Breaking the Aggregation Bottleneck in Federated Recommendation: A Personalized Model Merging Approach

Jundong Chen[1,2], **Honglei Zhang**[1,2], **Chunxu Zhang**[3], **Fangyuan Luo**[4], **Yidong Li**[1,2*]

[1]Key Laboratory of Big Data & Artificial Intelligence in Transportation, Ministry of Education, China
[2]Beijing Jiaotong University [3]Jilin University [4]Beijing University of Technology
{jundongchen, honglei.zhang, ydli}@bjtu.edu.cn, cxzhang19@mails.jlu.edu.cn, luofangyuan@bjut.edu.cn

Abstract

Federated recommendation (FR) facilitates collaborative training by aggregating local models from massive devices, enabling client-specific personalization while ensuring privacy. However, we empirically and theoretically demonstrate that server-side aggregation can undermine client-side personalization, leading to suboptimal performance, which we term the aggregation bottleneck. This issue stems from the inherent heterogeneity across numerous clients in FR, which drives the globally aggregated model to deviate from local optima. To this end, we propose FedEM, which elastically merges the global and local models to compensate for impaired personalization. Unlike existing personalized federated recommendation (pFR) methods, FedEM (1) investigates the aggregation bottleneck in FR through theoretical insights, rather than relying on heuristic analysis; (2) leverages off-the-shelf local models rather than designing additional mechanisms to boost personalization. Extensive experiments on real-world datasets demonstrate that our method preserves client personalization during collaborative training, outperforming state-of-the-art baselines. Our code is available at https://github.com/jundongchen13/FedEM.

Introduction

Federated recommendation (FR), as an emerging on-device learning paradigm, ensures that clients' raw data remains local during the training process, thus protecting user privacy (Sun et al. 2024; Yin et al. 2024). Pioneering works include FedMF (Chai et al. 2020) and FedNCF (Perifanis and Efraimidis 2022), which apply matrix factorization and neural collaborative filtering, respectively, within the federated framework. Conceptually, the clients upload local models after local training to the server for global aggregation. Later, they download the global model as the new local model for the next training round (Chen et al. 2023).

However, due to the varying user preferences in FR, the interaction data from different clients are not independent and identically distributed (Non-IID), which naturally leads to data heterogeneity (Sun et al. 2024). Traditional FR methods fail to address this issue by sharing one same global model across all clients (Zhang et al. 2023b). Hence, personalized federated recommendation (pFR) has been introduced to tailor client-specific models. For example, PFedRec (Zhang et al. 2023a) enables dual personalization of both local and global components, while FedRAP (Li, Long, and Zhou 2024) trains an extra personalized model locally. Although effective in practice, existing pFR methods suffer from two key limitations: (1) They overlook the degradation of local personalization caused by global aggregation in FR; (2) They rely on heuristically designed personalization mechanisms, which limit their compatibility.

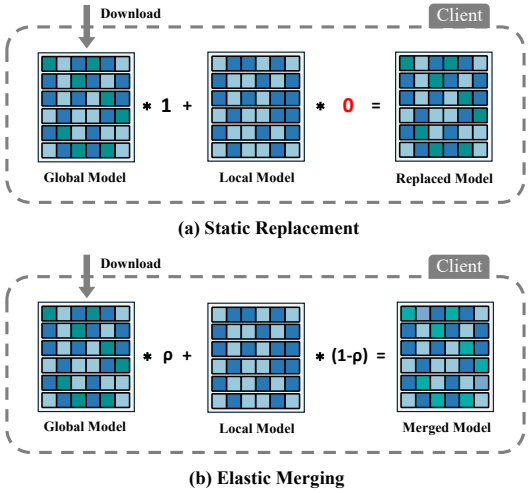

Figure 1: Traditional FR methods directly replace the local model with the global model downloaded from the server, while our FedEM can elastically merge both the global model and local model with the weight ρ, thereby delicately balancing collaboration and personalization.

For the first limitation, we theoretically reveal that aggregation can cause a loss of local information in the global model. This issue is exacerbated by the unique nature of FR tasks, which often involve millions of clients, several orders of magnitude more than typical federated learning scenarios. Besides, existing FR methods typically adopt a `static replacement` scheme, as illustrated in Figure 1(a), where the global model replaces the local one for both training and inference. Such a scheme propagates the impact of aggregation, causing optimization to deviate from the client-specific optimum and ultimately undermining local personalization. We refer to this issue as the aggregation bottleneck.

*Corresponding author.

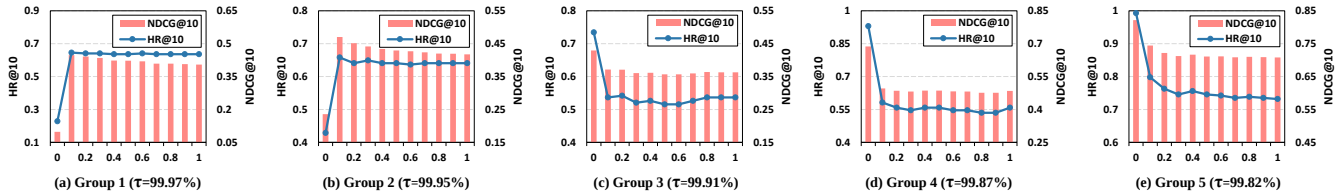

Figure 2: The performance of different client groups under varying merging weights ρ. The degree of data sparsity τ decreases from group 1 to group 5. Here, $\rho = 1$ stands for the traditional static replacement scheme.

For the second limitation, we explore a simpler and more general solution. Grounded in model merging theory (Zhou et al. 2024; Yang et al. 2024), we aim to compensate for local information loss with the off-the-shelf local model, thereby breaking the aggregation bottleneck. Specifically, we adopt an `elastic merging` scheme, as illustrated in Figure 1(b), where ρ denotes the weight of the global model parameters, while $1 - \rho$ stands for the weight of the local ones. We conduct empirical validation based on FedMF. To account for client heterogeneity, we divide the clients into five groups according to privacy-insensitive statistics of their local data, *e.g.*, data sparsity. For each group, we report the average performance metrics (HR@10 and NDCG@10) under different ρ. As shown in Figure 2, we observe that: (1) Static replacement scheme, *i.e.*, $\rho = 1$, yields suboptimal performance, which is consistent with our theoretical analysis; (2) Elastic merging scheme effectively enhances model performance. Owing to heterogeneity across clients, each group achieves optimal trade-offs under different merging weights ρ. Given this, we adopt client-specific merging that elastically adjusts to local personalization needs.

Taking both limitations into account, we propose a simple yet theoretically guaranteed pFR method called **Fed**erated recommendation via **E**lastic **M**erging (**FedEM**). It merges the aggregated global model with the off-the-shelf local model in a balanced way, effectively absorbing collaborative information while preserving client-specific personalization. To sum up, our main contributions are as follows:

- To the best of our knowledge, we are the first to theoretically analyze the aggregation bottleneck in FR scenarios, *i.e.*, the loss of local information caused by global aggregation, which further harms local personalization and leads to suboptimal performance.
- Based on model merging, we propose FedEM to bridge the gap between global collaboration and local personalization. Unlike existing heuristic methods, our approach is theoretically grounded and empirically validated.
- The proposed elastic merging module is model-agnostic and can be seamlessly integrated as a plug-in to enhance FR/pFR methods constrained by aggregation bottleneck.
- Extensive experiments demonstrate that FedEM consistently outperforms state-of-the-art (SOTA) methods.

Related Work

Personalized Federated Learning

Federated learning (FL) is a distributed learning paradigm to mitigate privacy issues (Huang et al. 2024; Guo et al. 2025). Traditional FL methods, such as FedAvg (McMahan et al. 2017), struggle to derive a global model generalized for each client when the local data is Non-IID (Huang et al. 2021). To that end, some personalized federated learning (pFL) methods aim to fine-tune the globally aggregated model for each client to obtain the personalized ones (Collins et al. 2021; Fallah, Mokhtari, and Ozdaglar 2020). For instance, FedALA exploits the general information in the lower layers of the global model to enhance the capability of feature extraction of the local model targeting federated vision tasks (Zhang et al. 2023c); Still other methods tend to locally learn a personalized model for each client (Li et al. 2021), *e.g.*, pFedMe (T Dinh, Tran, and Nguyen 2020) uses Moreau envelopes as the client regulation loss to decouple personalized model optimization from the global model learning; Other methods take a global view of personalized aggregation for each client (Ji et al. 2019; Zhang et al. 2021), *e.g.*, pFedGraph (Ye et al. 2023) optimizes the collaboration graph to perform weighted aggregation for different clients.

Personalized Federated Recommendation

In federated recommendation (FR), there is natural heterogeneity among clients due to their different preferences, thus necessitating personalization of the local model (Sun et al. 2024; Yin et al. 2024). Similar to pFL, personalized federated recommendation (pFR) methods can also be categorized into three research lines as follows. (1) Fine-tune the global model (Zhang et al. 2024c): PFedRec (Zhang et al. 2023a) personalizes both the score function and global model to alleviate data heterogeneity. FedCIA (Han et al. 2025) leverages the aggregated item similarity matrix to guide local model training, aiming to achieve global fine-tuning. (2) Learn an additional personalized model (Chen et al. 2025): FedRAP (Li, Long, and Zhou 2024) applies an additive model to item embeddings to enhance personalization. (3) Perform personalized global aggregation (Luo, Xiao, and Song 2022; Zhang et al. 2024b): FedFast (Muhammad et al. 2020) performs global aggregation by identifying representatives from different clusters based on user profile similarities. GPFedRec (Zhang et al. 2024a) performs graph-guided aggregation to recover inter-client correlations, generating client-specific global models. Unlike the above methods, which follow the traditional static replacement scheme and rely on heuristic personalization mechanisms, FedEM adopts an elastic merging scheme, offering a simple yet theoretically grounded solution. During global collaborative training, it leverages the off-the-shelf local model to enhance personalization for each client.

Preliminaries

Let $\mathcal{U}$ denote the set with n users/clients, $\mathcal{I}$ the set with m items. Then $\mathcal{D}_u = \{(u, i, r_{ui} | i \in \mathcal{I}_u)\}$ denotes the local interaction dataset of client u, where $\mathcal{I}_u$ is for the items observed by client u, and each entry $r_{ui} \in \{0, 1\}$ is for the label on item i by client u. The goal of FR is to predict $\hat{r}_{ui}$ of client u for each unobserved item $i \in \mathcal{I} \setminus \mathcal{I}_u$ on local devices. Formally, the global optimization objective over n clients of FR tasks is:

$$\min_{(\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n; \mathbf{Q}_1^l, \mathbf{Q}_2^l, \cdots, \mathbf{Q}_n^l)} \sum_{u=1}^{n} p_u \mathcal{L}_u \left(\mathbf{p}_u, \mathbf{Q}_u^l; \mathcal{D}_u\right), \quad (1)$$

where $\mathbf{p}_u \in \mathbb{R}^d$ and $\mathbf{Q}_u^l \in \mathbb{R}^{m \times d}$ denote local user embedding and local item embedding table for client u, respectively, d is the dimension of the embedding vector. In this work, we treat item embedding table $\mathbf{Q}$ as the intermediate model parameters for uploads and downloads, since it is the standard configuration in embedding-based FRs (Ammad-Ud-Din et al. 2019). The server aggregates local parameters $\mathbf{Q}_u^l$ with weight p_u to derive global model $\mathbf{Q}^g$, *e.g.*, $p_u = |\mathcal{D}_u| / \sum_{v=1}^{n} |\mathcal{D}_v|$ in FedAvg (McMahan et al. 2017) and $p_u = 1/n$ in FCF (Ammad-Ud-Din et al. 2019). $\mathcal{L}_u$ is the task-specific objective to facilitate local training.

In this work, we focus on the typical recommendation task with implicit feedback, where the rating r_{ui} of item i by user u is either 1 or 0, indicating interested versus uninterested interaction, respectively. Therefore, we adopt the binary cross-entropy (BCE) loss (He et al. 2017), which is commonly used in binary-value problems, as the local objective $\mathcal{L}_u$. Formally, the BCE loss is defined as:

$$\begin{aligned} \mathcal{L}_u(\mathbf{p}_u, \mathbf{Q}_u^l; \mathcal{D}_u) = - \sum_{(u,i,r_{ui}) \in \mathcal{D}_u} \big[& r_{ui} \log \hat{r}_{ui} \\ & + (1 - r_{ui}) \log (1 - \hat{r}_{ui}) \big], \end{aligned} \quad (2)$$

where $\hat{r}_{ui} = \phi(\mathbf{p}_u \mathbf{q}_i^\top)$ denotes the predicted rating of item i by user u, where $\mathbf{q}_i \in \mathbf{Q}_u^l$ and $\phi(\cdot)$ is the Sigmoid function.

