Discrete Federated Multi-behavior Recommendation for Privacy-Preserving Heterogeneous One-Class Collaborative Filtering

Enyue Yang[1] Weike Pan[1]* Qiang Yang[2] Zhong Ming[1,3,4]

[1]College of Computer Science and Software Engineering,
Shenzhen University, China

[2]Department of Computer Science and Engineering,
Hong Kong University of Science and Technology, China

[3]Shenzhen Technology University, China

[4]Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), China

Problem Definition

Privacy-Preserving Heterogeneous One-Class Collaborative Filtering (PHOCCF)

- Input:
  - The server is allowed to own the complete (user, item) purchase matrix $\mathbf{P} \in \{0, 1\}^{n \times m}$.
  - Each client (i.e., user) has a set of purchased items $\mathcal{I}_u^{\mathcal{P}}$ and a set of examined items $\mathcal{I}_u^{\mathcal{E}}$.
- Goal: Predict the purchase preference value of each user u to each not yet purchased item $j \in \mathcal{I} \backslash \mathcal{I}_u^{\mathcal{P}}$ and recommend the top ranked items in a privacy-preserving, storage-efficient and computation-efficient manner.

Assumption

- We assume that the purchase behaviors can be collected by the server.
- It is often impractical to assume that the organizations do not collect any raw data and they can still meet the basic business imperatives.
- For example, in an online e-commerce platform, if the platform does not know what items a specific user has bought, it could not deliver the items to the user.

Overall of Our Solution

- We propose a novel framework named discrete federated multi-behavior recommendation (DFMR) for PHOCCF. To the best of our knowledge, DFMR is the first federated learning framework for PHOCCF.
- We use discrete hashing techniques to encode the user and item vectors via binary codes, which makes it possible to store massive vectors effectively. We then extend DCF to PHOCCF, and design a global model updating module and a personal model updating module to update the parameters.
- We design a memorization module called cache updating module to address the computational bottlenecks. By deriving the update formulas, we enable some terms to be independent of the items or users, and then pre-calculate them only once in each training round.

Related Work (1/4)

- Federated learning aims to collaboratively train a global machine learning model among multiple clients without sharing the raw data.
- Federated recommendation presents a new challenge of storage overhead for the embeddings.
  - Traditional recommendation systems often use unique identifiers (IDs) to represent items and then generate some trainable embeddings for items.
  - The size of the embeddings will increase as the number of items increases, and they account for the majority of storage compared to other neural network parameters.
- Discrete hashing techniques provide a promising alternative for the embeddings by encoding real-valued embeddings via binary codes.

Related Work (2/4)

- Federated recommendation aims to make accurate recommendations in a distributed machine learning paradigm in a privacy-preserving manner.
- The most relevant works are DeepRec [Han et al., 2021] and LightFR [Zhang et al., 2022].

Related Work (3/4)

- DeepRec [Han et al., 2021] is an on-device deep learning framework for privacy-preserving sequential recommendation.
- Our DFMR is inspired by DeepRec, in which the authors argue that collecting the business necessary data does not violate the privacy-related laws.
- There are two significant differences between our DFMR and DeepRec.
  - The problem we study is HOCCF while the problem they study is single-behavior sequential recommendation.
  - DeepRec is not a federated learning method. Our training process is the same as that of most traditional federated recommendation methods.

Related Work (4/4)

- LightFR [Zhang et al., 2022] is a federated version of DCF [Zhang et al., 2016], which encodes the item and user vectors via binary codes.
- We discuss the differences among DCF, LightFR and our DFMR.
  - In LightFR and DCF, the authors study the item ranking task with explicit feedback instead of heterogeneous implicit feedback in this paper.
  - For implicit feedback, a constraint term which lies in the summation over the missing data is required in the objection function and will cause the computational bottleneck. Thus, we need an additional module to reduce the computational complexity, for which LightFR and DCF do not need.
  - Our DFMR is able to exploit the difference between examinations and purchases to improve the recommendation performance, which can thus outperform DCF with implicit feedback.

DFMR

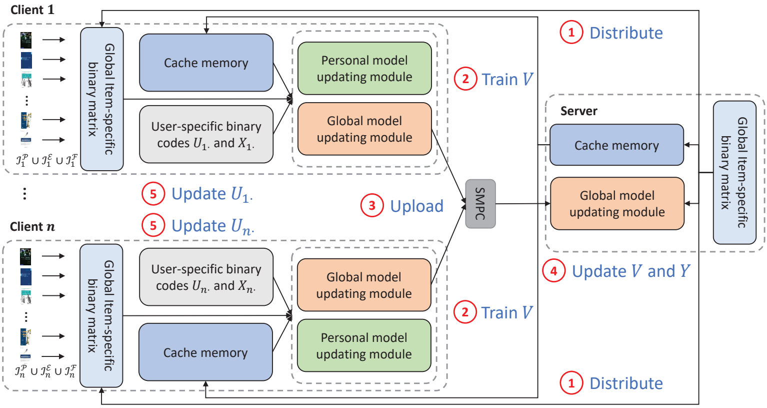


Figure: The framework of our discrete federated multi-behavior recommendation (DFMR).

