Federated Graph Link Prediction Implementation

DU Jiaxing 1155204280

April 2024

1 Introduction

Graph Neural Networks (GNNs) are highly effective models for analyzing and learning from graph-structured data. Their ability to represent intricate connections and dependencies between nodes and edges in a graph makes them appropriate for a multitude of applications, including node, graph, and edge-level classification and regression tasks. However, traditional GNNs require centralized graph-structured data, which raises privacy concerns and practical limitations when dealing with sensitive data sources, such as bank transaction systems, healthcare industries, and social networks. The motivation of Federated Graph Neural Networks (FedGNNs) is to combine the advantages of GNNs and federated learning. By training GNN models in a federated manner, local knowledge from different graph sources can be aggregated, leading to a more comprehensive and generalizable global model while each party holds its data locally for training.

In this report, we focus on some real work applications of federated learning on graph-structure data. First, We will briefly review previous works on different FedGNN tasks in section 2. In section 3 and 4, we will discuss how we apply FedGNN in two scenarios: movie recommendation system and IBM Anti-money laundering (AML) detection. The motivation and model design are included. In section 5, we will present the training results on several evaluation metrics and compare them with some competitive methods. The summary of our findings and the significance of this project are in section 6. We maintain the source code at `https://github.com/Worshipper6/Industrial-project`.

2 Background

The incredible development of federated learning (FL) has benefited various tasks in traditional machine learning and deep learning, and the existing frameworks such as FedML (He et al., 2020), FATE (Liu et al., 2022) and FedScale (Lai et al., 2022) have made the deployment easy in real-world applications. In general, federated learning has two main settings: horizontal FL and vertical FL (Yang et al., 2019). In horizontal FL, datasets of different parties have large overlaps in features but small overlaps in samples. In contrast, datasets in vertical FL have large overlaps in samples but small overlaps in features. Privacy protection techniques, including MPC, Differential Privacy (Dwork, 2008), and Homomorphic Encryption (Rivest et al., 1978), are widely adopted for encrypting transmission. For example, in horizontal FL, we can add some noise to the gradients or parameters of local clients before passing them to the central server. In vertical FL, homomorphic encryption applies to encrypt training loss and gradients from different parties collaboratively. However, the process usually involves a trade-off between accuracy and privacy, so balancing the risk of privacy leakage and model performance is worth considering.

With the prevalence of GNNs in learning distributed representations from graph-structured data, He et al. (2021) first introduced FedGraphNN, an open-source FL framework supporting the unique characteristics and requirements of graph-structured data. They organize various distributed graph scenarios into three settings: graph, subgraph and node-level FL, corresponding to three kinds of tasks in GNNs. All these FedGNNs are realized in Message Passing Neural Network (MPNN) framework. After each clients finish local training, the central server aggregates model parameters or gradients and pass back to clients. The following steps are performed iteratively to gather information in the graph.

For client k and the layer index l, node v aggregate all meassages from its neighbors $\mathcal{N}(v)$ and feed forward to produce a new state

$$\text{Aggregate neighbors' info}: a^{(l)}(v) = \text{AGGREGATE}\left(\{h^{(l-1)}(u) | u \in \mathcal{N}(v)\}\right)$$

$$\text{Feed forward}: \quad h^{(l)}(v) = \text{UPDATE}\left(h^{(l-1)}(v), a^{(l)}(v)\right)$$

We can use FedAvg (McMahan et al., 2017) to aggregate the training results of each client. Assume $\mathbf{W}^{(i)}$ is a combination of learnable weights in GNN of client k, t is current communication round. The average function on the server averages the parameters of all clients at time t

$$\text{Local backward propagation}: \mathbf{W}_t^{(i)} \leftarrow \mathbf{W}_t^{(i)} - \lambda \nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W}, b)$$

$$\text{Global update parameter}: \quad \mathbf{W}_{t+1} \leftarrow \sum_{i=1}^{k} \frac{N_i}{N} \mathbf{W}_t^{(i)}$$

In terms of three kinds of graph mining tasks, we focus on subgraph-level link prediction with the most widely used FL algorithm of FedAvg, for its wide application in financial industries.