Methodology

Motivation

Aggregation Bottleneck. Most existing FR methods adopt fixed-weight aggregation on the server side to enable collaborative training across clients. Taking FCF (Ammad-Ud-Din et al. 2019) for example, that is, $p_u = 1/n$, the global model derived by aggregation in round t can be represented as:

$$\begin{aligned} \mathbf{Q}^{g(t)} &= \frac{1}{n} \sum_{u=1}^{n} \mathbf{Q}_u^{l(t)} = \frac{1}{n} \mathbf{Q}_u^{l(t)} + \frac{n-1}{n} \mathbf{Q}_r^{g(t)}, \\ \text{where} \quad \mathbf{Q}_r^{g(t)} &= \frac{1}{n-1} \sum_{v \neq u}^{n} \mathbf{Q}_v^{l(t)}. \end{aligned} \quad (3)$$

Here, $\mathbf{Q}_r^{g(t)}$ denotes the aggregated model over the remaining clients except for ego Client u. However, unlike typical FL settings, FR tasks often involve tens of thousands or even millions of clients. Therefore, we have:

$$\lim_{n \to \infty} \frac{1}{n} = 0, \ \lim_{n \to \infty} \frac{n-1}{n} = 1, \quad (4)$$

which means $\mathbf{Q}^{g(t)} \approx \mathbf{Q}_r^{g(t)}$, indicating a loss of local information in the global model. At this point, for client u, adopting the static replacement scheme may lead to the aggregation bottleneck formally established in Lemma 1.

Assumption 1 (Smoothness and Strong Convexity)**.** *The local objective function $\mathcal{L}_u(\mathbf{Q})$ is assumed to be continuously differentiable, L-smooth, and μ-strongly convex with respect to the model parameter $\mathbf{Q}$. This assumption can be approximately satisfied in our setting by incorporating ℓ_2 regularization into the BCE loss (Boyd and Vandenberghe 2004; Bottou, Curtis, and Nocedal 2018). Such assumptions are also standard in theoretical analyses of federated learning (McMahan et al. 2017; T Dinh, Tran, and Nguyen 2020).*

Lemma 1 (Performance Bottleneck of Global Aggregation)**.** *Based on Assumption 1, let $\mathbf{Q}_u^*$ denote the local optimal model. Then, following the static replacement scheme, i.e., using the global model $\mathbf{Q}^{g(t)}$ for local training, may lead to:*

$$\langle \nabla \mathcal{L}_u(\mathbf{Q}^{g(t)}), \mathbf{Q}^{g(t)} - \mathbf{Q}_u^* \rangle \leq 0, \quad (5)$$

which shows that the globally aggregated model can cause training optimization to deviate from the client-specific optimum, thereby undermining local personalization. This issue becomes more pronounced under the high inherent client heterogeneity in FR, leading to a performance bottleneck.

Framework

As illustrated in Figure 3, the overall framework of FedEM consists of the following main steps in each round t:

❶ Elastic Merging: client u merges aggregated global model $\mathbf{Q}_u^{g(t-1)}$ and off-the-shelf local model $\mathbf{Q}_u^{l(t-1)}$ to derive a merged model $\mathbf{Q}_u^{m(t)}$ at the beginning of this round t. Here, existing FR methods follow the static replacement scheme and directly discard the local model.

❷ Local Training: client u utilizes its own data $\mathcal{D}_u$ to update the merged model $\mathbf{Q}_u^{m(t)}$ and local parameters like $\mathbf{p}_u^{(t-1)}$. Then, client u uploads the updated model $\mathbf{Q}_u^{l(t)}$ to the server.

❸ Global Aggregation: the server performs aggregation based on the similarity over uploaded $\{\mathbf{Q}_1^{l(t)}, \cdots, \mathbf{Q}_n^{l(t)}\}$, generating the global model $\mathbf{Q}_u^{g(t)}$ for client u.

Notably, both $\mathbf{Q}_u^{g(0)}$ and $\mathbf{Q}_u^{l(0)}$ represent randomly initialized models when $t = 1$. After T rounds above, each client can get its own personalized model.

Elastic Merging

Theoretical Analysis. Existing pFR methods, *e.g.*, PFedRec (Zhang et al. 2023a), FedRAP (Li, Long, and Zhou 2024), and FedCIA (Han et al. 2025), heuristically design additional personalization mechanisms. Although empirically effective, these methods fail to identify the aggregation bottleneck as the underlying cause of insufficient personalization in FR, thereby limiting their potential for further improvement. Additionally, their complexity and limited compatibility pose challenges for practical deployment.

Inspired by the model merging theory (Zhou et al. 2024; Yang et al. 2024), we merge the global model $\mathbf{Q}^{g(t)}$ and the

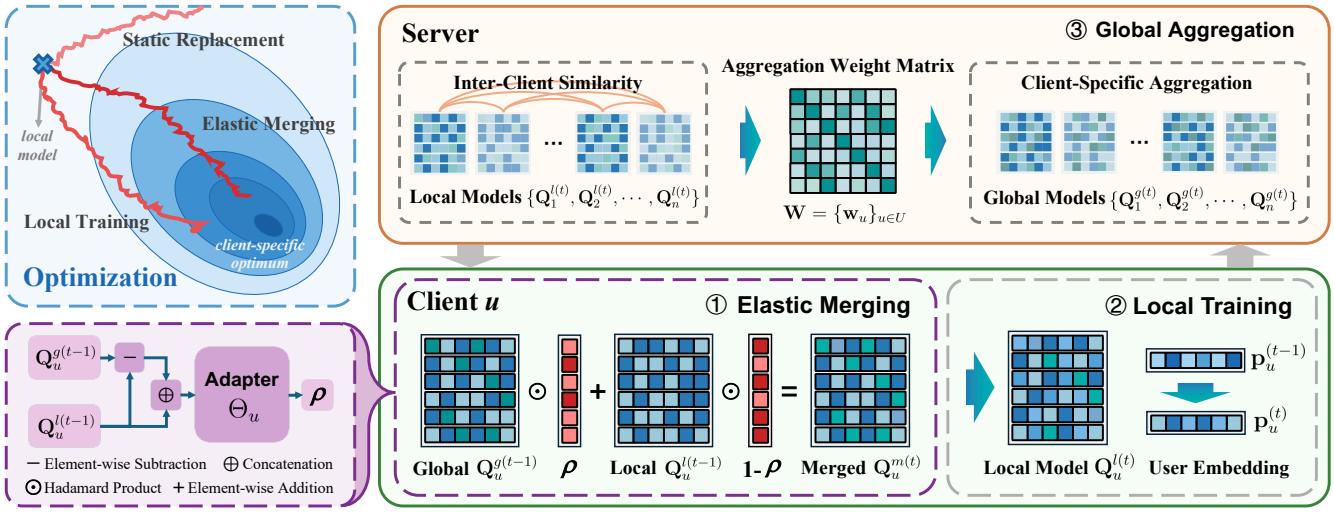

Figure 3: The framework of FedEM. Unlike the static replacement scheme in existing FR methods, we propose an elastic merging scheme to mitigate the optimization deviation caused by global aggregation, effectively breaking the performance bottleneck. Additionally, the server can perform similarity-based aggregation to further alleviate the aggregation bottleneck.

off-the-shelf local model $\mathbf{Q}_u^{l(t)}$ to overcome the aggregation bottleneck. For embedding-based FRs, $\forall \rho \in [0, 1]$,

$$f(\rho \mathbf{Q}^{g(t)} + (1-\rho)\mathbf{Q}_u^{l(t)}) = \rho f(\mathbf{Q}^{g(t)}) + (1-\rho) f(\mathbf{Q}_u^{l(t)}), \tag{6}$$

where $f(\mathbf{Q})$ denotes the predicted rating, *i.e.*, $\mathbf{p}_u \cdot \mathbf{Q}$. Thus, the two models $\mathbf{Q}^{g(t)}$ and $\mathbf{Q}_u^{l(t)}$ satisfy the model merging conditions in the parameter space (Zhou et al. 2024). We define the merged model as:

$$\mathbf{Q}_u^{m(t+1)} = \rho \mathbf{Q}^{g(t)} + (1-\rho)\mathbf{Q}_u^{l(t)}, \tag{7}$$

which is used as the initialization for local training in round $t+1$. By following the elastic merging scheme, we compensate for the loss of local information in the aggregated model. The merged model benefits from global collaborative training while mitigating the risk of deviating from the client-specific optimum, thus preserving personalization.

Lemma 2 (Compensation Effect of Elastic Merging)**.** *Based on Assumption 1, the merged model can compensate for the optimization deviation caused by aggregation, guiding the update toward the client-specific optimum, i.e., it satisfies:*

$$\begin{aligned} &\langle \nabla \mathcal{L}_u(\mathbf{Q}_u^{m(t+1)}), \mathbf{Q}_u^{m(t+1)} - \mathbf{Q}_u^* \rangle \\ &\geq (1-\rho)\langle \nabla \mathcal{L}_u(\mathbf{Q}_u^{l(t)}), \mathbf{Q}_u^{l(t)} - \mathbf{Q}_u^* \rangle - \rho C, \end{aligned} \tag{8}$$

where C denotes the deviation introduced by global aggregation. This deviation can be lower-bounded by both the state of the local model $\mathbf{Q}_u^{l(t)}$ and the global-local discrepancy $\mathbf{Q}_u^{\Delta(t)} = \mathbf{Q}^{g(t)} - \mathbf{Q}_u^{l(t)}$. The first term on the right-hand side of the inequality, $A = \langle \nabla \mathcal{L}_u(\mathbf{Q}_u^{l(t)}), \mathbf{Q}_u^{l(t)} - \mathbf{Q}_u^* \rangle > 0$, reflects the preservation of client-specific optimization information. For effective local personalization, the update must remain aligned with the client-specific optimum, which requires the left-hand side of the inequality to be positive. To achieve this, we need to adaptively control the merging weight $0 \leq \rho < \frac{A}{A+C}$ based on the deviation C in each round. Accordingly, we propose the Elastic Merging module, which learns ρ from $\mathbf{Q}_u^{l(t)}$ and $\mathbf{Q}_u^{\Delta(t)}$, ensuring client-wise alignment and mitigating the aggregation bottleneck.

Detailed proofs of all lemmas are deferred to ***Appendix A***.

Practical Implementation. On the client side, we introduce the Elastic Merging (EM) module to dynamically preserve personalized information from the local model during global collaborative training. Furthermore, we utilize the vector $\boldsymbol{\rho} \in \mathbb{R}^m$ for fine-grained merging at the item level. Based on the analysis in Lemma 2, we first compute the global-local model discrepancy $\mathbf{Q}_u^{\Delta(t-1)} = \mathbf{Q}_u^{g(t-1)} - \mathbf{Q}_u^{l(t-1)}$, and then concatenate it with the local model $\mathbf{Q}_u^{l(t-1)}$ to obtain $\mathbf{Q}_u^{\text{cat}(t-1)} = \mathbf{Q}_u^{\Delta(t-1)} \oplus \mathbf{Q}_u^{l(t-1)}$. We feed $\mathbf{Q}_u^{\text{cat}(t-1)} \in \mathbb{R}^{m \times 2d}$ into the core component of the EM module, *i.e.*, the adapter, to generate the elastic vector $\boldsymbol{\rho}$ for each client u.

Here, we implement the adapter with an L-layer multi-layer perceptron (MLP). Formally, we have:

$$\boldsymbol{\rho} = \phi_{output}(\Theta_u^L \bullet (\cdots \phi(\Theta_u^2 \bullet (\phi(\Theta_u^1 \bullet \mathbf{Q}_u^{\text{cat}(t-1)}))))), \tag{9}$$

where $\phi(\cdot)$ and $\phi_{output}(\cdot)$ denote the mapping functions for the intermediate and output layers, respectively. Here, we employ ReLU for ϕ and Sigmoid for ϕ_{output} to ensure that each element of vector $\boldsymbol{\rho}$ is in the range $[0, 1]$. Besides, notation $\bullet$ denotes the matrix product operation, and Θ_u^i denotes the parameters of the i-th layer. Then, the merged model can be formulated as:

$$\begin{aligned} \mathbf{Q}_u^{m(t+1)} &= \boldsymbol{\rho} \odot \mathbf{Q}_u^{g(t-1)} + (\mathbf{1} - \boldsymbol{\rho}) \odot \mathbf{Q}_u^{l(t-1)} \\ &= \mathbf{Q}_u^{l(t-1)} + \boldsymbol{\rho} \odot \mathbf{Q}_u^{\Delta(t-1)}, \end{aligned} \tag{10}$$

where $\odot$ denotes the Hadamard product. Hence, by preserving the local information in $\mathbf{Q}_u^{l(t)}$, $\boldsymbol{\rho}$ serves to selectively absorb the collaborative information in $\mathbf{Q}_u^{\Delta(t)}$, thus preventing the aforementioned aggregation bottleneck.

