FedGNN: Federated Graph Neural Network for Privacy-Preserving Recommendation

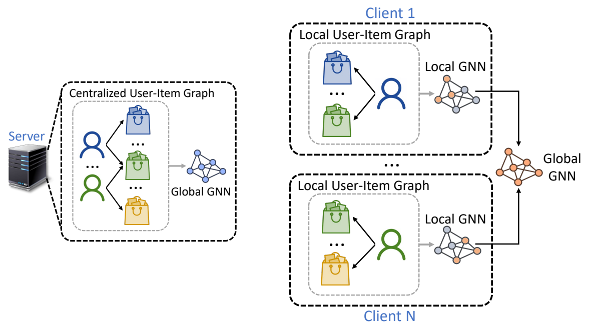
Chuhan Wu [1] Fangzhao Wu [2] Yang Cao [3] Lingjuan Lyu [4] Yongfeng Huang [1] Xing Xie [2]

Abstract

Graph neural network (GNN) is widely used for recommendation to model high-order interactions between users and items. Existing GNN-based recommendation methods rely on centralized storage of user-item graphs for model learning, which may arouse privacy concerns and risks due to the privacy-sensitivity of user behaviors. In this paper, we propose a privacy-preserving GNN-based recommendation method based on federated learning, named *FedGNN*. It can collaboratively train GNN models from decentralized user data and meanwhile exploit high-order user-item interaction information with privacy well protected. In our method, we locally train GNN model in each user client based on the local user-item graph inferred from the local user-item interaction data, where the gradients of GNN are communicated between clients and a server for aggregation and local GNN update. Since local gradients may contain private information, we apply differential privacy techniques to the local gradients to protect user privacy. In addition, to protect the items that users have interactions with, we incorporate randomly sampled items as pseudo interacted items for anonymity. Since local data only contain first-order interaction information, we propose a user-item graph expansion method to expand local user-item graphs and propagate high-order information in a privacy-preserving way. Extensive experiments on six datasets show the effectiveness of *FedGNN* in GNN-based recommendation and privacy protection.

[*]Equal contribution [1]Tsinghua University, Beijing 100084, China [2]Microsoft Research Asia, Beijing 100080, China [3]Kyoto University, Kyoto 615-8558, Japan [4]Ant group, Hangzhou 310013, China. Correspondence to: Chuhan Wu <wuchuhan15@gmail.com>.

This work was presented at the International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML 2021 (FL-ICML'21). This workshop does not have official proceedings and this paper is non-archival. Copyright 2021 by the author(s).



(a) Centralized GNN. (b) Decentralized GNN in FL.

Figure 1. Comparisons between centralized and decentralized training of GNN models for recommendation.

1. Introduction

Graph neural network (GNN) is widely used by many personalized recommendation methods in recent years (Ying et al., 2018; Jin et al., 2020), since it can capture high-order interactions between users and items on the user-item graph to enhance the user and item representations (Zhou et al., 2018; Zhang et al., 2019). For example, Berg et al. (2017) proposed to use graph convolutional autoencoders to learn user and item representations from the user-item bipartite graph. Wang et al. (2019) proposed to use a three-hop graph attention network to capture the high-order interactions between users and items. These existing GNN-based recommendation methods usually necessitate centralized storage of the entire user-item graph to learn GNN models and the representations of users and items, which means that the user-item interaction data needs to be centrally stored, as shown in Fig. 1(a). However, user-item interaction data is highly privacy-sensitive, and its centralized storage can lead to the privacy concerns of users and the risk of data leakage (Lyu et al., 2017; Shin et al., 2018; Lyu et al., 2020c;b). Moreover, under the pressure of strict data protection regulations such as GDPR[1], online platforms may not be able to centrally store user-item interaction data to learn GNN models for recommendation in the future (Yang et al., 2019).

An intuitive way to tackle the privacy issue of user-item interaction data is to locally store the raw data on user de-

[1]https://gdpr-info.eu

vices and learn local GNN models based on it. However, for most users the volume of interaction data on their devices is too small to locally train accurate GNN models. Federated learning is a potential way to collaboratively learn GNN models from the local data decentralized on a large number of user clients in a privacy-preserving way. As shown in Fig. 1(b), the clients learn their local GNN models based on the local user-item graphs inferred from the local user-item interaction data, and aggregate these local models into a global one for recommendation. However, there exist two challenges in this framework. First, the local GNN model trained on local user data may convey private information, and it is challenging to protect user privacy when synthesizing the global GNN model from the local ones. Second, the local user data only contains *first-order* user-item interactions, and users' interaction items cannot be directly exchanged due to privacy restrictions. Thus, it is very challenging to exploit the high-order user-item interactions to enhance GNN model learning without privacy leakage.

