=0}^{N-1} \frac{Z_i^1 {H_i^2}^\top}{\left\|Z_i^1\right\| \left\|H_i^2\right\|}$$

Self-Supervised Objective — BGRL

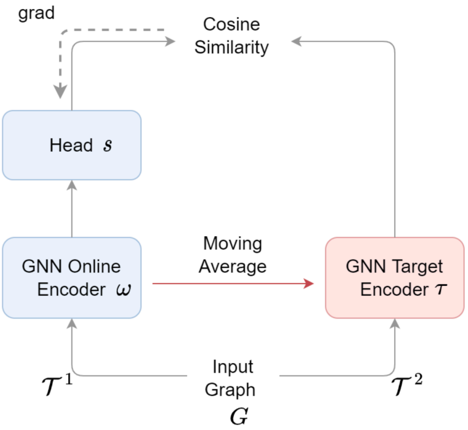

Figure 4: Overview of BGRL

Update parameters

$$\tilde{\mathcal{L}}(\omega_\theta, s_\phi, \tau_\rho) = -\frac{2}{N}\sum_{i=0}^{N-1} \frac{Z_i^1 {H_i^2}^\top}{\|Z_i^1\| \, \|H_i^2\|}$$

1. gradient descent only w.r.t θ, ϕ
2. update ρ by exponential moving average

$$\rho = \epsilon\rho + (1-\epsilon)\theta$$

where ϵ is the decay rate.

Self-Supervised Objective — BGRL

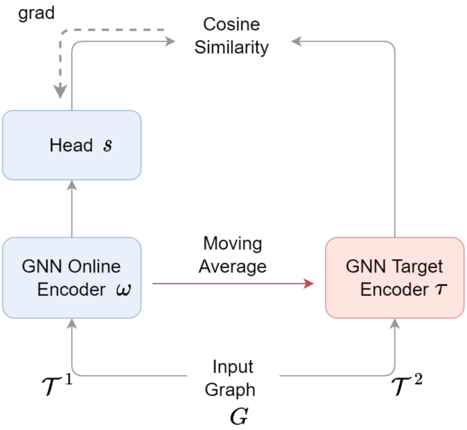

Figure 5: Overview of BGRL

During federation

1. θ, ϕ are shared across clients
2. server aggregates them in each round
3. global shared model is $f = \omega$

Method

Distributed Objective

$$\min_{\{p_c\},f} \sum_{c \in C} \left[\mathcal{L}_c\left(p_c, f, G_c, Y_c\right) \mathbf{1}_{T_c \text{ exists}} + \lambda_c \tilde{\mathcal{L}}_c\left(f, G_c\right) \right]$$

Next

- Self-Supervised Objective
- **Task Specific Supervision**

Method - Task Specific Supervision

Supervision from the node labels if available

- Predict value of G_c

$$Z_c = \text{softmax}\left(p_c \circ f\left(G_c\right)\right), \text{ and } Z_c \in \mathbb{R}^{N_c \times m_c}$$

- Cross-entropy loss

$$\mathcal{L}\left(p_c, f, G_c, Y_c\right) = -\sum_{i \in V_c^l} \sum_{j=1}^{m_c} Y_{c(ij)} \ln Z_{c(ij)}$$

Experiments — Datasets

1. Twitch Gamer Networks
   - nodes ⇒ users, edges ⇒ friendships
   - same feature space
   - a binary node classification task ⇒ to predict if a user uses explicit language

Table 1: Statistics of Twitch Gamer Networks.

	twitch-DE	twitch-EN	twitch-ES	twitch-FR	twitch-PT	twitch-RU
#nodes	9,498	7,126	4,648	6,549	1,912	4,385
#edges	153,138	35,324	59,382	112,666	31,299	37,304
density	0.003	0.002	0.006	0.005	0.017	0.004