Specifically, the adapter parameters Θ_u can be updated using stochastic gradient descent (SGD) as follows:

$$\Theta_u = \Theta_u - \beta \cdot \frac{\partial \mathcal{L}_u(\mathbf{p}_u^{(t-1)}, \mathbf{Q}_u^{m(t)}; \mathcal{D}_u)}{\partial \Theta_u}, \quad (11)$$

where β is the learning rate for the adapter. Notably, only the parameters Θ_u are updated with $\boldsymbol{\rho}$ varied in this step, while other model parameters, *i.e.*, $\mathbf{p}_u^{(t-1)}$, $\mathbf{Q}_u^{l(t-1)}$ and $\mathbf{Q}_u^{g(t-1)}$, are frozen. Unlike other FR methods, $\mathbf{Q}_u^{m(t)}$ is utilized to initialize the local model for local training rather than $\mathbf{Q}_u^{g(t-1)}$.

Local Training

The objective of local training is to update the user embedding $\mathbf{p}_u^{(t-1)}$ and the item embedding table $\mathbf{Q}_u^{m(t)}$ to obtain $\mathbf{p}_u^{(t)}$ and $\mathbf{Q}_u^{l(t)}$. Note that the adapter Θ_u is not involved during this process. Formally, the update can be written as:

$$\mathbf{p}_u = \mathbf{p}_u - \eta \cdot \frac{\partial \mathcal{L}_u}{\partial \mathbf{p}_u}, \quad \mathbf{Q}_u^l = \mathbf{Q}_u^l - \eta \cdot \frac{\partial \mathcal{L}_u}{\partial \mathbf{Q}_u^l}, \quad (12)$$

where η is the learning rate for local training. To protect user privacy, after local training, client u only uploads $\mathbf{Q}_u^{l(t)}$ to the server for global aggregation.

Global Aggregation

As shown in Lemma 1, traditional fixed-weight aggregation is ineffective in FR, and the aggregation bottleneck is further exacerbated by inherent client heterogeneity. In addition to local elastic merging, we further perform global similarity-based aggregation to alleviate the impact of such heterogeneity. Inspired by the previous work (Ye et al. 2023) in FL, we additionally incorporate client similarity, generating a client-specific weight vector $\mathbf{w}_u = [w_{u1}, w_{u2}, \ldots, w_{un}] \in \mathbb{R}^n$ to conduct tailored aggregation for each client u. Specifically, $\mathbf{w}_u$ is obtained by optimizing the following objective:

$$\mathcal{L}_g = \sum_{v=1}^{n} \left((w_{uv} - p_v)^2 + \alpha (w_{uv} - \sigma(\mathbf{Q}_u^l, \mathbf{Q}_v^l))^2 \right), \quad (13)$$

where $p_u = |\mathcal{D}_u| / \sum_{v=1}^{n} |\mathcal{D}_v|$ denotes the aggregation weight used in the backbone FedMF (Chai et al. 2020). Here, $\sigma(\mathbf{Q}_u^l, \mathbf{Q}_v^l) = 1/(1 + \|\mathbf{Q}_u^l - \mathbf{Q}_v^l\|^2)$ denotes the similarity function. This term ensures that the aggregation weight w_{uv} increases when the two client models are highly similar. Besides, α is a similarity-related hyperparameter. After global aggregation, each client u can download the global model $\mathbf{Q}_u^{g(t)} = \sum_{v=1}^{n} w_{uv} \mathbf{Q}_v^{l(t)}$ to start the next round $t+1$.
The detailed procedure of our algorithm is in ***Appendix B***.

Discussions

Complexity Analysis

Given n clients and m items, with embedding dimension d, the computational complexity of the local backbone model is $\mathcal{O}((m+1)d)$. Our proposed EM module is implemented as an L-layer MLP, whose complexity is $\mathcal{O}(Ld^2)$. Since $m \gg d > L$ in practice, the EM module introduces negligible overhead to the local device, making it well-suited for on-device recommendation.

Privacy Analysis

FedEM naturally preserves user privacy under the federated paradigm. Moreover, the local merging parameters Θ_u and $\boldsymbol{\rho}$ are not shared with the server, reducing the risk of privacy leakage (Chai et al. 2020). To further enhance privacy, we incorporate local differential privacy (LDP) (Qi, Wang, and Huang 2024) into our method. The privacy budget ε is guaranteed by $\mathcal{S}_u/\delta$, where δ is the noise strength and $\mathcal{S}_u$ denotes the global sensitivity of client u. We upper-bound $\mathcal{S}_u$ by $2p_u\eta Z$, where Z is the gradient clipping threshold.
Detailed discussions can be found in ***Appendix C***.

Experiments

Experimental Settings

Datasets. We evaluate our proposed method on four datasets with varying client scale and data sparsity: Filmtrust (Guo, Zhang, and Yorke-Smith 2013), ML-100K (Harper and Konstan 2015), ML-1M (Harper and Konstan 2015), and LastFM-2K (Cantador, Brusilovsky, and Kuflik 2011).
Evaluation Protocols. We follow the popular *leave-one-out* evaluation (Bayer et al. 2017; He et al. 2017), and report the performance by *Hit Ratio* (HR@K) and *Normalized Discounted Cumulative Gain* (NDCG@K) (He et al. 2015).
Compared Baselines. We compare FedEM with SOTA centralized methods, *e.g.*, MF (Koren, Bell, and Volinsky 2009), NCF (He et al. 2017), LightGCN (He et al. 2020), and federated methods, *e.g.*, FedMF (Chai et al. 2020), FedNCF (Perifanis and Efraimidis 2022), FedFast (Muhammad et al. 2020), PFedRec (Zhang et al. 2023a), CoLR (Nguyen et al. 2024), GPFedRec (Zhang et al. 2024a), FedRAP (Li, Long, and Zhou 2024), FedCIA (Han et al. 2025).
Implementation Details. For a fair comparison, we set the global round $T = 100$, batch size $B = 256$, and embedding dimension $d = 16$ for all methods. Besides, we adopt the default/optimal settings for other hyperparameters as reported in their original papers. All baseline methods reach convergence under the given settings.
Detailed experimental settings are provided in ***Appendix D***.

Overall Performance

The performance comparison of different methods is summarized in Table 1. FedEM outperforms centralized methods on most datasets. As a pFR method, FedEM customizes item embeddings for each client, enabling it to better capture user preferences compared to centralized methods that share the same item embeddings across all clients. FedEM also surpasses other federated methods. Existing pFR methods heuristically design additional personalization mechanisms without tackling the aggregation bottleneck, *i.e.*, the loss of local personalization caused by global aggregation. In contrast, our method effectively and efficiently leverages off-the-shelf personalized information from the local model to bridge this gap. Additionally, on relatively dense datasets such as ML-100K and ML-1M, pFR methods, including FedEM, can surpass centralized methods due to the locally sufficient training of personalized models. However, for highly sparse datasets such as LastFM-2K, with a sparsity level of up to 99.07%, the federated setting severely limits local

Datasets	Methods	CenRec			FedRec								Ours
		MF	NCF	LightGCN	FedMF	FedNCF	FedFast	PFedRec	CoLR	GPFedRec	FedRAP	FedCIA	FedEM
Filmtrust	HR@10	0.6936	0.6856	0.7804	0.6577	0.6597	0.6637	0.6756	0.6377	0.6836	0.8024	0.8543	**0.8932**
	NDCG@10	0.5341	0.5476	0.6475	0.5290	0.5337	0.4951	0.5398	0.5105	0.5425	0.5455	0.6992	**0.7701**
ML-100K	HR@10	0.6585	0.6066	0.8356	0.4889	0.4252	0.4687	0.6882	0.4952	0.7010	0.8897	0.9512	**0.9958**
	NDCG@10	0.3781	0.3398	0.5754	0.2721	0.2290	0.2702	0.3913	0.2772	0.4069	0.7950	0.7741	**0.9427**
ML-1M	HR@10	0.6053	0.5897	0.8217	0.4871	0.4230	0.4137	0.6730	0.4533	0.6776	0.8619	0.9012	**0.9507**
	NDCG@10	0.3376	0.3325	0.5478	0.2733	0.2285	0.2333	0.3898	0.2475	0.3973	0.7661	0.7890	**0.8392**
LastFM-2K	HR@10	0.8440	0.7904	0.8463	0.5934	0.4996	0.5225	0.7833	0.5335	**0.7975**	0.6210	0.7108	0.7935
	NDCG@10	0.6161	0.6024	0.6890	0.3963	0.3307	0.3239	0.6822	0.3580	0.6690	0.5923	0.6601	**0.7151**

Table 1: Performance comparison on four datasets, reported by HR@10 and NDCG@10. CenRec and FedRec represent centralized and federated recommendation methods, respectively. The best FedRec results are bold, and the second ones are underlined.

data availability, making it difficult to outperform centralized methods. Nevertheless, FedEM still achieves superior performance compared to all pFR methods under such challenging scenarios, demonstrating its general applicability.

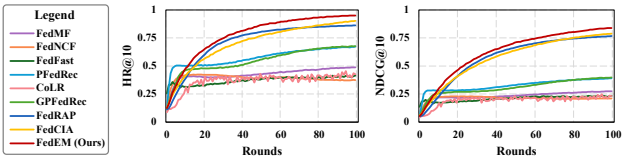

Figure 4: Convergence comparison on the ML-1M dataset.

We further compare the convergence of different methods, with results on the ML-1M dataset shown in Figure 4. Compared with other methods, FedEM demonstrates a faster convergence in the early stages and achieves more stable and superior performance in the later stages. This is attributed to the elastic merging scheme, which retains the local information from the previous round, thereby enhancing both performance and convergence stability.

Datasets	Metrics	FedMF w/ SR	FedSim w/ SR	FedSim w/ SM	FedSim w/ DM	FedSim w/ EM
Filmtrust	HR@10	0.6577	0.7206	0.7774	0.8433	**0.8932**
	NDCG@10	0.5290	0.5501	0.5959	0.7051	**0.7701**
ML-100K	HR@10	0.4889	0.6320	0.7190	0.9427	**0.9958**
	NDCG@10	0.2721	0.4457	0.5277	0.8175	**0.9427**
ML-1M	HR@10	0.4871	0.6166	0.7219	0.9078	**0.9507**
	NDCG@10	0.2733	0.4103	0.5278	0.7839	**0.8392**
LastFm-2K	HR@10	0.5934	0.7258	0.7455	0.7912	**0.7935**
	NDCG@10	0.3963	0.6498	0.6633	0.7131	**0.7151**

Table 2: Ablation study results. The best results are in bold.

Ablation Study

Component analysis. Based on the backbone FedMF, we apply similarity-based aggregation on the server side and denote this variant as FedSim. Moreover, to validate the rationality and effectiveness of the EM module, we introduce the following variants: (1) Static Replacement (SR): Replace the local model with the global model for local training. (2) Static Merging (SM): Set a fixed scalar ρ shared across all clients to merge the global and local models. (3) Dynamic Merging (DM): Each client adopts a local adapter to generate a personalized scalar ρ for model merging. (4) Elastic Merging (EM): The adapter generates a vector $\boldsymbol{\rho}$ to perform fine-grained model merging at the item level for each client.

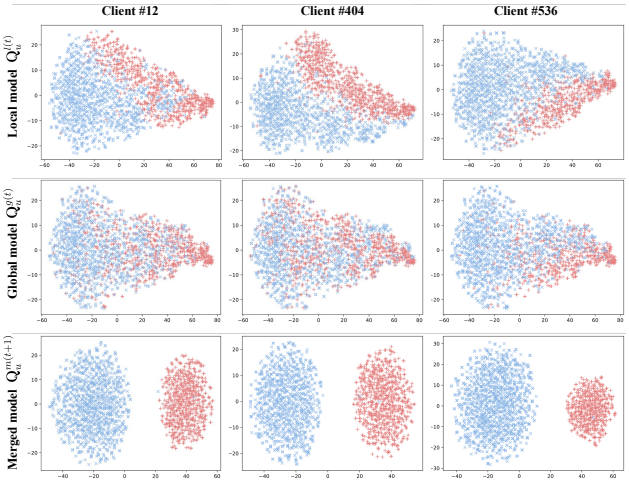

Figure 5: Model visualization of item embeddings on ML-100K. Items interacted with by the user are marked with red $+$, while unobserved items are marked with blue $\times$.