In this paper, we propose a federated framework named *FedGNN* for privacy-preserving GNN-based recommendation, which can effectively exploit high-order user-item interaction information by collaboratively training GNN models for recommendation in a privacy-preserving way. In *FedGNN* user-item interaction data is locally stored on user clients and there is no global user-item graph due to privacy restrictions. Each client locally learns a GNN model and the embeddings of user and items based on the user-item graph inferred from the local user-item interaction data on this device. The user devices compute the gradients of GNN and user/item embeddings and upload them to a central server, which aggregates the gradients from a number of users and distributes them to user devices for local updates. Since the communicated gradients contain private information, we propose a privacy-preserving model update method to protect user-item interaction information. More specifically, we apply differential privacy (DP) techniques to the local gradients computed by user clients to protect user privacy. In addition, in order to protect the real items that a user has interactions with, we generate random embedding gradients of randomly sampled pseudo interacted items. Besides, to exploit high-order information of the user-item graph without leaking user privacy, we propose a privacy-preserving user-item graph expansion method that aims to find the neighbors of users with co-interacted items and exchange their embeddings periodically, which can expand local user-item graphs and propagate high-order interaction information. We conduct massive experiments on six widely used benchmark datasets for recommendation, and the results show that *FedGNN* can achieve competitive results with existing centralized GNN-based recommendation methods and meanwhile effectively protect user privacy.

2. Methodology

In this section, we first present the problem definitions in our federated framework named *FedGNN* to train GNN-based recommendation model in a privacy-preserving way, then introduce the details of *FedGNN*, and finally provide some discussions and analysis on privacy protection.

2.1. Problem Formulation

Denote $\mathcal{U} = \{u_1, u_2, ..., u_P\}$ and $\mathcal{T} = \{t_1, t_2, ..., t_Q\}$ as the sets of users and items respectively, where P is the number of users and Q is the number of items. Denote the rating matrix between users and items as $\mathbf{Y} \in \mathbb{R}^{P \times Q}$, which is used to form a bipartite user-item graph $\mathcal{G}$ based on the observed ratings $\mathbf{Y}_o$. We assume that the user u_i has interactions with K items, which are denoted by $[t_{i,1}, t_{i,2}, ..., t_{i,K}]$. These items and the user u_i can form a first-order local user-item subgraph $\mathcal{G}_i$. The ratings that given to these items by user u_i are denoted by $[y_{i,1}, y_{i,2}, ..., y_{i,K}]$. To protect user privacy (both the private ratings and the items a user has interactions with), each user device locally keeps the interaction data of this user, and the raw data never leaves the user device. We aim to predict the user ratings based on the interaction data $\mathcal{G}_i$ locally stored on user devices in a privacy-preserving way. Note that there is no global user-item interaction graph in *FedGNN* and local graphs are built and stored in different device, which is very different from existing federated GNN methods (Mei et al., 2019; Jiang et al., 2020; He et al., 2021; Ni et al., 2021) that require the entire graph is built and stored together in at least one platform or device.

2.2. FedGNN Framework

Next, we introduce the framework of *FedGNN* to train GNN-based recommendation model in a privacy-preserving way. It can leverage the highly decentralized user interaction data to learn GNN models for recommendation by exploiting the high-order user-item interactions in a privacy-preserving way. The framework of *FedGNN* is shown in Fig. 2. It mainly consists of a central server and a large number of user clients. The user client keeps a local subgraph that consists of the user interaction histories with items and the neighbors of this user with co-interacted items with this user (we will introduce how to incorporate user neighbors in Section 2.4). Each client learns the user/item embeddings and the GNN models from its local subgraph, and uploads the gradients to a central server. The central server is responsible for coordinating these user clients in the model learning process by aggregating the gradients received from a number of user clients and delivering the aggregated gradients to them. Next, we introduce how they work in detail.