3 Recommendation System

3.1 Data Sources

- **Epinions**: The dataset is collected from consumer review platform `Epinions.com`, which is a widely used benchmark in the field of social network analysis and recommendation systems research (Tang et al., 2012). It contains 53717 user-generated reviews and ratings for various products, along with the social network connections between the users. The rating scale in the dataset typically ranges from 1 to 5, where 1 indicates the lowest rating and 5 indicates the highest rating. Each product has its own categories, such as Electronics, Kids & Family, Restaurants.
- **MovieLens Latest**: The dataset contains 100836 movie ratings and associated information provided by 610 users, available at `https://files.grouplens.org/datasets/movielens/ml-latest-small-README.html`. It is commonly used for developing and evaluating recommendation algorithms and techniques. Users rate movies on a numerical scale, typically ranging from 1 to 5, to indicate their preference or opinion. In addition, the dataset also includes other details about the movies themselves, such as movie names, genres, release years.

3.2 Models

We adopt GCN (Kipf & Welling, 2016), GAT (Schlichtkrull et al., 2018), GraphSage (Hamilton et al., 2017) to do link regression on the user ratings. The standard practice is to employ pairwise

node embedding loss functions, based on a encoding-decoding framework (Berg et al., 2017). First, we use GNN to map node pairs $(\mathbf{z}_u, \mathbf{z}_v) \in \mathcal{D}$ in the graph to a vector space. Then we use pairwise decoders, usually an inner product, to capture relationship or similarity between pairs of nodes.

$$\text{ENC} : \mathbf{Z} = GNN(\mathcal{V}, \mathcal{E}), \quad \text{DEC} : \hat{y}_i = DEC(\mathbf{z}_u, \mathbf{z}_v)$$

where $\mathcal{V}$, $\mathcal{E}$ denote nodes and edges, $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times d}$ contains the embedding of all nodes
Next, implement a loss function

$$\mathcal{L} = \sum_{(u,v) \in \mathcal{D}} loss(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_v, y_i)$$

which can be mean square error for regression and cross entropy for classification

Federated learning is a potential way to learn GNN models from the local decentralized data on a large number of user clients in a privacy-preserving way. We assume there exists privacy concerns in the user-item prediction. In Epinions dataset, each item category act as a party, in total 28 clients for training. They hold original user-item data locally and do not share with other categories. The local devices learn parameters of GNN and user-item embeddings and send them to a central server, which aggregates the parameters from selected users and distributes them back for further local updates.

In MovieLens dataset, we treat each user as a party, in total 610 clients. Each user device only contains first-order interaction information, making it difficult to capture inter-user information. We follow the setting of Wu et al. (2021) to expand local user-item graphs. The neighbors with co-interacted items with each user will be passed by the central server, under the assumption that users prefer to trust people who rate the same movie. This process is conducted by homomorphic encryption through a trusted third-party server (Chai et al., 2020). After graph expansion, the user client keeps a local subgraph that consists of the user-items interaction and the neighbor users interaction for local training and updates learnable parameters to a central server for aggregation.

4 Anti-Money Laundering (AML) Detection

4.1 Motivation

Money laundering refers to the process of making illegally obtained funds appear legitimate. With the development of technology, criminals' money laundering methods are becoming more and more hidden. Graph structure appears to be a natural way to capture complex patterns. An Anti-Money Laundering (AML) transaction graph is a representation of transactions among bank accounts aimed at detecting and preventing money laundering activities. Due to the legal and privacy concerns, banks generally cannot share customer transaction information. Therefore, we need to form sub-graphs for each bank and perform federated graph training.