From the results in Table 2, we observe that incorporating similarity into global aggregation helps mitigate the negative impact of client heterogeneity on local personalization. Focusing on local strategies, the SR scheme adopted by existing methods directly discards the local model, leading to suboptimal performance. By merging the global and local models, SM retains some personalized information and achieves better results. However, SM applies the same merging weight to all clients, neglecting client heterogeneity and diverse personalization needs. DM improves upon SM by introducing the adapter to generate client-specific weights, enabling more tailored merging. Building on DM, EM further leverages the embedding-based nature of recommendation models to perform fine-grained merging at the item level. This design allows for a more balanced fusion of collaborative information from global aggregation and the off-the-shelf personalized information from local training, offering a more efficient and effective solution.

Datasets	Metrics	FedMF			FedNCF			PFedRec			GPFedRec			FedRAP		
		w/o EM	w/ EM	Improv.	w/o EM	w/ EM	Improv.	w/o EM	w/ EM	Improv.	w/o EM	w/ EM	Improv.	w/o EM	w/ EM	Improv.
Filmtrust	HR@10	0.6577	0.7804	↑ 18.66%	0.6597	0.7315	↑ 10.88%	0.6756	0.9691	↑ 43.44%	0.6836	0.8723	↑ 27.60%	0.8024	0.8044	↑ 0.25%
	NDCG@10	0.5290	0.6426	↑ 21.47%	0.5337	0.6852	↑ 28.39%	0.5398	0.8544	↑ 58.28%	0.5425	0.6706	↑ 23.61%	0.5455	0.5498	↑ 0.79%
ML-100K	HR@10	0.4889	0.8759	↑ 79.16%	0.4252	0.7529	↑ 77.07%	0.6882	0.9905	↑ 43.93%	0.7010	0.9247	↑ 31.91%	0.8897	0.9735	↑ 9.42%
	NDCG@10	0.2721	0.7533	↑ 176.85%	0.2290	0.7162	↑ 212.75%	0.3913	0.9238	↑ 136.08%	0.4069	0.6938	↑ 70.51%	0.7950	0.8964	↑ 12.75%
ML-1M	HR@10	0.4871	0.8199	↑ 68.32%	0.4230	0.7606	↑ 79.81%	0.6730	0.9656	↑ 43.48%	0.6776	0.8823	↑ 30.21%	0.8619	0.9538	↑ 10.66%
	NDCG@10	0.2733	0.6606	↑ 141.71%	0.2285	0.6979	↑ 205.43%	0.3898	0.7978	↑ 104.67%	0.3973	0.6777	↑ 70.58%	0.7661	0.8648	↑ 12.88%
LastFM-2K	HR@10	0.5934	0.7013	↑ 18.18%	0.4925	0.6564	↑ 33.28%	0.7833	0.8810	↑ 12.47%	0.7975	0.8385	↑ 5.14%	0.6210	0.6320	↑ 1.77%
	NDCG@10	0.3963	0.6061	↑ 52.94%	0.3215	0.5445	↑ 69.36%	0.6822	0.8052	↑ 18.03%	0.6690	0.7475	↑ 11.73%	0.5923	0.5869	–

Table 3: Performance improvement by integrating the Elastic Merging (EM) module into existing FR/pFR baselines. "Improv." indicates the performance gain over the original baselines.

Model Visualization. We incorporate the EM module into FedMF and perform t-SNE visualization of the local, global, and merged models, as shown in Figure 5. The locally trained model contains rich personalized information, with the preferred items being relatively concentrated. Although global aggregation introduces more diverse collaborative information into the model, it also disrupts user preferences. Using such a global model for local training can harm personalization. In contrast, the EM module performs model merging at the item level, effectively leveraging the off-the-shelf personalized information from the local model while selectively incorporating collaborative information from the global model, leading to better performance.

Compatibility Study

Our proposed EM module can be seamlessly integrated as a plug-in into existing FR/pFR methods, aiming to overcome the performance bottleneck caused by global aggregation. Specifically, we take FedMF, FedNCF, PFedRec, GPFedRec, and FedRAP as examples, where elastic merging is performed after the client downloads the global model, instead of following static replacement. As shown in Table 3, all methods exhibit significant performance improvements after incorporating the EM module, validating both the limitations of the static replacement and the effectiveness of elastic merging. The EM module is lightweight and easily integrable, incurring little to no overhead on local clients and demonstrating good compatibility with existing models.

Hyperparameter Analysis

Adapter architecture. In our experiments, we set the embedding dimension to $d = 16$, and by default, the adapter is implemented as an MLP with three hidden layers, following the structure $[32, 16, 8, 1]$. Additionally, we explore MLPs with varying numbers of hidden layers, and the corresponding results are illustrated in Figure 6. When the number of MLP layers L is small, the adapter struggles to effectively balance the local and global models, leading to suboptimal results. As L increases, the model performance does not improve significantly, but the adapter becomes more complex. The default structure achieves a good trade-off between performance and architectural simplicity.

Similarity Coefficient. During global aggregation, we find that setting the similarity coefficient α around 1 can effectively alleviate the negative impact of heterogeneity discussed in Lemma 1 and improve overall model performance.

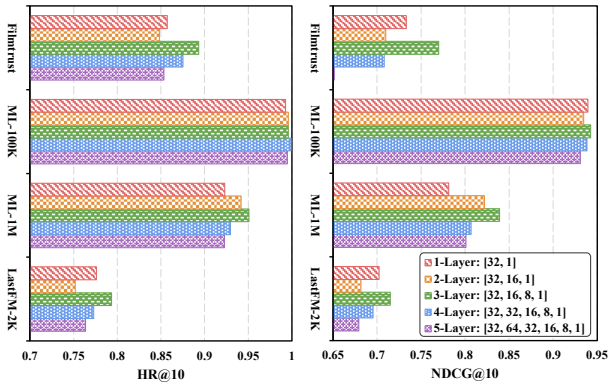

Figure 6: Performance under different adapter architectures.

Privacy Protection

We evaluate the performance of privacy-enhanced FedEM by incorporating the local differential privacy (LDP) strategy. As shown in Table 4, the performance on all datasets slightly degrades with LDP, but remains within an acceptable range and still outperforms other baselines, demonstrating the robustness of our method.

Datasets	Filmtrust		ML-100K		ML-1M		LastFM-2K	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
w/o LDP	0.8932	0.7701	0.9958	0.9427	0.9507	0.8392	0.7935	0.7151
w/ LDP	0.8802	0.7353	0.9936	0.9399	0.9392	0.8193	0.7707	0.6930
Degradation	↓ 1.46%	↓ 4.52%	↓ 0.22%	↓ 0.30%	↓ 1.21%	↓ 2.37%	↓ 2.87%	↓ 3.09%

Table 4: Performance of applying LDP to our method.

Extensive experimental results are provided in ***Appendix E***.

Conclusion

In this work, we experimentally and theoretically identify a performance bottleneck in FR caused by global aggregation. Unlike existing methods that heuristically design personalization mechanisms, we address this bottleneck directly with a simple yet effective method called FedEM. Grounded in model merging theory, FedEM exploits the off-the-shelf local models to compensate for the aggregated global model. It elastically strikes a balance between global collaboration and local personalization, achieving excellent performance and compatibility compared to other SOTA methods.

References

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

Bayer, I.; He, X.; Kanagal, B.; and Rendle, S. 2017. A generic coordinate descent framework for learning from implicit feedback. In *WWW*, 1341–1350.

Bottou, L.; Curtis, F. E.; and Nocedal, J. 2018. Optimization methods for large-scale machine learning. *SIAM review*, 60(2): 223–311.

Boyd, S. P.; and Vandenberghe, L. 2004. *Convex optimization*. Cambridge university press.

Cantador, I.; Brusilovsky, P.; and Kuflik, T. 2011. Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011). In *RecSys*, 387–388.

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

Chen, G.; Zhang, X.; Su, Y.; Lai, Y.; Xiang, J.; Zhang, J.; and Zheng, Y. 2023. Win-win: a privacy-preserving federated framework for dual-target cross-domain recommendation. In *AAAI*, 4149–4156.

Chen, J.; Zhang, H.; Li, H.; Zhang, C.; Li, Z.; and Li, Y. 2025. Beyond Personalization: Federated Recommendation with Calibration via Low-rank Decomposition. *arXiv preprint arXiv:2506.09525*.

Choi, W.-S.; Tomei, M.; Vicarte, J. R. S.; Hanumolu, P. K.; and Kumar, R. 2018. Guaranteeing local differential privacy on ultra-low-power systems. In *ISCA*, 561–574.

Collins, L.; Hassani, H.; Mokhtari, A.; and Shakkottai, S. 2021. Exploiting shared representations for personalized federated learning. In *ICML*, 2089–2099.

Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. 2006. Calibrating noise to sensitivity in private data analysis. In *TCC*, 265–284.

Fallah, A.; Mokhtari, A.; and Ozdaglar, A. 2020. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In *NeurIPS*, 3557–3568.

Guo, G.; Zhang, J.; and Yorke-Smith, N. 2013. A Novel Bayesian Similarity Measure for Recommender Systems. In *IJCAI*, 2619–2625.

Guo, X.; Yu, K.; Cui, L.; Yu, H.; and Li, X. 2025. Federated Causally Invariant Feature Learning. In *AAAI*, 16978–16986.

Han, M.; Li, D.; Xia, J.; Liu, J.; Gu, H.; Zhang, P.; Gu, N.; and Lu, T. 2025. FedCIA: Federated Collaborative Information Aggregation for Privacy-Preserving Recommendation. In *SIGIR*, 1687–1696.

Harper, F. M.; and Konstan, J. A. 2015. The movielens datasets: History and context. *ACM TIIS*, 5(4): 1–19.

He, X.; Chen, T.; Kan, M.-Y.; and Chen, X. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *CIKM*, 1661–1670.

He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; and Wang, M. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*, 639–648.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *WWW*, 173–182.

Huang, W.; Ye, M.; Shi, Z.; Wan, G.; Li, H.; Du, B.; and Yang, Q. 2024. Federated learning for generalization, robustness, fairness: A survey and benchmark. *IEEE TPAMI*, 46(12): 9387–9406.

Huang, Y.; Chu, L.; Zhou, Z.; Wang, L.; Liu, J.; Pei, J.; and Zhang, Y. 2021. Personalized cross-silo federated learning on non-iid data. In *AAAI*, 7865–7873.

Ji, S.; Pan, S.; Long, G.; Li, X.; Jiang, J.; and Huang, Z. 2019. Learning private neural language modeling with attentive aggregation. In *IJCNN*, 1–8.

Karimireddy, S. P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; and Suresh, A. T. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, 5132–5143. PMLR.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer*, 30–37.

Li, T.; Hu, S.; Beirami, A.; and Smith, V. 2021. Ditto: Fair and robust federated learning through personalization. In *ICML*, 6357–6368.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2: 429–450.

Li, Z.; Long, G.; and Zhou, T. 2024. Federated recommendation with additive personalization. In *ICLR*.

Luo, S.; Xiao, Y.; and Song, L. 2022. Personalized federated recommendation via joint representation learning, user clustering, and model adaptation. In *CIKM*, 4289–4293.

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

Minto, L.; Haller, M.; Livshits, B.; and Haddadi, H. 2021. Stronger privacy for federated collaborative filtering with implicit feedback. In *RecSys*, 342–350.

Muhammad, K.; Wang, Q.; O'Reilly-Morgan, D.; Tragos, E.; Smyth, B.; Hurley, N.; Geraci, J.; and Lawlor, A. 2020. Fedfast: Going beyond average for faster training of federated recommender systems. In *SIGKDD*, 1234–1242.

Nguyen, N.-H.; Nguyen, T.-A.; Nguyen, T.; Hoang, V. T.; Le, D. D.; and Wong, K.-S. 2024. Towards efficient communication and secure federated recommendation system via low-rank training. In *WWW*, 3940–3951.

Perifanis, V.; and Efraimidis, P. S. 2022. Federated neural collaborative filtering. *Knowledge-Based Systems*, 242: 108441.