The local subgraph on each user client is constructed from

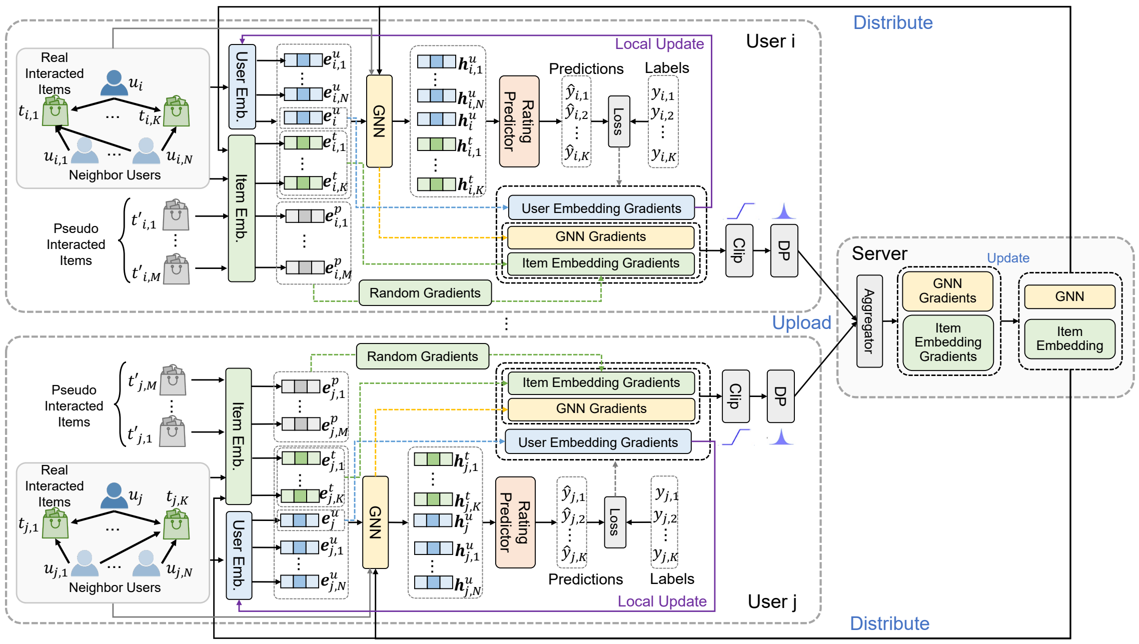


Figure 2. The framework of *FedGNN*.

the user-item interaction data and the neighboring users that have co-interacted items with this user. The node of this user is connected to the nodes of the items it interacted with, and these item nodes are further connected to the anonymous neighboring users. An embedding layer is first used to convert the user node u_i, the K item nodes $[t_{i,1}, t_{i,2}, ..., t_{i,K}]$ and the N neighboring user nodes $[u_{i,1}, u_{i,2}, ..., u_{i,N}]$ into their embeddings, which are denoted as $\mathbf{e}_i^u$, $[\mathbf{e}_{i,1}^t, \mathbf{e}_{i,2}^t, ..., \mathbf{e}_{i,K}^t]$ and $[\mathbf{e}_{i,1}^u, \mathbf{e}_{i,2}^u, ..., \mathbf{e}_{i,N}^u]$, respectively. Since the user embeddings may not be accurate enough when the model is not well-tuned, we first exclude the neighboring user embeddings at the beginning of model learning, and then incorporate them into model learning when they have been tuned. Note that the embeddings of the user u_i and the item embeddings are synchronously updated during model training, while the embeddings of neighboring users are periodically updated.