Experiments — Datasets

2 Amazon Co-purchase Networks

- nodes $\Rightarrow$ products, edges $\Rightarrow$ co-purchase (“also buy”)
- same feature space
- a node classification task $\Rightarrow$ to classify the fine-grained sub-segments
- different class label domains

Table 2: Statistics of Amazon Co-purchase Networks.

	computer	photo	phone	tool	guitar	art
#nodes	10,055	4,705	16,683	4,827	2,506	6,610
#edges	87,512	28,818	113,760	43,458	10,342	90,678
#classes	9	8	7	6	5	8

Experiments — Baselines

Metric:

$$\text{F1-Micro Score} = \frac{TP}{TP + 0.5 * (FP + FN)}$$

Baselines	**Supervision**	**Methods**
No-Fed-Sup	label	train a GNN model for each client individually
No-Fed-Self-Freeze	self	first train a GNN model then freeze the node representations perform linear evaluation
No-Fed-Self-Finetune	self	first train a GNN model perform fine-tuning with a MLP task head on top
Fed-Sup	label	trains a local GNN encoder + local MLP task head Share local GNN models across clients Server aggregates via FedAvg

Experiments — Results

Table 3: Node classification task on Twitch Gamer Networks.

	twitch-DE	twitch-EN	twitch-ES	twitch-FR	twitch-PT	twitch-RU
No-Fed-Rand-Init-GNN	0.502 ± 0.103	0.501 ± 0.047	0.496 ± 0.198	0.510 ± 0.131	0.533 ± 0.156	0.507 ± 0.248
No-Fed-Sup	0.673 ± 0.010	0.584 ± 0.014	0.725 ± 0.014	0.626 ± 0.016	0.678 ± 0.023	0.751 ± 0.015
No-Fed-Self-Freeze	0.685 ± 0.009	0.617 ± 0.012	0.731 ± 0.010	0.626 ± 0.015	0.696 ± 0.021	0.752 ± 0.009
No-Fed-Self-Finetune	0.703 ± 0.012	**0.665** ± **0.023**	0.746 ± 0.015	0.635 ± 0.018	0.710 ± 0.026	**0.762** ± **0.011**
Fed-Sup	0.677 ± 0.009	0.600 ± 0.009	0.723 ± 0.013	0.627 ± 0.018	0.683 ± 0.013	0.743 ± 0.015
Fed-Self-Freeze (ours)	0.686 ± 0.007	0.606 ± 0.012	0.733 ± 0.007	0.626 ± 0.014	0.699 ± 0.022	0.752 ± 0.009
Fed-Self-Finetune (ours)	**0.706** ± **0.013**	0.657 ± 0.024	**0.745** ± **0.013**	**0.636** ± **0.021**	**0.712** ± **0.024**	0.761 ± 0.014