There are several difficulties to model transaction activities. First, the data label, illicit and normal is usually highly imbalance, only 0.1% transactions are illicit. Most machine learning algorithms have a high false negative rate, indicating that illicit transactions are incorrectly flagged as legitimate. Second, the transaction graph is directed and consists complicated structure like cycles and multiple edges between the same nodes (Egressy et al., 2023), making it difficult to model using standard message-passing GNN.

4.2 Data Sources

- **AML HI-Small**: The dataset is a series of several synthetic AML datasets released by IBM (Altman et al., 2024), available at `https://www.kaggle.com/datasets/ealtman2019/ibm-transactions-for-anti-money-laundering-aml`. Due to the limited computing power of single machine, we choose small scale dataset with higher illicit (HI-Small) to analyze. It consists of 4268525 transaction records from 336889 bank accounts. Each record consists of incoming and outgoing bank account id, as well as timestamp, amount received, currency, payment format four features. Our task is to predict whether the payment is illicit or not.

We split the dataset with a ratio of 60% training, 20% validation and 10% test by timestamps, following the setting of Egressy et al. (2023). We form a transaction network where nodes are bank accounts are nodes and edges are transactions.

4.3 Models

Unlike the recommendation systems above, transaction happens in one-way and there can be multiple transaction between two nodes. Egressy et al. (2023) considered financial transaction networks as directed multigraphs. Message Passing Neural Network (MPNN) framework are used in the family of GNNs, where the message passing is on the both sides. While in directed graphs, we need to distinguish the incoming and outgoing neighbors. We follow Egressy et al. (2023)'s work in centralized training to make some adaptions on standard message-passing GNNs. The aggregation of incoming neighbors and outgoing neighbors are aggregated subsequently.

For a directed edge (u, v), we

$$\begin{aligned}
&\text{Aggregate incoming neighbors}: && a_{in}^{(l)}(v) = \text{AGGREGATE}\left(\{h^{(l-1)}(u) | u \in \mathcal{N}_{in}(v)\}\right) \\
&\text{Aggregate outgoing neighbors}: && a_{out}^{(l)}(v) = \text{AGGREGATE}\left(\{h^{(l-1)}(u) | u \in \mathcal{N}_{out}(v)\}\right) \\
&\text{Feed forward}: && h^{(l)}(v) = \text{UPDATE}\left(h^{(l-1)}(v), a_{in}^{(l)}(v), a_{out}^{(l)}(v)\right)
\end{aligned}$$

the transaction direction can be recognized in the graph. Note that the edges may also have features, which should be included in massage passing.

Apart from reverse message passing, Egressy et al. (2023) also use techniques including port numbering and ego IDs to improve performance. Based on their experiments, adding reverse message boost the F1-score most. Therefore, we adopt reverse message in our FL experiments.

R-GCN (Schlichtkrull et al., 2018) and PNA (Corso et al., 2020) are used to do link prediction. The implementation is the same as 3.2, except the edges also have several features. These models consider more complicated structure during the message passing. R-GCN learns separate sets of weights for each relation type, allowing the model to capture the unique patterns and dependencies associated with each type of edge. PNA leverages the concept of graph invariant transformations that incorporates node features directly into the aggregation process. We compare the model performances before and after including the reverse passing.

5 Empirical Analysis

5.1 Link Regression

The test results of Epinions and MovieLens are shown in Table 1 and Figure 1, 2. The centralized and local training results are benchmark to evaluate the performance. We observe the FL Mean Square Error (MSE) is close to the centralized training and better than client local training. Three GNN models have very similar performance in MovieLens. We notice that GCN tends to be robust in FL and local training. Overall, there is no dominance among these three models.