Next, we apply a graph neural network to these embeddings to model the interactions between nodes on the local first-order sub-graph. Various kinds of GNN network can be used in *FedGNN*, such as graph convolution network (GCN) (Kipf & Welling, 2017), gated graph neural network (GGNN) (Li et al., 2016) and graph attention network (GAT) (Velickovic et al., 2018). The GNN model outputs the hidden representations of the user and item nodes, which are denoted as $\mathbf{h}_i^u$, $[\mathbf{h}_{i,1}^t, \mathbf{h}_{i,2}^t, ..., \mathbf{h}_{i,K}^t]$ and $[\mathbf{h}_{i,1}^u, \mathbf{h}_{i,2}^u, ..., \mathbf{h}_{i,N}^t]$, respectively. Then, a rating predictor module is used to predict the ratings given by the user u_i to her interacted items (denoted by $[\hat{y}_{i,1}, \hat{y}_{i,2}, ..., \hat{y}_{i,K}]$) based on the embeddings of items and this user. These predicted ratings are compared against the gold ratings locally stored on the user device to compute the loss function. For the user u_i, the loss function $\mathcal{L}_i$ is computed as $\mathcal{L}_i = \frac{1}{K}\sum_{j=1}^{K} |\hat{y}_{i,j} - y_{i,j}|^2$. We use the loss $\mathcal{L}_i$ to derive the gradients of the models and embeddings, which are denoted by $\mathbf{g}_i^m$ and $\mathbf{g}_i^e$, respectively. These gradients will be further uploaded to the server for aggregation.

The server aims to coordinate all user devices and compute the global gradients to update the model and embedding parameters in these devices. In each round, the server awakes a certain number of user clients to compute gradients locally and send them to the server. After the server receiving the gradients from these users, the aggregator in this server will aggregate these local gradients into a unified one $\mathbf{g}$.[2] Then, the server sends the aggregated gradients to each client to conduct local parameter update. Denote the parameter set in the i-th user device as Θ_i. It is updated by $\Theta_i = \Theta_i - \alpha\mathbf{g}$ (α is the learning rate). This process is iteratively executed until the model converges.[3] We then introduce two modules for privacy protection in FedGNN, i.e., a privacy-preserving model update module for protecting gradients and a privacy-preserving user-item graph expansion module to protect user privacy in high-order user-item interaction modeling.

2.3. Privacy-Preserving Model Update

If we directly upload the GNN model and item embedding gradients, then there may be some privacy issues due to

[2]We use the FedAvg algorithm to implement the aggregator.

[3]When the model learning process completes, the user clients will upload their locally inferred hidden user embeddings to the server for providing future recommendation services.

following reasons. First, for embedding gradients, only the items that a user has interactions with have non-zero gradients to update their embeddings, and the server can directly recover the full user-item interaction history based on the non-zero item embedding gradients. Second, besides the embedding gradients, the gradients of the GNN model and rating predictor may also leak private information of user histories and ratings (Zhu et al., 2019), because the GNN model gradients encode the preferences of users on items. In existing methods such as FedMF (Chai et al., 2020), homomorphic encryption techniques are applied to gradients to protect private ratings. However, they need to locally maintain the embedding table of the entire item set $\mathcal{T}$ in user devices and upload it in every iteration to achieve user interaction history protection, which is impractical due to the huge storage and communication cost.

To tackle these challenges, we propose two strategies to protect user privacy in the model update process. The first one is pseudo interacted item sampling. Concretely, we sample M items that the user has not interacted with, and randomly generate their gradients $\mathbf{g}_i^p$ using a Gaussian distribution with the same mean and co-variance values with the real item embedding gradients. The real embedding gradients $\mathbf{g}_i^e$ are combined with the pseudo item embedding gradients $\mathbf{g}_i^p$, and the unified gradient of the model and embeddings on the i-th user device (Line 27 in Algorithm 1) is modified as $\mathbf{g}_i = (\mathbf{g}_i^m, \mathbf{g}_i^e, \mathbf{g}_i^p)$. The second one is differential privacy (Lyu et al., 2020a). Following (Qi et al., 2020), we clip the local gradients on user clients based on their L2-norm with a threshold δ, and apply a differential privacy (DP) module with zero-mean Laplacian noise to the unified gradients to achieve better user privacy protection, which are formulated as follows:

$$\mathbf{g}_i = clip(\mathbf{g}_i, \delta) + Laplace(0, \lambda), \qquad (1)$$

where λ is the strength of noise.[4] The protected gradients $\mathbf{g}_i$ are uploaded to the server for aggregation.