Objective Function

$$\arg\min_{U,V,X,Y} \mathcal{L}_{\text{DCF}} + \mathcal{L}_{\text{Purc}} + \mathcal{L}_{\text{Exam}} + \mathcal{L}_{\text{Reg}}$$
$$\text{s.t. } \mathbf{1}^T X = \mathbf{0}, X^T X = n\mathbf{I}, \mathbf{1}^T Y = \mathbf{0}, Y^T Y = m\mathbf{I} \quad (1)$$
$$U \in \{\pm 1\}^{n\times d}, V \in \{\pm 1\}^{m\times d},$$

where $\mathcal{L}_{\text{DCF}} = \sum_{u\in\mathcal{U}} \sum_{i\in\mathcal{I}_u^{\mathcal{P}}} w_{ui}(r_{ui} - \hat{r}_{ui})^2 + s\sum_{u\in\mathcal{U}} \sum_{k\in\mathcal{I}\setminus\mathcal{I}_u^{\mathcal{P}}} \hat{r}_{uk}^2$, $\mathcal{L}_{\text{Purc}} =$ $\lambda \sum_{u\in\mathcal{U}} \sum_{i\in\mathcal{I}_u^{\mathcal{P}}} c_i \left[\sum_{j\in\mathcal{I}_u^{\mathcal{E}}} \left(\gamma_1 - (\hat{r}_{ui} - \hat{r}_{uj})\right)^2 + \sum_{k\in\mathcal{I}\setminus(\mathcal{I}_u^{\mathcal{P}}\cup\mathcal{I}_u^{\mathcal{E}})} \left(\gamma_2 - (\hat{r}_{ui} - \hat{r}_{uk})\right)^2\right]$,
$\mathcal{L}_{\text{Exam}} = (1-\lambda) \sum_{u\in\mathcal{U}} \sum_{j\in\mathcal{I}_u^{\mathcal{E}}} c_j \sum_{k\in\mathcal{I}\setminus(\mathcal{I}_u^{\mathcal{P}}\cup\mathcal{I}_u^{\mathcal{E}})} \left(\gamma_3 - (\hat{r}_{uj} - \hat{r}_{uk})\right)^2$,
$\mathcal{L}_{\text{Reg}} = -2\alpha tr\left(UX^T\right) - 2\beta tr\left(VY^T\right)$.

Note that $X \in \mathbb{R}^{n\times d}$ and $Y \in \mathbb{R}^{m\times d}$ are used to relax the balance and decorrelation constraints [Zhang et al., 2016].

Modules Overview

- **Global Model Updating Module.**
- Personal Model Updating Module.
- Cache Updating Module.

Global Model Updating Module

Those parameters (i.e., V and Y) which are independent of the user preference are treated as the global parameters and can be maintained in the server.

We denote V_{if} as the f-th bit of $V_{i\cdot}$ and $V_{i\bar{f}}$ as the rest binary vector excluding V_{if}.

For V_{if}, we fix U, X, Y and $V_{i\bar{f}}$ to be constant and then the original objective function can be represented as the objective function of V_{if}, i.e.,