Table 1: Link regression performance

Datasets	**FL**			**Centralized**		
(MSE)	GCN	GAT	SAGE	GCN	GAT	SAGE
Epinions	1.374	**1.359**	1.443	**1.317**		1.360
MovieLens	**1.052**	1.088	1.133	**0.822**	0.880	

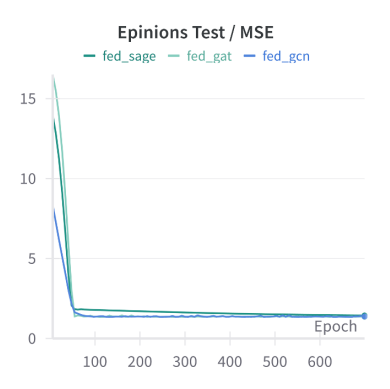

Figure 1: Epinions Federated Learning

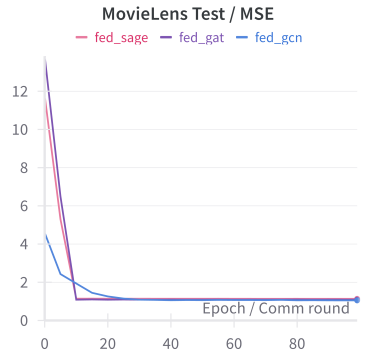

Figure 2: MovieLens Federated Learning

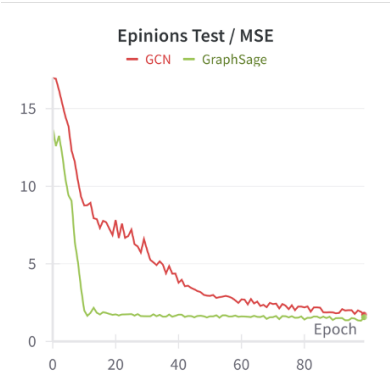

Figure 3: Epinions centralized training

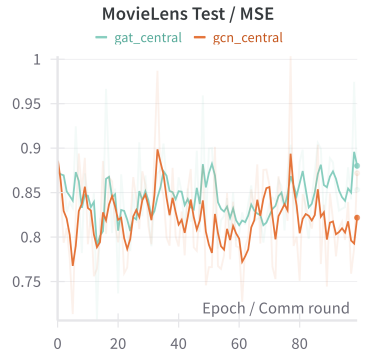

Figure 4: Epinions local training

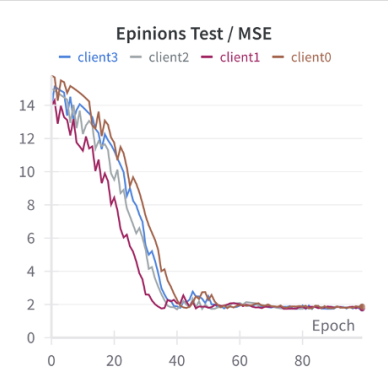

Figure 5: MovieLens centralized training

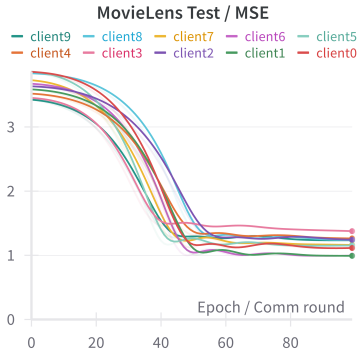

Figure 6: MovieLens local training (selected 10 clients)

5.2 Link Classification

We divide AML dataset into five parts uniformly for FL. Our objective is to simulate that several regulatory agencies each have some bank transaction data. Due to privacy issues, the original data cannot be shared. Each agency can only update parameters and node embeddings to the central server for aggregation.

For highly imbalanced classification task, we not only need to evaluate the overall prediction accuracy but also assess performance in each class separately. Minority class F1 score is often used to assess minor class performance. As shown in Table 2 and Figure 7, 8, 9, all models except GAT can achieve a desirable ROC-AUC score, above 90%. The F1, recall, and precision score of GAT always stay at zero, indicating that it cannot discriminate the positive class (illicit transactions) at all. The result is consistent with Egressy et al. (2023). They performed experiments on synthetic subgraphs and GAT cannot capture important AML patterns like degin and fan-in. Compared with GAT, RGCN and PNA can capture most AML patterns, and the F1 score is acceptable (above 40% in Table 2). In fact, there are many traditional ways to boost the recall (true positive rate), like over-sampling the minority class and adjusting class weights and thresholds. However, the improvement of recall rate is usually achieved by sacrificing the prediction accuracy of the negative class, which is also undesirable in many scenarios. As a result, improving model capabilities to detect complicate subgraph patterns is a better solution.