60/20/20 training, validation and test node random split

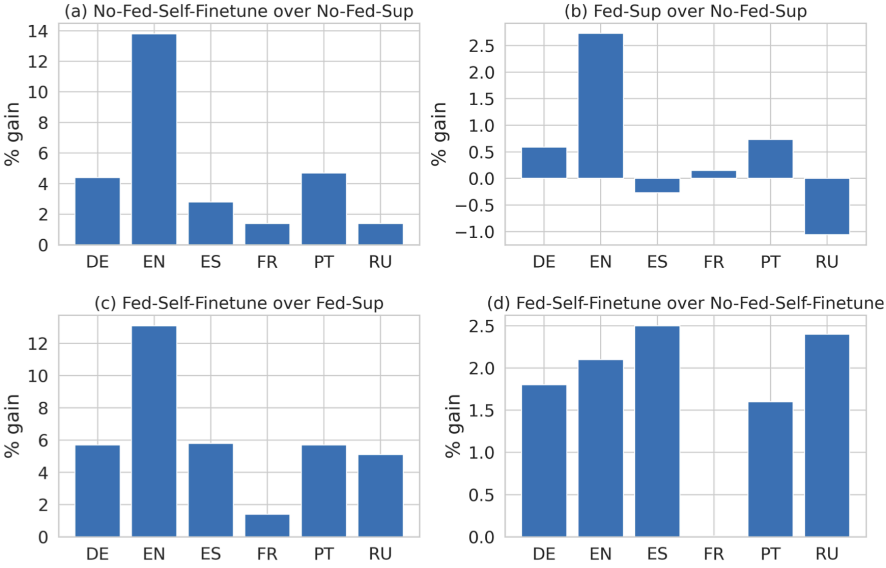

Figure 6: Performance gains in Twitch network experiments

Experiments — Results

Table 4: Node classification task results on Amazon Co-purchase Networks.

	computer	photo	phone	guitar	tool	art
No-Fed-Rand-Init-GNN	0.687 ± 0.005	0.709 ± 0.008	0.673 ± 0.005	0.737 ± 0.014	0.622 ± 0.008	0.653 ± 0.009
No-Fed-Sup	0.787 ± 0.009	0.801 ± 0.008	0.759 ± 0.007	0.866 ± 0.012	0.764 ± 0.012	0.741 ± 0.010
No-Fed-Self-Freeze	0.837 ± 0.004	0.841 ± 0.007	**0.778** ± **0.003**	0.882 ± 0.011	0.800 ± 0.007	0.784 ± 0.006
No-Fed-Self-Finetune	0.789 ± 0.012	0.792 ± 0.014	0.757 ± 0.012	0.845 ± 0.026	0.758 ± 0.018	0.735 ± 0.015
Fed-Sup	0.769 ± 0.003	0.754 ± 0.009	0.741 ± 0.004	0.807 ± 0.019	0.722 ± 0.010	0.704 ± 0.013
Fed-Self-Freeze (ours)	**0.849** ± **0.005**	**0.868** ± **0.009**	0.776 ± 0.005	**0.897** ± **0.011**	**0.818** ± **0.010**	**0.807** ± **0.005**
Fed-Self-Finetune (ours)	0.786 ± 0.009	0.791 ± 0.012	0.753 ± 0.005	0.849 ± 0.019	0.764 ± 0.016	0.738 ± 0.015

60/20/20 training, validation and test node random split

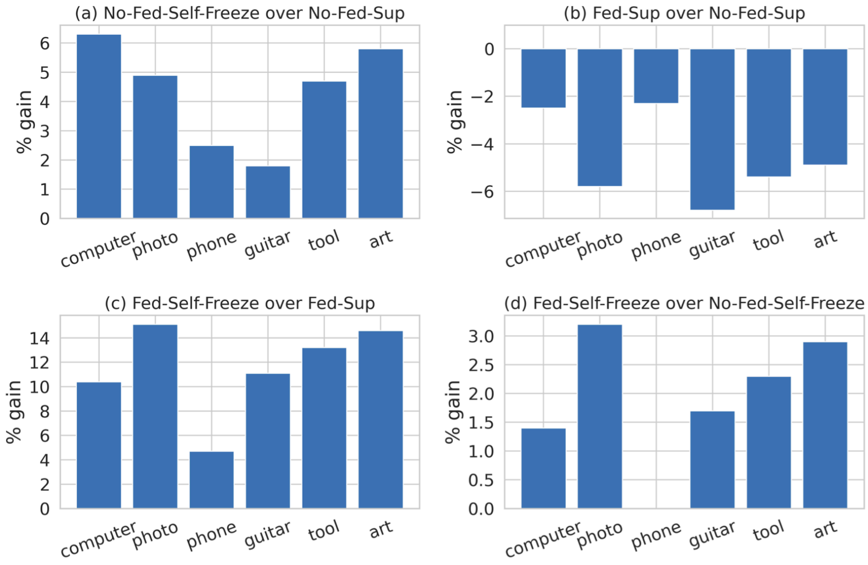

Figure 7: Performance gains in Amazon network experiments

Thanks!