Qi, T.; Wang, H.; and Huang, Y. 2024. Towards the Robustness of Differentially Private Federated Learning. In *AAAI*, 19911–19919.

Shalev-Shwartz, S.; and Ben-David, S. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Sun, Z.; Xu, Y.; Liu, Y.; He, W.; Kong, L.; Wu, F.; Jiang, Y.; and Cui, L. 2024. A survey on federated recommendation systems. *IEEE TNNLS*, 1–15.

T Dinh, C.; Tran, N.; and Nguyen, J. 2020. Personalized federated learning with moreau envelopes. In *NeurIPS*, 21394–21405.

Wei, K.; Li, J.; Ding, M.; Ma, C.; Yang, H. H.; Farokhi, F.; Jin, S.; Quek, T. Q.; and Poor, H. V. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE TIFS*, 3454–3469.

Yang, E.; Shen, L.; Guo, G.; Wang, X.; Cao, X.; Zhang, J.; and Tao, D. 2024. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*.

Ye, R.; Ni, Z.; Wu, F.; Chen, S.; and Wang, Y. 2023. Personalized federated learning with inferred collaboration graphs. In *ICML*, 39801–39817.

Yin, H.; Qu, L.; Chen, T.; Yuan, W.; Zheng, R.; Long, J.; Xia, X.; Shi, Y.; and Zhang, C. 2024. On-Device Recommender Systems: A Comprehensive Survey. *arXiv preprint arXiv:2401.11441*.

Zhang, C.; Li, S.; Xia, J.; Wang, W.; Yan, F.; and Liu, Y. 2020. BatchCrypt: Efficient homomorphic encryption for Cross-Silo federated learning. In *USENIX ATC*, 493–506.

Zhang, C.; Long, G.; Zhou, T.; Yan, P.; Zhang, Z.; Zhang, C.; and Yang, B. 2023a. Dual personalization on federated recommendation. In *IJCAI*, 4558–4566.

Zhang, C.; Long, G.; Zhou, T.; Zhang, Z.; Yan, P.; and Yang, B. 2024a. GPFedRec: Graph-guided personalization for federated recommendation. In *KDD*, 4131–4142.

Zhang, H.; Li, H.; Chen, J.; Cui, S.; Yan, K.; Wuerkaixi, A.; Zhou, X.; Shen, Z.; and Li, Y. 2024b. Beyond Similarity: Personalized Federated Recommendation with Composite Aggregation. *arXiv preprint arXiv:2406.03933*.

Zhang, H.; Liu, H.; Li, H.; and Li, Y. 2024c. Transfr: Transferable federated recommendation with pre-trained language models. *arXiv preprint arXiv:2402.01124*.

Zhang, H.; Luo, F.; Wu, J.; He, X.; and Li, Y. 2023b. LightFR: Lightweight Federated Recommendation with Privacy-preserving Matrix Factorization. *ACM TOIS*, 41(4): 1–28.

Zhang, J.; Hua, Y.; Wang, H.; Song, T.; Xue, Z.; Ma, R.; and Guan, H. 2023c. Fedala: Adaptive local aggregation for personalized federated learning. In *AAAI*, 11237–11244.

Zhang, M.; Sapra, K.; Fidler, S.; Yeung, S.; and Alvarez, J. M. 2021. Personalized federated learning with first order model optimization. In *ICLR*.

Zhou, Z.; Chen, Z.; Chen, Y.; Zhang, B.; and Yan, J. 2024. On the Emergence of Cross-Task Linearity in Pretraining-Finetuning Paradigm. In *ICML*.

Appendix Overview

We provide the following supplementary materials to further support and elaborate on the main content of our paper:

A. **Detailed Theoretical Analysis**: We present formal assumptions and detailed proofs for our theoretical results, including the Performance Bottleneck of Global Aggregation and the Compensation Effect of Elastic Merging.

B. **Algorithms**: A comprehensive description of the FedEM algorithm is provided, including the full procedural flow and pseudocode.

C. **Detailed Discussions**: We further discuss the complexity and privacy of the proposed method.

D. **Detailed Experimental Settings**: We elaborate on the datasets, evaluation protocols, compared baselines, and implementation details used in our experiments.

E. **Extensive Experimental Results**: We include additional experimental results covering performance comparisons, convergence behavior, parameter visualizations, hyperparameter analysis, and privacy-related evaluations.

A Detailed Theoretical Analysis

To facilitate theoretical analysis, we begin by stating a set of standard assumptions regarding the local objective functions. Specifically, we assume that each client's local loss function $\mathcal{L}_u(\mathbf{Q})$ is continuously differentiable, L-smooth, and μ-strongly convex with respect to the model parameter $\mathbf{Q}$. In particular, the binary cross-entropy (BCE) loss we adopt is smooth and convex (Boyd and Vandenberghe 2004; Shalev-Shwartz and Ben-David 2014), and the addition of ℓ_2 regularization renders the overall objective approximately strongly convex (Bottou, Curtis, and Nocedal 2018). Additionally, such assumptions are also standard in theoretical analyses of federated learning (McMahan et al. 2017; T Dinh, Tran, and Nguyen 2020).

Assumption 2 (L-smoothness)**.** *The local loss function $\mathcal{L}_u$ of client u is assumed to be L-*smooth*, i.e., its gradient is L-Lipschitz continuous. That is, for any $\mathbf{Q}_1, \mathbf{Q}_2$, it holds that:*

$$\|\nabla \mathcal{L}_u(\mathbf{Q}_1) - \nabla \mathcal{L}_u(\mathbf{Q}_2)\| \leq L\|\mathbf{Q}_1 - \mathbf{Q}_2\|. \tag{14}$$

An equivalent characterization of L-smoothness is the following upper bound on the function value:

$$\begin{aligned}\mathcal{L}_u(\mathbf{Q}_2) \leq \mathcal{L}_u(\mathbf{Q}_1) + \langle \nabla \mathcal{L}_u(\mathbf{Q}_1), \mathbf{Q}_2 - \mathbf{Q}_1 \rangle \\ + \frac{L}{2}\|\mathbf{Q}_2 - \mathbf{Q}_1\|^2.\end{aligned} \tag{15}$$

Here, $L > 0$ is the smoothness constant, which quantifies the maximum rate of change in the gradient. Intuitively, L-smoothness implies that the loss surface is locally well-behaved, enabling stable gradient-based optimization.

Assumption 3 (μ-strong convexity)**.** *The local loss function $\mathcal{L}_u$ of client u is assumed to be μ-*strongly convex*, i.e., it satisfies the following inequality for any $\mathbf{Q}_1, \mathbf{Q}_2$:*

$$\begin{aligned}\mathcal{L}_u(\mathbf{Q}_2) \geq \mathcal{L}_u(\mathbf{Q}_1) + \langle \nabla \mathcal{L}_u(\mathbf{Q}_1), \mathbf{Q}_2 - \mathbf{Q}_1 \rangle \\ + \frac{\mu}{2}\|\mathbf{Q}_2 - \mathbf{Q}_1\|^2.\end{aligned} \tag{16}$$

Here, $\mu > 0$ is the strong convexity constant, which quantifies the curvature of the loss surface. Intuitively, μ-strong convexity ensures that $\mathcal{L}_u$ has a unique minimizer, and that gradient-based updates will converge rapidly and stably to this minimum.

Based on the aforementioned assumptions, we first examine the limitations of global aggregation in federated recommendation. We then provide a theoretical analysis of how the proposed elastic merging scheme alleviates this issue.

Lemma 3 (Performance Bottleneck of Global Aggregation)**.** *Let $\mathbf{Q}_u^*$ denote the local optimal model. Following the static replacement scheme, i.e., using the global model $\mathbf{Q}^{g(t)}$ for local training at round $t+1$, may lead to:*

$$\langle \nabla \mathcal{L}_u(\mathbf{Q}^{g(t)}), \mathbf{Q}^{g(t)} - \mathbf{Q}_u^* \rangle \leq 0, \tag{17}$$

which suggests that the global aggregated model can cause the training optimization to deviate from the client-specific optimum, thereby undermining local personalization. This issue becomes more pronounced under the inherent high client heterogeneity in FR, leading to a performance bottleneck.

Proof. Most existing FR methods employ weighted aggregation on the server side to share collaborative information across clients. Taking FCF (Ammad-Ud-Din et al. 2019) for example, that is,

$$\begin{aligned} \mathbf{Q}^{g(t)} &= \frac{1}{n}\sum_{u=1}^{n} \mathbf{Q}_u^{l(t)} = \frac{1}{n}\mathbf{Q}_u^{l(t)} + \frac{n-1}{n}\mathbf{Q}_r^{g(t)}, \\ \text{where} \quad \mathbf{Q}_r^{g(t)} &= \frac{1}{n-1}\sum_{v \neq u}^{n} \mathbf{Q}_v^{l(t)}. \end{aligned} \tag{18}$$

Here, $\mathbf{Q}_r^{g(t)}$ denotes the aggregated model over the remaining clients except ego Client u. However, FR tasks often involve tens of thousands or even more clients, so we have,

$$\lim_{n\to\infty} \frac{1}{n} = 0, \lim_{n\to\infty} \frac{n-1}{n} = 1. \tag{19}$$

Thus, when n is large, $\mathbf{Q}^{g(t)} \approx \mathbf{Q}_r^{g(t)}$, that is, the global model contains almost no information from the own data $\mathcal{D}_u$ of client u. Since $\mathcal{L}_u$ is L-smooth, we have:

$$\|\nabla \mathcal{L}_u(\mathbf{Q}^{g(t)}) - \nabla \mathcal{L}_u(\mathbf{Q}_r^{g(t)})\| \leq L\|\mathbf{Q}^{g(t)} - \mathbf{Q}_r^{g(t)}\| \approx 0. \tag{20}$$

Thus, we have $\nabla \mathcal{L}_u(\mathbf{Q}^{g(t)}) \approx \nabla \mathcal{L}_u(\mathbf{Q}_r^{g(t)})$ when n is large. The gradient direction of the global model is almost entirely determined by the non-local model $\mathbf{Q}_r^{g(t)}$.

Based on the above derivation, in the following analysis, we use $\mathbf{Q}_r^{g(t)}$ to represent $\mathbf{Q}^{g(t)}$. Denote the client-specific optimum by $\mathbf{Q}_u^* = \arg\min_{\mathbf{Q}} \mathcal{L}_u(\mathbf{Q})$. Then, due to the μ-strong convexity of $\mathcal{L}_u$, for any local model $\mathbf{Q}_u^{l(t)}$ trained on the data distribution $\mathcal{D}_u$, it holds that:

$$\begin{aligned} &\left\langle \nabla \mathcal{L}_u(\mathbf{Q}_u^{l(t)}), \mathbf{Q}_u^{l(t)} - \mathbf{Q}_u^* \right\rangle \geq \mathcal{L}_u(\mathbf{Q}_u^{l(t)}) - \mathcal{L}_u(\mathbf{Q}_u^*) \\ &+ \frac{\mu}{2}\|\mathbf{Q}_u^{l(t)} - \mathbf{Q}_u^*\|^2 = \mathcal{L}_u(\mathbf{Q}_u^{l(t)}) + \frac{\mu}{2}\|\mathbf{Q}_u^{l(t)} - \mathbf{Q}_u^*\|^2 \\ &\geq \frac{\mu}{2}\|\mathbf{Q}_u^{l(t)} - \mathbf{Q}_u^*\|^2 \geq 0, \end{aligned} \tag{21}$$

which implies that the optimization direction of the trained model $\mathbf{Q}_u^{l(t)}$ is toward its local optimum $\mathbf{Q}_u^*$.