2.4. Privacy-Preserving User-Item Graph Expansion

Then, we introduce our privacy-preserving user-item graph expansion method that aims to find the neighbors of users and extend the local user-item graphs in a privacy-preserving way. In existing GNN-based recommendation method based on centralized graph storage, high-order user-item interactions can be directly derived from the global user-item graph. However, when user data is decentralized, it is a non-trivial task to incorporate high-order user-item interactions without violating user privacy protection. To solve this problem, we propose a privacy-preserving user-item graph expansion method that finds the anonymous neighbors of users to enhance user and item representation learning, where user privacy does not leak. Its framework is shown in Fig. 3. The central server that maintains the recommendation services first generates a public key, and then distributes it to all user clients for encryption. After receiving the public key, each user device applies homomorphic encryption (Chai et al., 2020) to the IDs of the items she interacted based on this key because the IDs of these items are privacy-sensitive. The encrypted item IDs as well as the embedding of this user are uploaded to a trusted third-party server. This server finds the users who interacted with the same items via item matching, and then provides each user with the embeddings of her anonymous neighbors. In this stage, the server for recommendation never receives the private information of users, and the third-party server cannot obtain any private information of users and items since it cannot decrypt the item IDs. We connect each anonymous user node with its interacted item nodes. In this way, the local user-item graph can be enriched by the high-order user-item interactions without harming the protection of user privacy.

[4]The privacy budget ϵ can be bounded by $\frac{2\delta}{\lambda}$.

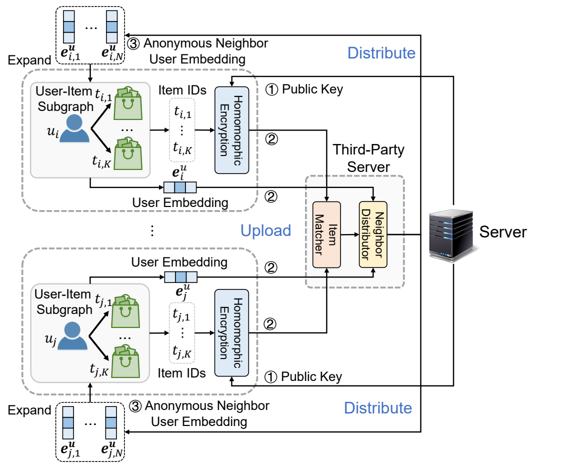


Figure 3. The framework of the privacy-preserving user-item graph expansion method.

2.5. Analysis on Privacy Protection

The user privacy is protected by four aspects in *FedGNN*. First, in *FedGNN* the recommendation server never collects raw user-item interaction data, and only local computed gradients are uploaded to this server. Based on the data processing inequality (McMahan et al., 2017), we can infer that these gradients contain much less private information than the raw user interaction data. Second, the third-party server also cannot infer private information from the encrypted item IDs since it cannot obtain the private key. However, if the recommendation server colludes with the third-party server by exchanging the private key and item table, the user interaction history will not be protected. Fortunately, the private ratings can still be protected by our privacy-preserving model update method. Third, in *FedGNN* we propose a pseudo interacted item sampling method to protect the real interacted items by sampling a number of items that are not interacted with a user. Since gradients of both kinds of items have the same mean and co-variance values, it is difficult to discriminate the real interacted items from the pseudo ones if the number of pseudo interacted items is sufficiently large. It is proved in (Liu et al., 2021) that *FedGNN* can achieve $\frac{K}{M}-$index privacy. Thus, the number of pseudo interacted items can be relatively larger to achieve better privacy protection as long as the computation resources of user devices permit. Fourth, we apply the DP technique to the gradients locally computed by the user device, making it more difficult to recover the raw user consumption history from these gradients. It is shown in (Qi et al., 2020) that the upper bound of the privacy budget ϵ is $\frac{2\delta}{\lambda}$, which means that we can achieve a smaller privacy budget ϵ by using a smaller clipping threshold δ or a larger noise strength λ. However, the model gradients will be inaccurate if ϵ is too small. Thus, we need to properly choose both hyperparameters to balance model performance and privacy protection.

3. Experiments

3.1. Dataset and Experimental Settings

In our experiments, following (Berg et al., 2017) we use six widely used benchmark datasets for recommendation, including MovieLens[5] (100K, 1M, and 10M), Flixster, Douban, and YahooMusic. We use the preprocessed subsets of the Flixster, Douban, and YahooMusic datasets provided by (Monti et al., 2017).[6] We denote the three versions of MovieLens as ML-100K, ML-1M and ML-10M respectively, and we denote YahooMusic as Yahoo. The detailed statistics of these datasets are summarized in Table 1.

Table 1. Statistics of the datasets.