Next we discuss the FL performance. As shown in Table 2, without reverse passing, the F1 score of FL drops significantly compared with centralized training, from 0.3042 to 0.1187 in RGCN and 0.4181 to 0.1679 in PNA. After adding reverse passing, the FL performance improves to 0.3766. In centralized training, the improvement of F1 score by reverse passing PNA is not very obvious. However, we observe dramatic improvements of reverse passing in federated learning, from 0.1679 to 0.3766.

[1]With reverse message passing

Table 2: AML Minority class F1 score

F1 score	GAT	RGCN	PNA	r-PNA [1]
FL	0	0.1187	0.1679	**0.3766**
Centralized	0	0.3042	0.4181	**0.4814**
Local (avg)	0			0.2345

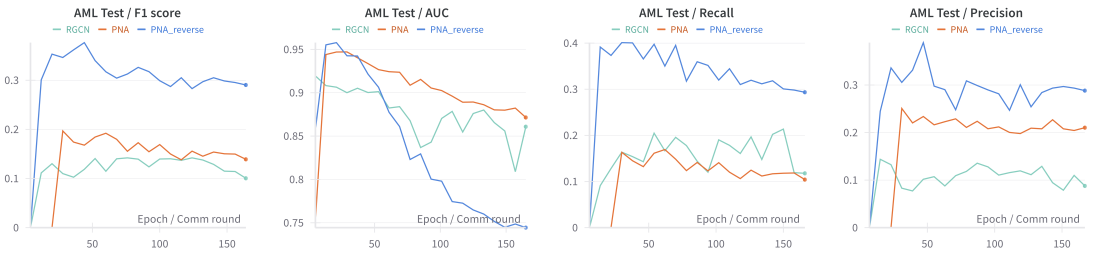

Figure 7: AML federated learning

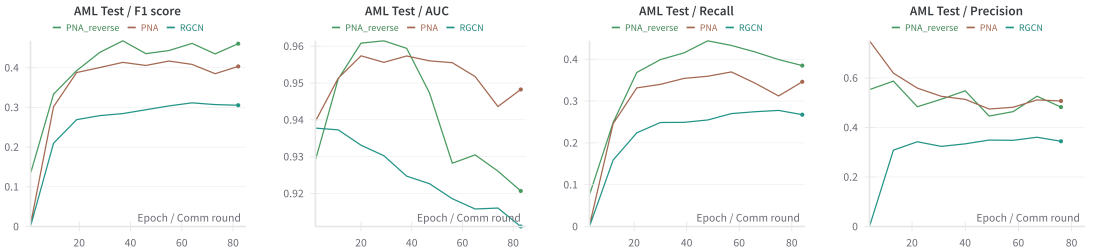

Figure 8: AML centralized training

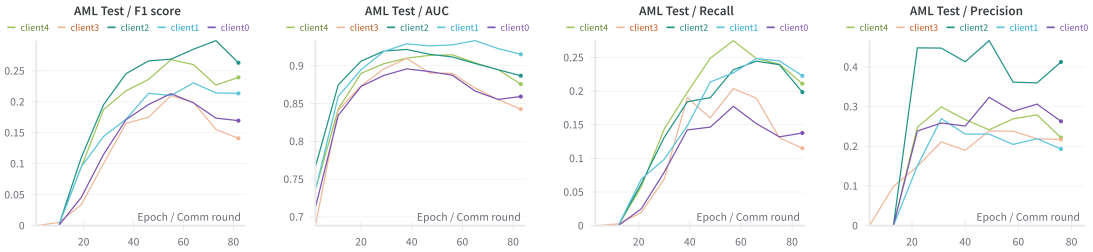

Figure 9: AML local training

6 Discussion

6.1 Summary

We have implemented several FedGNN tasks in link regression and classification. In common cases like the Epinions and MovieLens dataset, the model selection does not dominate the performance much. The performance of FL, centralized training, and even client local training do