Under non-IID settings in FR, the optimal solutions $\mathbf{Q}_u^*$ of local loss functions $\mathcal{L}_u$ differ significantly across clients. That is, for $v \neq u$, we have $\|\mathbf{Q}_v^* - \mathbf{Q}_u^*\| \gg 0$. In this case, the aggregated reference model $\mathbf{Q}_r^{g(t)} = \frac{1}{n-1}\sum_{v\neq u}^{n} \mathbf{Q}_v^{l(t)}$, naturally deviates from the local optimum $\mathbf{Q}_u^*$, leading to $\|\mathbf{Q}_r^{g(t)} - \mathbf{Q}_u^*\| > 0$. Following prior works on federated learning under heterogeneous settings (Karimireddy et al. 2020; Li et al. 2020), such deviation is lower bounded by a constant that depends on the level of data heterogeneity. Specifically, it holds that:

$$\|\mathbf{Q}_r^{g(t)} - \mathbf{Q}_u^*\| \geq \Omega(\zeta), \tag{22}$$

where ζ is a heterogeneity coefficient. A larger ζ indicates a greater divergence between the global model and the local optimum. Consequently, when ζ is sufficiently large, the model $\mathbf{Q}_r^{g(t)}$ may lie on the wrong side of the loss landscape with respect to $\mathcal{L}_u$. Specifically, the condition implied in Eq. (21) may no longer hold, that is,

$$\langle \nabla \mathcal{L}_u(\mathbf{Q}_r^{g(t)}), \mathbf{Q}_r^{g(t)} - \mathbf{Q}_u^* \rangle \leq 0. \tag{23}$$

Under the traditional static replacement scheme, the next local gradient descent update is given by:

$$\mathbf{Q}_u^{l(t+1)} = \mathbf{Q}_r^{g(t)} - \eta \nabla \mathcal{L}_u(\mathbf{Q}_r^{g(t)}), \tag{24}$$

where $\mathbf{Q}_r^{g(t)}$ is the received global model. We analyze the squared distance between the updated model and the local optimum $\mathbf{Q}_u^*$:

$$\begin{aligned} &\|\mathbf{Q}_u^{l(t+1)} - \mathbf{Q}_u^*\|^2 = \|\mathbf{Q}_r^{g(t)} - \mathbf{Q}_u^*\|^2 \\ &- 2\eta \langle \nabla \mathcal{L}_u(\mathbf{Q}_r^{g(t)}), \mathbf{Q}_r^{g(t)} - \mathbf{Q}_u^* \rangle + \eta^2 \|\nabla \mathcal{L}_u(\mathbf{Q}_r^{g(t)})\|^2. \end{aligned} \tag{25}$$

Rearranging terms, we can obtain:

$$\begin{aligned} &\|\mathbf{Q}_u^{l(t+1)} - \mathbf{Q}_u^*\|^2 - \|\mathbf{Q}_r^{g(t)} - \mathbf{Q}_u^*\|^2 = \\ &- 2\eta \langle \nabla \mathcal{L}_u(\mathbf{Q}_r^{g(t)}), \mathbf{Q}_r^{g(t)} - \mathbf{Q}_u^* \rangle + \eta^2 \|\nabla \mathcal{L}_u(\mathbf{Q}_r^{g(t)})\|^2 \geq 0. \end{aligned} \tag{26}$$

Therefore, we have:

$$\|\mathbf{Q}_u^{l(t+1)} - \mathbf{Q}_u^*\|^2 \geq \|\mathbf{Q}_r^{g(t)} - \mathbf{Q}_u^*\|^2 \geq \Omega(\zeta^2), \tag{27}$$

which indicates that the local model gradually moves away from the local optimum. These analyses demonstrate that the local model may be optimized in a direction opposite to the local objective, which can hinder convergence and degrade personalization. This issue becomes more pronounced as the heterogeneity level ζ increases, thereby resulting in a performance bottleneck during global aggregation.
Hence, the proof is complete. □

Lemma 4 (Compensation Effect of Elastic Merging)**.** *Following the elastic merging scheme, we define the merged model as:*

$$\mathbf{Q}_u^{m(t+1)} = \rho \mathbf{Q}^{g(t)} + (1-\rho)\mathbf{Q}_u^{l(t)}, \quad \rho \in [0,1]. \tag{28}$$

The merged model can compensate for the optimization deviation caused by aggregation, guiding the update toward the client-specific optimum, i.e., it satisfies:

$$\begin{aligned}&\langle\nabla\mathcal{L}_u(\mathbf{Q}_u^{m(t+1)}),\mathbf{Q}_u^{m(t+1)}-\mathbf{Q}_u^*\rangle\\&\geq(1-\rho)\langle\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)}),\mathbf{Q}_u^{l(t)}-\mathbf{Q}_u^*\rangle-\rho C,\end{aligned}\tag{29}$$

where C denotes the deviation introduced by global aggregation. This deviation can be lower bounded by both the state of the local model $\mathbf{Q}_u^{l(t)}$ and the global-local discrepancy $\mathbf{Q}_u^{\Delta(t)}=\mathbf{Q}^{g(t)}-\mathbf{Q}_u^{l(t)}$. Specifically, the bound is given by $C=\|\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)})\|(\|\mathbf{Q}_u^{l(t)}-\mathbf{Q}_u^*\|+\|\mathbf{Q}_u^{\Delta(t)}\|)+L\|\mathbf{Q}_u^{\Delta(t)}\|(\|\mathbf{Q}_u^{l(t)}-\mathbf{Q}_u^*\|+\rho\|\mathbf{Q}_u^{\Delta(t)}\|)$. The first term on the right-hand side of the inequality, $A=\langle\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)}),\mathbf{Q}_u^{l(t)}-\mathbf{Q}_u^*\rangle>0$, reflects the preservation of client-specific optimization information. Thus, we can adaptively control the merging weight $0\leq\rho<\frac{A}{A+C}$ to facilitate local personalized optimization during global collaborative training.

Proof. Based on the definition $\mathbf{Q}_u^{m(t+1)}=\rho\mathbf{Q}^{g(t)}+(1-\rho)\mathbf{Q}_u^{l(t)}$, we define the global-local discrepancy as $\mathbf{Q}_u^{\Delta(t)}=\mathbf{Q}^{g(t)}-\mathbf{Q}_u^{l(t)}$, such that the merged model can be rewritten as $\mathbf{Q}_u^{m(t+1)}=\mathbf{Q}_u^{l(t)}+\rho\mathbf{Q}_u^{\Delta(t)}$. By L-smoothness, we have:

$$\nabla\mathcal{L}_u(\mathbf{Q}_u^{m(t+1)})=\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)})+\mathbf{R},\tag{30}$$

where $\|\mathbf{R}\|\leq L\|\mathbf{Q}_u^{m(t+1)}-\mathbf{Q}_u^{l(t)}\|=L\rho\|\mathbf{Q}_u^{\Delta(t)}\|$. We now analyze the optimization direction of the merged model:

$$\begin{aligned}&\langle\nabla\mathcal{L}_u(\mathbf{Q}_u^{m(t+1)}),\mathbf{Q}_u^{m(t+1)}-\mathbf{Q}_u^*\rangle\\&=\langle\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)})+\mathbf{R},\,\mathbf{Q}_u^{m(t+1)}-\mathbf{Q}_u^*\rangle\\&=\langle\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)}),\,\mathbf{Q}_u^{m(t+1)}-\mathbf{Q}_u^*\rangle+\langle\mathbf{R},\,\mathbf{Q}_u^{m(t+1)}-\mathbf{Q}_u^*\rangle\\&=\langle\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)}),\,(1-\rho)(\mathbf{Q}_u^{l(t)}-\mathbf{Q}_u^*)+\rho(\mathbf{Q}^{g(t)}-\mathbf{Q}_u^*)\rangle\\&\quad+\langle\mathbf{R},\,\mathbf{Q}_u^{m(t+1)}-\mathbf{Q}_u^*\rangle\\&=(1-\rho)\langle\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)}),\,\mathbf{Q}_u^{l(t)}-\mathbf{Q}_u^*\rangle+\\&\quad(\rho\langle\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)}),\,\mathbf{Q}^{g(t)}-\mathbf{Q}_u^*\rangle+\langle\mathbf{R},\,\mathbf{Q}_u^{m(t+1)}-\mathbf{Q}_u^*\rangle).\end{aligned}\tag{31}$$

Moreover, the lower bound can be formulated as:

$$\begin{aligned}&\langle\nabla\mathcal{L}_u(\mathbf{Q}_u^{m(t+1)}),\mathbf{Q}_u^{m(t+1)}-\mathbf{Q}_u^*\rangle\\&\geq(1-\rho)\langle\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)}),\,\mathbf{Q}_u^{l(t)}-\mathbf{Q}_u^*\rangle-\\&\quad(\rho\|\langle\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)}),\,\mathbf{Q}^{g(t)}-\mathbf{Q}_u^*\rangle\|+\|\langle\mathbf{R},\,\mathbf{Q}_u^{m(t+1)}-\mathbf{Q}_u^*\rangle\|).\end{aligned}\tag{32}$$

By applying the Cauchy–Schwarz inequality and the triangle inequality, we obtain the following upper bound:

$$\begin{aligned}&\|\langle\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)}),\,\mathbf{Q}^{g(t)}-\mathbf{Q}_u^*\rangle\|\\&\leq\|\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)})\|\cdot\|\mathbf{Q}^{g(t)}-\mathbf{Q}_u^*\|\\&=\|\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)})\|\cdot\|\mathbf{Q}_u^{l(t)}-\mathbf{Q}_u^*+\mathbf{Q}_u^{\Delta(t)}\|\\&\leq\|\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)})\|\cdot(\|\mathbf{Q}_u^{l(t)}-\mathbf{Q}_u^*\|+\|\mathbf{Q}_u^{\Delta(t)}\|)\end{aligned}\tag{33}$$

Similarly, we have:

$$\begin{aligned}&\|\langle\mathbf{R},\,\mathbf{Q}_u^{m(t+1)}-\mathbf{Q}_u^*\rangle\|\leq\|\mathbf{R}\|\cdot\|\mathbf{Q}_u^{m(t+1)}-\mathbf{Q}_u^*\|\\&=\|\mathbf{R}\|\cdot\|\mathbf{Q}_u^{l(t)}-\mathbf{Q}_u^*+\rho\mathbf{Q}_u^{\Delta(t)}\|\\&\leq L\rho\|\mathbf{Q}_u^{\Delta(t)}\|\cdot(\|\mathbf{Q}_u^{l(t)}-\mathbf{Q}_u^*\|+\rho\|\mathbf{Q}_u^{\Delta(t)}\|)\end{aligned}\tag{34}$$

Thus, Eq. (32) can be reformulated as:

$$\begin{aligned}&\langle\nabla\mathcal{L}_u(\mathbf{Q}_u^{m(t+1)}),\mathbf{Q}_u^{m(t+1)}-\mathbf{Q}_u^*\rangle\\&\geq(1-\rho)\langle\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)}),\,\mathbf{Q}_u^{l(t)}-\mathbf{Q}_u^*\rangle\\&\quad-\rho\|\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)})\|\cdot(\|\mathbf{Q}_u^{l(t)}-\mathbf{Q}_u^*\|+\|\mathbf{Q}_u^{\Delta(t)}\|)\\&\quad-L\rho\|\mathbf{Q}_u^{\Delta(t)}\|\cdot(\|\mathbf{Q}_u^{l(t)}-\mathbf{Q}_u^*\|+\rho\|\mathbf{Q}_u^{\Delta(t)}\|).\end{aligned}\tag{35}$$

For simplicity, we denote $A=\langle\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)}),\mathbf{Q}_u^{l(t)}-\mathbf{Q}_u^*\rangle$ as the optimization information of the local model, *i.e.*, its original update toward the client-specific optimum. Besides, $C=\|\nabla\mathcal{L}_u(\mathbf{Q}_u^{l(t)})\|(\|\mathbf{Q}_u^{l(t)}-\mathbf{Q}_u^*\|+\|\mathbf{Q}_u^{\Delta(t)}\|)+L\|\mathbf{Q}_u^{\Delta(t)}\|(\|\mathbf{Q}_u^{l(t)}-\mathbf{Q}_u^*\|+\rho\|\mathbf{Q}_u^{\Delta(t)}\|)$ denotes the optimization deviation introduced by global aggregation. To ensure that the merged model avoids the performance bottleneck analyzed in Lemma 3, that is, its optimization direction should be aligned with the local objective, we require the left-hand side of Eq. (35) to be non-negative. Thus, we need to adaptively control the merging weight $0\leq\rho<\frac{A}{A+C}$ to compensate for the deficiency of the aggregation bottleneck.