Dataset	**#Users**	**#Items**	**#Ratings**	**Rating Levels**
Flixster	3,000	3,000	26,173	0.5,1,...,5
Douban	3,000	3,000	136,891	1,2,...,5
Yahoo	3,000	3,000	5,335	1,2,...100
ML-100K	943	1,682	100,000	1,2,...,5
ML-1M	6,040	3,706	1,000,209	1,2,...,5
ML-10M	69,878	10,677	10,000,054	0.5,1,...,5

In our experiments, we use graph attention network (GAT) (Velickovic et al., 2018) as the GNN model, and use dot product to implement the rating predictor. The user and item embeddings and their hidden representations learned by graph neural networks are 256-dim. We update the neighbor user embeddings after every epoch. The gradient clipping threshold δ is set to 0.1, and the strength of Laplacian noise in the DP module is set to 0.2 to achieve 1-differential privacy. The number of pseudo interacted items is set to 1,000. The number of users used in each round of model training is 128, and the total number of epoch is 3. SGD optimizer with a learning rate of 0.01 is used. The splits of datasets are the same as those used in (Berg et al., 2017), and these hyperparameters are selected according to the validation performance.We use rooted mean square error (RMSE) as performance metric, and we report the average RMSE scores over 10 repetitions.

[5]https://grouplens.org/datasets/movielens/

[6]https://github.com/fmonti/mgcnn

Table 2. Performance of different methods in terms of RMSE. Results of FedGNN and the best-performed baseline are in bold.

Methods	**Flixster**	**Douban**	**Yahoo**	**ML-100K**	**ML-1M**	**ML-10M**
PMF	1.375	0.886	26.6	0.965	0.883	0.856
SVD++	1.155	0.869	24.4	0.952	0.860	0.834
GRALS	1.313	0.833	38.0	0.934	0.849	0.808
sRGCNN	1.179	0.801	22.4	0.922	0.837	0.789
GC-MC	**0.941**	0.734	**20.5**	**0.905**	**0.832**	**0.777**
PinSage	0.945	**0.732**	21.0	0.914	0.840	0.790
NGCF	0.954	0.742	20.9	0.916	0.833	0.779
GAT	0.952	0.737	21.2	0.913	0.835	0.784
FCF	1.064	0.823	22.9	0.957	0.874	0.847
FedMF	1.059	0.817	22.2	0.948	0.872	0.841
FedGNN	**0.980**	**0.775**	**20.7**	**0.910**	**0.839**	**0.793**

3.2. Performance Evaluation

First, we compare the performance of our *FedGNN* approach with several recommendation methods based on centralized storage of user data and several privacy-preserving ones based on federated learning, including: (1) PMF (Mnih & Salakhutdinov, 2008), probability matrix factorization; (2) SVD++ (Koren, 2008), a variant of singular value decomposition; (3) GRALS (Rao et al., 2015), a collaborative filtering approach with graph information; (4) sRGCNN (Monti et al., 2017), a matrix completion method with recurrent multi-graph neural networks; (5) GC-MC (Berg et al., 2017), a matrix completion method based on graph convolutional autoencoders; (6) PinSage (Ying et al., 2018), a recommendation approach based on 2-hop GCN networks; (7) NGCF (Wang et al., 2019), neural graph collaborative filtering; (8) GAT (Velickovic et al., 2018), graph attention network; (9) FCF (Ammad et al., 2019), privacy-preserving recommendation based on federated collaborative filtering; (10) FedMF (Chai et al., 2020), privacy-preserving recommendation based on secure matrix factorization. The recommendation performance of these methods is summarized in Table 2. We observe that the methods which incorporate high-order information of the user-item graph (e.g., *GC-MC*, *PinSage* and *NGCF*) achieve better performance than those based on first-order information only (*PMF*). This is because modeling the high-order interactions between users and items can enhance user and item representation learning, and thereby improves the accuracy of recommendation. In addition, compared with the methods based on centralized user-item interaction data storage like *GC-MC* and *NGCF*, *FedGNN* can achieve comparable or even better performance. It shows that *FedGNN* can protect user

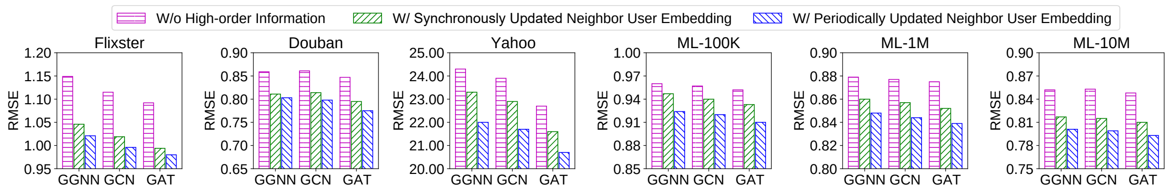


Figure 4. Influence of neighbor user information and different GNN architectures.