not differ too much. Federated GCN is already good enough and it could take much longer time to train on complicated models. However, when there exist special patterns in the transaction graph, like highly imbalance of the two classes or multi-graph connections, partial training can not easily capture the overall information of the dataset. In this case, we need to carefully study the graph pattern to make fine model adjustments. We observe that reverse passing is extremely important in Federated AML detection. It reduces the gap between FL and centralized training F1 score significantly.

6.2 Limitation and Future Study

In this report, due to the computational power and limited memory of a single machine, we only consider using reverse passing to boost the F1 score for the AML dataset and adopt a small batch size. In the research of Egressy et al. (2023), they also add port numbering and ego IDs as Multi-GNN to capture special patterns in the graph. By including all three techniques, centralized training can reach an F1 score of 68%. After tuning the hyperparameters, we believe these techniques can also perform well in FL.

We want to simulate more patterns in FL settings. Besides uniformly dividing the datasets into several parts, it is also interesting to split the dataset according to banks (in total, 970 banks) for federated learning. The performance of FL on non-IID large-scale data is worth exploring.

References

Altman, E., Blanuša, J., Von Niederhäusern, L., Egressy, B., Anghel, A., & Atasu, K. (2024). Realistic synthetic financial transactions for anti-money laundering models. *Advances in Neural Information Processing Systems*, *36*.

Berg, R. v. d., Kipf, T. N., & Welling, M. (2017). Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*.

Chai, D., Wang, L., Chen, K., & Yang, Q. (2020). Secure federated matrix factorization. *IEEE Intelligent Systems*, *36*(5), 11–20.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., & Veličković, P. (2020). Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, *33*, 13260–13271.

Dwork, C. (2008). Differential privacy: A survey of results. In *International conference on theory and applications of models of computation* (pp. 1–19).

Egressy, B., Von Niederhäusern, L., Blanusa, J., Altman, E., Wattenhofer, R., & Atasu, K. (2023). Provably powerful graph neural networks for directed multigraphs. *arXiv preprint arXiv:2306.11586*.

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in neural information processing systems*, *30*.

He, C., Balasubramanian, K., Ceyani, E., Yang, C., Xie, H., Sun, L., ... others (2021). Fedgraphnn: A federated learning benchmark system for graph neural networks. In *Iclr 2021 workshop on distributed and private machine learning (dpml)*.

He, C., Li, S., So, J., Zeng, X., Zhang, M., Wang, H., ... others (2020). Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Lai, F., Dai, Y., Singapuram, S., Liu, J., Zhu, X., Madhyastha, H., & Chowdhury, M. (2022). Fedscale: Benchmarking model and system performance of federated learning at scale. In *International conference on machine learning* (pp. 11814–11827).

Liu, R., Xing, P., Deng, Z., Li, A., Guan, C., & Yu, H. (2022). Federated graph neural networks: Overview, techniques and challenges. *arXiv preprint arXiv:2202.07256*.

McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics* (pp. 1273–1282).

Rivest, R. L., Adleman, L., Dertouzos, M. L., et al. (1978). On data banks and privacy homomorphisms. *Foundations of secure computation*, *4*(11), 169–180.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., & Welling, M. (2018). Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, eswc 2018, heraklion, crete, greece, june 3–7, 2018, proceedings 15* (pp. 593–607).

Tang, J., Gao, H., & Liu, H. (2012). mtrust: Discerning multi-faceted trust in a connected world. In *Proceedings of the fifth acm international conference on web search and data mining* (pp. 93–102).

Wu, C., Wu, F., Cao, Y., Huang, Y., & Xie, X. (2021). Fedgnn: Federated graph neural network for privacy-preserving recommendation. *arXiv preprint arXiv:2102.04925*.

Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *10*(2), 1–19.