Building on the above analysis, it is necessary to take into account the deviation term C, *i.e.*, the state of the local model $\mathbf{Q}_u^{l(t)}$ and the global-local discrepancy $\mathbf{Q}_u^{\Delta(t)}$, to determine a tailored merging weight ρ. Intuitively, we need to measure the local model's need for the collaborative information encoded in $\mathbf{Q}_u^{\Delta(t)}$, so as to guide the merged model toward the local optimum and preserve personalization. Hence, the proof is complete. □

B Algorithms

We present the FedEM algorithm in detail in Algorithm 1. At the beginning of each round, client u merges the downloaded global model $\mathbf{Q}_u^{g(t-1)}$ with the off-the-shelf local model $\mathbf{Q}_u^{l(t-1)}$ to derive a merged model $\mathbf{Q}_u^{m(t)}$, while updating the adapter Θ_u. It adopts an elastic merging scheme instead of the static replacement scheme used in existing FR methods. Then, client u updates $\mathbf{Q}_u^{m(t)}$ and other local parameters, *e.g.*, $\mathbf{p}_u^{(t-1)}$ using its own data $\mathcal{D}_u$, and uploads the updated model $\mathbf{Q}_u^{l(t)}$ to the server. The server aggregates $\{\mathbf{Q}_1^{l(t)},\cdots,\mathbf{Q}_n^{l(t)}\}$ based on similarity to generate a personalized global model $\mathbf{Q}_u^{g(t)}$ for each client. Initially, $\mathbf{Q}_u^{g(0)}$ and $\mathbf{Q}_u^{l(0)}$ are randomly initialized. After T rounds, each client obtains a personalized model.

C Detailed Discussions

Complexity Analysis

Given the limited local resources in federated recommendation scenarios, we primarily focus on analyzing the local complexity. Suppose there are n clients and m items, with

Algorithm 1: FedEM

Input: selected clients $\mathcal{U}_s$; global rounds T; local epochs E; batch size B; learning rates η, β; tuning coefficients α

Output: personalized model $\mathbf{p}_u^{(T)}$,$\mathbf{Q}_u^{l(T)}$, and adapter Θ_u for each client u

GlobalProcedure:

for each round $t = 1, 2, \cdots, T$ **do**
  for each client $u \in \mathcal{U}_s$ **in parallel do**
    Downloads global model $\mathbf{Q}_u^{g(t-1)}$ from the server;
    $\mathbf{Q}_u^{l(t)} \leftarrow$ ClientUpdate($\mathbf{Q}_u^{g(t-1)}$, u);
    Uploads local model $\mathbf{Q}_u^{l(t)}$ to the server;
  end for
  Generates $\mathbf{Q}_u^{g(t)} = \sum_{v=1}^{n} w_{uv} \mathbf{Q}_v^{l(t)}$ based on Eq.(13); ▷ Global Aggregation
end for

ClientUpdate($\mathbf{Q}_u^{g(t-1)}$, u):

for each local epoch $r = 1, 2, \cdots, E$ **do**
  for each batch $b = 1, 2, \cdots, B$ in $\mathcal{D}_u$ **do**
    Updates Θ_u with Eq.(11);
    Yields $\boldsymbol{\rho}$ based on Θ_u via Eq.(9);
    Obtains $\mathbf{Q}_u^{m(t)} = (\mathbf{1} - \boldsymbol{\rho})\mathbf{Q}_u^{l(t-1)} + \boldsymbol{\rho} \odot \mathbf{Q}_u^{g(t-1)}$;
  end for ▷ Elastic Merging
  for each batch $b = 1, 2, \cdots, B$ in $\mathcal{D}_u$ **do**
    Updates $\mathbf{p}_u^{(t-1)}$ and $\mathbf{Q}_u^{m(t)}$ with Eq.(12);
  end for ▷ Local Training
end for
Obtains updated Θ_u, $\mathbf{p}_u^{(t)}$ and $\mathbf{Q}_u^{l(t)}$ after E epochs;
return $\mathbf{Q}_u^{l(t)}$

an embedding dimension of d. The computational complexity of the local backbone model is $\mathcal{O}((m+1)d)$. Our proposed EM module is implemented as an L-layer MLP with a complexity of $\mathcal{O}(Ld^2)$. Since in practice $m \gg d > L$, the EM module incurs negligible overhead on local devices, making it well-suited for on-device recommendation.

Considering other SOTA methods, PFedRec introduces an L-layer personalized score function with a similar complexity of $\mathcal{O}(Ld^2)$, which is lightweight but delivers inferior performance compared to our method. Other high-performing approaches, such as FedRAP, require training and storing an additional model locally, resulting in a complexity overhead of $\mathcal{O}(md)$. FedCIA incurs an even higher cost of $\mathcal{O}(m^2 d)$ due to the computation of a local item similarity matrix. These methods are therefore unsuitable for resource-constrained on-device scenarios. In contrast, our method strikes a favorable balance between performance and computational complexity.

Privacy Analysis

Inherent Federated Framework. FedEM strictly follows the setting of FR that no raw data is shared with the third parties, safeguarding user privacy and data security (Sun et al. 2024; Yin et al. 2024).

Proposed Elastic Merging. In the traditional federated framework, each client updates the downloaded global model $\mathbf{Q}_u^{g(t-1)}$ to obtain its local model $\mathbf{Q}_u^{l(t)}$, which is then uploaded to the server. This direct update makes it feasible for the server to infer client-specific data distributions via model differences. Formally, $\mathcal{D}_u \leftarrow \text{Attack}(\mathbf{Q}_u^{l(t)} - \mathbf{Q}_u^{g(t-1)})$, where $\text{Attack}(\cdot)$ denotes any reconstruction or inference attack algorithm, such as Gradients Leak (Chai et al. 2020). By contrast, FedEM introduces local merging before training. Each client first interpolates a personalized initialization $\mathbf{Q}_u^{m(t)} = \boldsymbol{\rho}\mathbf{Q}_u^{g(t-1)} + (\mathbf{1} - \boldsymbol{\rho})\mathbf{Q}_u^{l(t-1)}$, and then performs local training on $\mathbf{Q}_u^{m(t)}$ to obtain $\mathbf{Q}_u^{l(t)}$. As a result, the attack surface shifts to $\mathcal{D}_u \leftarrow \text{Attack}(\mathbf{Q}_u^{l(t)} - \mathbf{Q}_u^{m(t)})$, where the server must know the internal merging weights $\boldsymbol{\rho}$ to reconstruct $\mathbf{Q}_u^{m(t)}$. Since $\boldsymbol{\rho}$ is generated locally by the client and not exposed to the server, FedEM effectively obfuscates the model update trajectory, increasing the difficulty of inference attacks and enhancing privacy protection.

Privacy-Enhanced FedEM. Our proposed elastic merging is model agnostic, which can be smoothly integrated with other privacy-enhanced FR models, *e.g.*, FMF-LDP (Minto et al. 2021). Besides, advanced privacy protection strategies, such as differential privacy (Qi, Wang, and Huang 2024) and homomorphic encryption (Zhang et al. 2020), are also applicable to our method, further enhancing user privacy. Following previous works (Dwork et al. 2006; Choi et al. 2018), we introduce the differential privacy strategy into our method by incorporating some zero-mean Laplacian noises to the uploaded models, that is,

$$\mathbf{Q}_u^{l*} = \mathbf{Q}_u^{l} + \text{Laplacian}(0, \delta), \tag{36}$$

where $\delta = \mathcal{S}_u/\varepsilon$ is the noise strength (Dwork et al. 2006). $\mathcal{S}_u$ represents the global sensitivity of client u, whose upper bound is derived in Lemma 5. A higher value of δ corresponds to stronger privacy guarantees.

Lemma 5 (Upper Bound of Global Sensitivity $\mathcal{S}_u$)**.** *Consider two global models, $\mathbf{Q}^g$ and $\mathbf{Q}^{g\prime}$, trained on datasets differing only by the data of client u, i.e., $\mathcal{D}_u$ and $\mathcal{D}'_u$. Then the following inequality holds:*

$$\begin{aligned} \mathcal{S}_u = \|\mathbf{Q}^g - \mathbf{Q}^{g\prime}\| &= \|p_u \eta (\nabla \mathbf{Q}_u^l - \nabla \mathbf{Q}_u^{l\prime})\| \\ &\leq p_u \eta (\|\nabla \mathbf{Q}_u^l\| + \|\nabla \mathbf{Q}_u^{l\prime}\|) \leq 2 p_u \eta Z, \end{aligned} \tag{37}$$

where p_u is the aggregation weight of client u's local model, and η is the local learning rate for updating embeddings. The gradients $\nabla \mathbf{Q}_u^l$ and $\nabla \mathbf{Q}_u^{l\prime}$ are computed on datasets $\mathcal{D}_u$ and $\mathcal{D}'_u$, respectively. They can be bounded via gradient clipping by a predefined constant Z (Wei et al. 2020).

D Detailed Experimental Settings

Datasets

We experiment on four benchmark datasets with varying client scales and data sparsity to comprehensively evaluate our proposed method. **Filmtrust** (Guo, Zhang, and Yorke-Smith 2013), **Movielens-100K (ML-100K)** (Harper and Konstan 2015) and **Movielens-1M (ML-1M)** (Harper and Konstan 2015) are for movie recommendation with user-movie ratings, **LastFM-2K** (Cantador, Brusilovsky, and Kuflik 2011) is for music recommendation with user-artist listening counts. Here, we convert ratings/counts greater than 0 to 1 to produce implicit data. In this work, we filter out all the users with less than 10 interactions from the above

four datasets. Besides, each user is treated as an independent client, and each client's data inherently exhibits great heterogeneity. The characteristics of the four datasets are summarized in Table 5.

Datasets	#User/Client	#Item	#Interaction	Sparsity
Filmtrust	1,002	2,042	33,372	98.37%
ML-100K	943	1,682	100,000	93.70%
ML-1M	6,040	3,706	1,000,209	95.53%
LastFm-2K	1,600	12,454	185,650	99.07%

Table 5: Statistics of the experimental datasets.

Evaluation Protocols

We follow the popular *leave-one-out* evaluation (Bayer et al. 2017; He et al. 2017), that is, hold out the latest interaction as the test set and regard the remaining data as train set. In addition, we take the last interaction of train set as the validation set for tuning hyper-parameters. To alleviate the high computational cost of ranking all items for each user, we follow the common practice (He et al. 2017; Zhang et al. 2023a). Specifically, for each user, we sample 99 negatives not seen during training and rank the test instance among the 100 items. Besides, the model performance is reported by *Hit Ratio* (HR@K) and *Normalized Discounted Cumulative Gain* (NDCG@K) metrics (He et al. 2015), indicating whether the test item is recommended and its rank in the top K recommended items, respectively. In this work, we set $K = 10$, and report the results as the average of 5 repeated experiments.

Compared Baselines

We comprehensively compare FedEM with SOTA centralized methods as follows:

Matrix Factorization (MF) (Koren, Bell, and Volinsky 2009): A foundational recommendation approach that decomposes the user-item interaction matrix into latent user and item embeddings, effectively capturing their respective features within a shared vector space.

Neural Collaborative Filtering (NCF) (He et al. 2017): Extending traditional embedding-based methods, NCF employs a multi-layer perceptron (MLP) to model complex, non-linear interactions between users and items based on their latent representations.

LightGCN (He et al. 2020): A graph-based collaborative filtering method that simplifies traditional GCNs by removing non-linear transformations and feature combinations, enabling efficient learning of embeddings through linear propagation over the user-item interaction graph.

Besides, we compare FedEM with some SOTA federated methods, such as:

FedMF (Chai et al. 2020): A federated version of MF, where user embeddings are updated locally, and encrypted item gradients are shared with the server for aggregation. In our experiments, we use its unencrypted variant to focus on performance evaluation.

FedNCF (Perifanis and Efraimidis 2022): It extends NCF to the federated setting by updating user embeddings locally, while both item embeddings and the MLP components are uploaded for global aggregation on the server.

FedFast (Muhammad et al. 2020): It groups users into clusters according to their profile similarity, and applies cluster-specific sampling and aggregation strategies to enhance training efficiency.

PFedRec (Zhang et al. 2023a): It utilizes a dual personalization mechanism that captures user preferences with a personalized score function and fine-grained personalization on item embeddings.

CoLR (Nguyen et al. 2024): It decomposes local updates into a low-rank matrix, where only a subset of parameters is trained and transmitted, while the remaining part is fixed and shared. This approach significantly reduces communication overhead while preserving model performance.

GPFedRec (Zhang et al. 2024a): It proposes a graph-based aggregation strategy that captures inter-client relationships to guide global learning of user-specific item embeddings, thereby enhancing local personalization.

FedRAP (Li, Long, and Zhou 2024): It considers both the global view and user-specific view of items by applying an additive model to item embedding. This method requires updating the local model and the additive one for each client.

FedCIA (Han et al. 2025): It proposes a model-free aggregation method that uploads and aggregates client-side item similarity matrices, achieving better information retention and stronger personalization for each client.