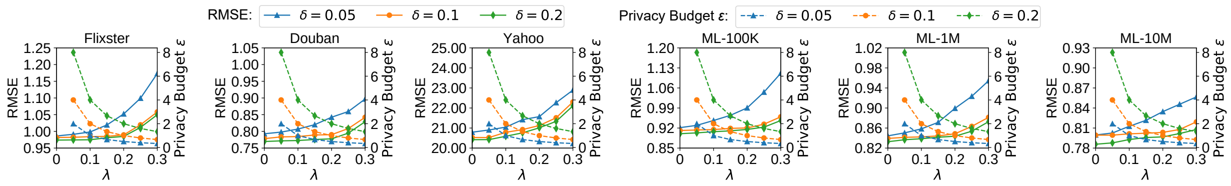


Figure 5. The recommendation RMSE and privacy budget ϵ w.r.t. different clipping threshold δ and noise strength λ.

privacy and meanwhile achieve satisfactory recommendation performance. Besides, among the compared privacy-preserving recommendation methods, *FedGNN* achieves the best performance. This is because *FedGNN* can incorporate high-order information of the user-item graphs, while *FCF* and *FedMF* cannot. Moreover, *FedGNN* can protect both ratings and user-item interaction histories, while *FCF* and *FedMF* can only protect ratings.

3.3. Model Effectiveness

Then, we verify the effectiveness of incorporating high-order information of the user-item graphs as well as the generality of our approach. We compare the performance of *FedGNN* and its variants with synchronously updated neighbor user embeddings or without high-order user-item interactions. In addition, we also compare their results under different implementations of their GNN models (GGNN, GCN and GAT). The results are shown in Fig. 4. Compared with the baseline performance reported in Table 2, the performance of *FedGNN* and its variants implemented with other different GNN models is satisfactory. This result shows that *FedGNN* is compatible with different GNN architectures. In addition, the variants that can utilize the high-order information perform better. It shows the effectiveness of *FedGNN* in incorporating high-order information of the user-item graph into recommendation. Moreover, we find that using periodically updated neighbor user embeddings is slightly better than using fully trainable ones that are synchronously updated in each iteration. This is because the neighboring user embeddings may be inaccurate at the beginning of model training, which is not beneficial for learning precise user and item representations.

3.4. Hyperparameter Analysis

Finally, we explore the influence of the gradient clip threshold δ and the strength λ of the Laplacian noise in the DP module on model performance and privacy protection. The results are plotted in Fig. 5.[7] We find that the difference between the model performance under $\delta = 0.1$ and $\delta = 0.2$ is quite marginal. However, if we clip the gradients with a smaller threshold such as 0.05, the prediction error will substantially increase. Thus, we prefer to set $\delta = 0.1$ because we can achieve better privacy protection without much sacrifice of model performance. In addition, the model performance declines with the growth of the noise strength λ, while the performance loss is not too heavy if λ is not too large. Thus, a moderate value of λ such as 0.2 is preferable to achieve a good balance between privacy protection and recommendation accuracy.

4. Conclusion

In this paper, we propose a federated framework for privacy-preserving GNN-based recommendation, which aims to collaboratively train GNN models from decentralized user data by exploiting high-order user-item interactions in a privacy-preserving manner. We locally train GNN model in each user client based on the local user-item graph stored on this device. Each client uploads the locally computed gradients to a server for aggregation, which are further sent to user clients for local updates. To protect user-item interaction data during model training, we apply differential privacy techniques to the local gradients and sample pseudo interacted items to protect the real items that users have interactions with. Besides, to incorporate high-order user-item interaction information into model learning, we propose a privacy-preserving user-item graph expansion method to extend local graphs and propagate high-order information. Experiments on six datasets show that our approach can achieve competitive performance with existing methods based on centralized storage of user-item interaction data and meanwhile effectively protect user privacy.

[7] A larger λ and smaller δ means better privacy protection.

References

Ammad, M., Ivannikova, E., Khan, S. A., Oyomno, W., Fu, Q., Tan, K. E., and Flanagan, A. Federated collaborative filtering for privacy-preserving personalized recommendation system. *arXiv preprint arXiv:1901.09888*, 2019.

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

Chai, D., Wang, L., Chen, K., and Yang, Q. Secure federated matrix factorization. *IEEE Intelligent Systems*, 2020.

He, C., Balasubramanian, K., Ceyani, E., Rong, Y., Zhao, P., Huang, J., Annavaram, M., and Avestimehr, S. Fedgraphnn: A federated learning system and benchmark for graph neural networks. *arXiv preprint arXiv:2104.07145*, 2021.

Jiang, M., Jung, T., Karl, R., and Zhao, T. Federated dynamic gnn with secure aggregation. *arXiv preprint arXiv:2009.07351*, 2020.

Jin, B., Gao, C., He, X., Jin, D., and Li, Y. Multi-behavior recommendation with graph convolutional networks. In *SIGIR*, pp. 659–668, 2020.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Koren, Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pp. 426–434, 2008.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. S. Gated graph sequence neural networks. In *ICLR*, 2016.

Liu, R., Cao, Y., Chen, H., Guo, R., and Yoshikawa, M. Flame: Differentially private federated learning in the shuffle model. In *AAAI*, 2021.

Lyu, L., He, X., Law, Y. W., and Palaniswami, M. Privacy-preserving collaborative deep learning with application to human activity recognition. In *CIKM*, pp. 1219–1228, 2017.

Lyu, L., He, X., and Li, Y. Differentially private representation for nlp: Formal guarantee and an empirical study on privacy and fairness. In *EMNLP: Findings*, pp. 2355–2365, 2020a.

Lyu, L., Yu, H., and Yang, Q. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133*, 2020b.

Lyu, L., Yu, J., Nandakumar, K., Li, Y., Ma, X., Jin, J., Yu, H., and Ng, K. S. Towards fair and privacy-preserving federated deep models. *IEEE Transactions on Parallel and Distributed Systems*, 31(11):2524–2541, 2020c.

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

Mei, G., Guo, Z., Liu, S., and Pan, L. Sgnn: A graph neural network based federated learning approach by hiding structure. In *IEEE Big Data*, pp. 2560–2568. IEEE, 2019.

Mnih, A. and Salakhutdinov, R. R. Probabilistic matrix factorization. In *NIPS*, pp. 1257–1264, 2008.

Monti, F., Bronstein, M., and Bresson, X. Geometric matrix completion with recurrent multi-graph neural networks. In *NIPS*, pp. 3697–3707, 2017.

Ni, X., Xu, X., Lyu, L., Meng, C., and Wang, W. A vertical federated learning framework for graph convolutional network. *arXiv preprint arXiv:2106.11593*, 2021.

Qi, T., Wu, F., Wu, C., Huang, Y., and Xie, X. Fedrec: Privacy-preserving news recommendation with federated learning. *arXiv preprint arXiv:2003.09592*, 2020.

Rao, N., Yu, H.-F., Ravikumar, P. K., and Dhillon, I. S. Collaborative filtering with graph information: Consistency and scalable methods. In *NIPS*, pp. 2107–2115, 2015.

Shin, H., Kim, S., Shin, J., and Xiao, X. Privacy enhanced matrix factorization for recommendation with local differential privacy. *TKDE*, 30(9):1770–1782, 2018.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR*, 2018.

Wang, X., He, X., Wang, M., Feng, F., and Chua, T.-S. Neural graph collaborative filtering. In *SIGIR*, pp. 165–174, 2019.

Yang, Q., Liu, Y., Chen, T., and Tong, Y. Federated machine learning: Concept and applications. *TIST*, 10(2):1–19, 2019.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, pp. 974–983, 2018.

Zhang, J., Shi, X., Zhao, S., and King, I. Star-gcn: stacked and reconstructed graph convolutional networks for recommender systems. In *IJCAI*, pp. 4264–4270. AAAI Press, 2019.

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

Zhu, L., Liu, Z., and Han, S. Deep leakage from gradients. *arXiv preprint arXiv:1906.08935*, 2019.