Implementation Details

In the experiments, we randomly sample $N = 4$ negative instances for each positive sample, following prior works (He et al. 2017; Zhang et al. 2023a). To ensure fair comparison, we set the embedding dimension $d = 16$ and batch size $B = 256$ for all methods, while other hyperparameter settings for baselines follow their default/optimal implementations. For centralized methods, the total number of training epochs is set to 100 to ensure convergence. For federated methods, we set the number of global rounds $R = 100$ and sample 100% of clients in each round to perform global aggregation for generality. Notably, for FedFast, we group clients into 5 clusters for stable training. For CoLR, the rank of decomposed updates is set to 4.

For our method, we adopt the same basic configurations as the backbone FedMF, *i.e.*, a learning rate $\eta = 0.1$ for user and item embeddings, and $E = 10$ local training epochs with the Adam optimizer. As for the additional hyperparameter α, we perform a grid search within the range $[0.5, 1.5]$ with a step size of 0.1. By default, the adapter is implemented as a 3-hidden-layer MLP with the structure $[32, 16, 8, 1]$. We also explore MLPs with different depths, including $[32, 1]$, $[32, 16, 1]$, $[32, 32, 16, 8, 1]$, and $[32, 64, 32, 16, 8, 1]$. The learning rate β for the adapter is set to be consistent with the embedding learning rate η, *i.e.*, $\beta = 0.1$. All experiments are conducted on a server equipped with four NVIDIA RTX A5000 GPUs, each with 24GB of memory.

Datasets	Methods	CenRec			FedRec								Ours
		MF	NCF	LightGCN	FedMF	FedNCF	FedFast	PFedRec	CoLR	GPFedRec	FedRAP	FedCIA	FedEM
Filmtrust	HR@5	0.6287	0.6537	0.7385	0.6267	0.6347	0.6248	0.6367	0.6018	0.6447	0.6766	0.7774	**0.8353**
	NDCG@5	0.5128	0.5374	0.6338	0.5188	0.5257	0.4826	0.5271	0.4991	0.5298	0.5046	0.6744	**0.7511**
ML-100K	HR@5	0.4719	0.4062	0.7063	0.3277	0.2694	0.3181	0.4772	0.3383	0.5133	0.8367	0.8717	**0.9745**
	NDCG@5	0.3177	0.2755	0.5395	0.2201	0.1794	0.2213	0.3230	0.2266	0.3461	0.7779	0.7479	**0.9359**
ML-1M	HR@5	0.4209	0.4121	0.6831	0.3353	0.2704	0.2921	0.4866	0.3060	0.4957	0.8086	0.8411	**0.9035**
	NDCG@5	0.2779	0.2750	0.5053	0.2243	0.1795	0.1940	0.3296	0.2002	0.3382	0.7489	0.7695	**0.8238**
LastFM-2K	HR@5	0.7628	0.6879	0.7833	0.4846	0.3987	0.3932	0.7226	0.4240	**0.7415**	0.5950	0.6690	0.7360
	NDCG@5	0.5896	0.5687	0.6687	0.3613	0.2985	0.2824	0.6628	0.3230	0.6508	0.5842	0.6468	**0.6967**

Table 6: Performance comparison on four datasets, reported by HR@5 and NDCG@5. CenRec and FedRec represent centralized and federated recommendation methods, respectively. The best FedRec results are bold and the second ones are underlined.

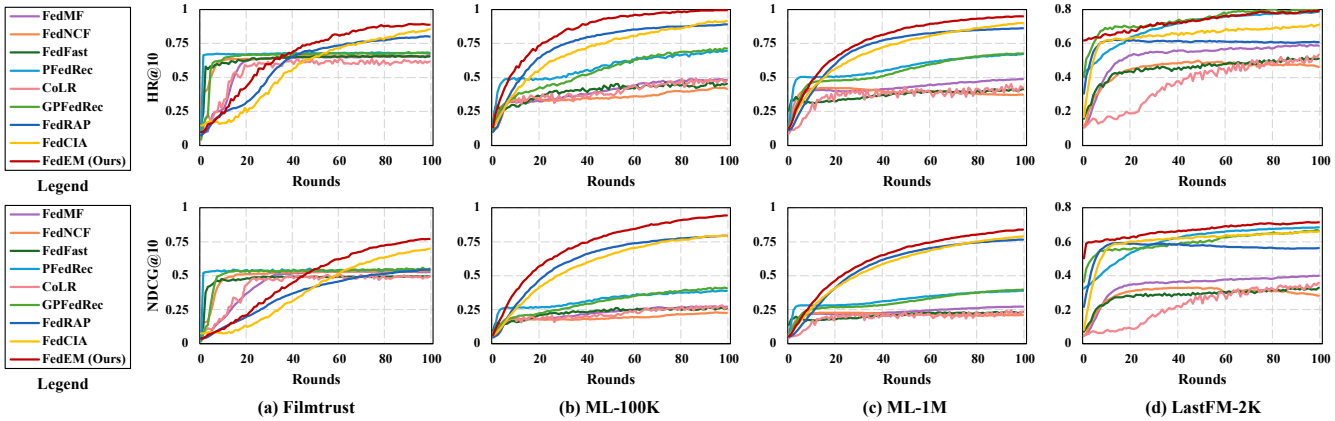

Figure 7: Convergence comparison evaluated by HR@10 and NDCG@10. The horizontal axis is the federated rounds.

E Extensive Experiment Results

More Baseline Comparison

Performance. Furthermore, we present additional performance comparisons on HR@5 and NDCG@5, as illustrated in Figure 6. Our method outperforms both SOTA centralized and federated methods, which is consistent with the evaluation results on HR@10 and NDCG@10. In particular, on relatively dense datasets such as ML-100K and ML-1M, our method significantly outperforms the strongest baselines, with HR@5 exceeding 0.9. For extremely sparse datasets like LastFM-2K, existing federated methods generally underperform compared to centralized approaches such as MF and LightGCN, due to insufficient local interaction data on clients that leads to underfitting. Nevertheless, our method still slightly surpasses centralized models on the NDCG@5 and NDCG@10 metrics, demonstrating its effectiveness even under severe data sparsity. The superior performance of our method stems from identifying and addressing a fundamental issue in FR, rather than relying on heuristic designs. Moreover, the adopted model merging strategy is both theoretically grounded and empirically validated.

Convergence. We further compare the convergence of different methods, with results on four datasets shown in Figure 7. Compared to other methods: (1) FedEM demonstrates faster convergence in the early stages, especially on ML-100K, ML-1M, and LastFM-2K, nearly outperforming all baselines. (2) FedMF achieves more stable and superior performance in the later stages. Our method exhibits even smoother final convergence with minimal fluctuations, particularly noticeable on ML-100K and LastFM-2K. Moreover, its performance surpasses that of all other methods across all datasets. These advantages stem from the elastic merging scheme, which preserves local information from the previous round, effectively preventing the loss of both optimization and personalization information, thereby enhancing convergence stability and overall performance.

More Ablation Study

Visualization of the vectors $\mathbf{w}$. On the server side, we generate a weight vector $\mathbf{w}_u \in \mathbb{R}^n$ for each client u to guide tailored aggregation. By stacking all the weight vectors, we obtain the aggregation matrix $\mathbf{W} = \{\mathbf{w}_u \mid u \in \mathcal{U}\} \in \mathbb{R}^{n \times n}$. Here, we randomly select 10 clients in the same round on ML-100K and visualize the aggregation matrix $\mathbf{W}$. As shown in Figure 8, different clients are assigned customized aggregation weights. The traditional fixed-weight aggregation, such as FedMF, assigns weights based on the relative amount of data, ensuring that clients with more training data have a larger influence in the aggregation. While this can incorporate more collaborative information, it may degrade the performance of the aggregated model when those dominant clients exhibit preferences dissimilar to others, as analyzed in Lemma 3. To this end, we incorporate similarity-

based weights to aggregate clients with similar preferences, thereby mitigating the negative impact of client heterogeneity and promoting more effective collaboration. This design further alleviates the performance bottleneck in global aggregation.

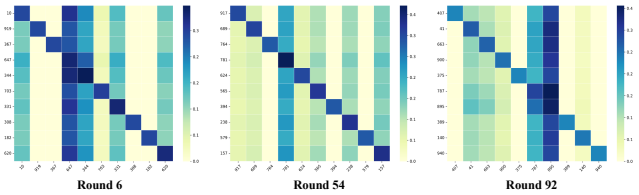

Figure 8: Visualization of matrices $\mathbf{W}$ for global aggregation of several randomly selected clients in different rounds.

Visualization of the vectors $\boldsymbol{\rho}$. We randomly select several clients on ML-100K in the same round and visualize their weight vectors $\boldsymbol{\rho}$ for elastic merging. For brevity, only the weights associated with the first 30 items are displayed, as shown in Figure 9. From the visualization perspective, FedEM takes into account the heterogeneity among clients, elastically yielding tailored $\boldsymbol{\rho}$ to coordinate the global and local models for each client. Additionally, it finely assigns different weights to each item within the same client, optimally exploiting both collaborative and personalized information. The proposed elastic merging scheme leverages the off-the-shelf local model to effectively compensate for the loss of local information introduced by global aggregation, thus breaking the performance bottleneck in FR.

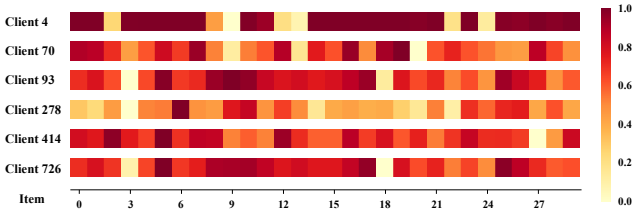

Figure 9: Visualization of vectors $\boldsymbol{\rho}$ for elastic merging of several randomly selected clients in the same round.

More Hyperparameter Analysis

Similarity Coefficient. During global aggregation, we extend the fixed-weight strategy by additionally incorporating model similarity to generate client-specific aggregation weights, which are modulated by a similarity coefficient α. As shown in Figure 10, we find that setting α around 1, *i.e.*, balancing relative data size and model similarity, can effectively improve overall model performance. This is because aggregating models from similar clients mitigates the inherent heterogeneity, thereby reducing the potential negative effects of aggregation. Specifically, the best results are obtained when $\alpha = 0.9$ on FilmTrust, ML-1M, and LastFM-2K, and when $\alpha = 1.1$ on ML-100K.

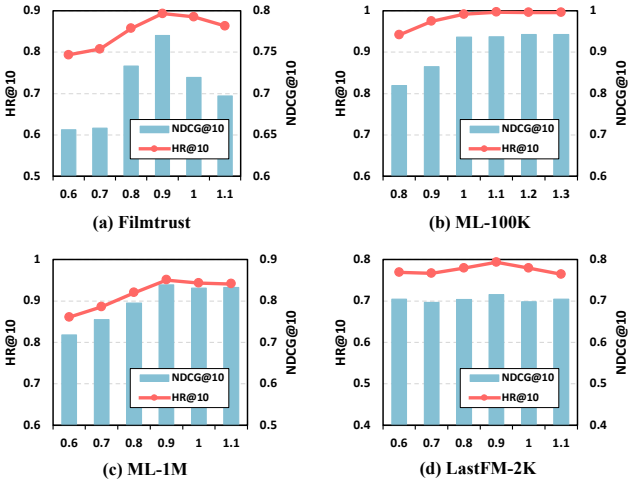

Figure 10: Performance under different similarity coefficients α for global aggregation.

More Privacy Analysis

Our method can be integrated with local differential privacy (LDP) to further enhance privacy protection. We evaluate the model performance under different noise strengths δ, as shown in Table 7. As expected, the performance degrades as δ increases, indicating stronger privacy guarantees. Notably, FedEM remains robust when $\delta \in [0.1, 0.3]$, and even under stronger privacy constraints, *e.g.*, $\delta = 0.5$, it still demonstrates competitive performance. In practice, to balance privacy and utility, we set the noise strength to $\delta = 0.3$.

δ	Filmtrust		ML-100K		ML-1M		LastFM-2K	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
0	0.8932	0.7701	0.9958	0.9427	0.9507	0.8392	0.7935	0.7151
0.1	0.8892	0.7617	0.9936	0.9405	0.9490	0.8337	0.7817	0.7047
0.2	0.8832	0.7459	0.9947	0.9319	0.9434	0.8249	0.7770	0.6999
0.3	0.8802	0.7353	0.9936	0.9399	0.9392	0.8193	0.7707	0.6930
0.4	0.8782	0.7194	0.9947	0.9342	0.9328	0.8122	0.7644	0.6857
0.5	0.8643	0.7025	0.9905	0.9143	0.9308	0.8075	0.7533	0.6778

Table 7: Performance of applying LDP to our method with different noise strength δ.