$$\arg\min_{V_{if}\in\{\pm 1\}} -V_{if}V_{if}^{*}, \tag{2}$$

Global Model Updating Module

$$
\begin{aligned}
V_{if}^{*} = & \sum_{u \in \mathcal{U}_i^{\mathcal{P}}} \Bigg[\frac{w_{ui}\left(d\left(2r_{ui}-1\right) - U_{u\bar{f}} V_{i\bar{f}}^{T}\right)}{2d^2} U_{uf} + \sum_{j \in \mathcal{I}_u^{\mathcal{E}}} \frac{\lambda c_i}{2d^2} \left(V_{jf} + 2d\gamma_1 U_{uf} - U_{uf} U_{u\bar{f}} \left(V_{i\bar{f}}^{T} - V_{j\bar{f}}^{T}\right)\right) \\
& + \sum_{k \in \mathcal{I} \setminus \left(\mathcal{I}_u^{\mathcal{P}} \cup \mathcal{I}_u^{\mathcal{E}}\right)} \frac{\lambda c_i}{2d^2} \left(V_{kf} + 2d\gamma_2 U_{uf} - U_{uf} U_{u\bar{f}} \left(V_{i\bar{f}}^{T} - V_{k\bar{f}}^{T}\right)\right) \Bigg] \\
& + \sum_{u \in \mathcal{U}_i^{\mathcal{E}}} \Bigg[-\frac{s\left(d + U_{u\bar{f}} V_{i\bar{f}}^{T}\right)}{2d^2} U_{uf} + \sum_{t \in \mathcal{I}_u^{\mathcal{P}}} \frac{\lambda c_t}{2d^2} \left(V_{tf} - 2d\gamma_1 U_{uf} + U_{uf} U_{u\bar{f}} \left(V_{t\bar{f}}^{T} - V_{i\bar{f}}^{T}\right)\right) \\
& + \sum_{k \in \mathcal{I} \setminus \left(\mathcal{I}_u^{\mathcal{P}} \cup \mathcal{I}_u^{\mathcal{E}}\right)} \frac{(1-\lambda)\, c_i}{2d^2} \left(V_{kf} + 2d\gamma_3 U_{uf} - U_{uf} U_{u\bar{f}} \left(V_{i\bar{f}}^{T} - V_{k\bar{f}}^{T}\right)\right) \Bigg] \\
& - \sum_{u \in \mathcal{U} \setminus \left(\mathcal{U}_i^{\mathcal{P}} \cup \mathcal{U}_i^{\mathcal{E}}\right)} \Bigg[\frac{s\left(d + U_{u\bar{f}} V_{i\bar{f}}^{T}\right)}{2d^2} U_{uf} - \sum_{t \in \mathcal{I}_u^{\mathcal{P}}} \frac{\lambda c_t}{2d^2} \left(V_{tf} - 2d\gamma_2 U_{uf} + U_{uf} U_{u\bar{f}} \left(V_{t\bar{f}}^{T} - V_{i\bar{f}}^{T}\right)\right) \\
& - \sum_{j \in \mathcal{I}_u^{\mathcal{E}}} \frac{(1-\lambda)\, c_j}{2d^2} \left(V_{jf} - 2d\gamma_3 U_{uf} + U_{uf} U_{u\bar{f}} \left(V_{j\bar{f}}^{T} - V_{i\bar{f}}^{T}\right)\right) \Bigg] + 2\beta Y_{if}.
\end{aligned}
\tag{3}
$$

Please refer to Appendix B of our paper for the detailed derivations of V_{if}^{*}.

Global Model Updating Module

The update rule of V_{if} can be derived as

$$V_{if} = \mathrm{sgn}\left(K(V_{if}, V_{if}^{*})\right), \tag{4}$$

where $K(x, y)$ is a function that $K(x, y) = y$ if $y \neq 0$ and $K(x, y) = x$ otherwise, and $\mathrm{sgn}(x)$ is a function with $\mathrm{sgn}(x) = 1$ if $x > 0$ and $\mathrm{sgn}(x) = -1$ otherwise.

Global Model Updating Module

The objective function of Y can be represented as follows,

$$\arg\max_{Y} tr\left(VY^T\right), \text{s.t. } \mathbf{1}^T Y = \mathbf{0},\ Y^T Y = m\mathbf{I}. \quad (5)$$

Let $\bar{V}_{if} = V_{if} - \frac{1}{m}\sum_{i=1}^{m} V_{if}$, $\bar{V}^T\bar{V} = \begin{bmatrix} P_V & \hat{P}_V \end{bmatrix} \begin{bmatrix} \sum_V^2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} P_V & \hat{P}_V \end{bmatrix}^T$,

where $P_V \in \mathbb{R}^{m\times d'}$ are the left singular vectors corresponding to the d' positive singular values in the diagonal matrix Σ_V and $\hat{P}_V \in \mathbb{R}^{m\times(d-d')}$ are the eigenvectors of the zero eigenvalues.

Let $Q_V = \bar{V}P_V\Sigma_V^{-1}$, we obtain $\hat{Q}_V \in \mathbb{R}^{m\times(d-d')}$ by Gram-Schmidt orthogonalization [Selek et al., 2022] based on $[Q_V\ \mathbf{1}]$.

The update rule of Y can be derived as

$$Y'^T = \sqrt{m}\begin{bmatrix} P_V & \hat{P}_V \end{bmatrix} \begin{bmatrix} Q_V & \hat{Q}_V \end{bmatrix}^T. \quad (6)$$

Modules Overview

- Global Model Updating Module.
- **Personal Model Updating Module.**
- Cache Updating Module.

Personal Model Updating Module

$U_{u\cdot}$ and $X_{u\cdot}$ are closely related to user u's preference, thus they are treated as the personal parameters.

We denote U_{uf} as the f-th bit of $U_{u\cdot}$ and $U_{u\bar{f}}$ as the rest binary vector excluding U_{uf}. The objective function of U_{uf} can be represented as follows,

$$\arg\min_{U_{uf}\in\{\pm 1\}} -U_{uf}U^*_{uf}, \tag{7}$$

The update rule of U_{uf} can be represented as follows,

$$U_{uf} = \operatorname{sgn}\left(K\left(U_{uf}, U^*_{uf}\right)\right). \tag{8}$$

Personal Model Updating Module

$$
\begin{aligned}
U^*_{uf} = & \sum_{i \in \mathcal{I}^{\mathcal{P}}_u} \frac{w_{ui}\left(d\left(2r_{ui}-1\right) - U_{u\bar{f}} V^T_{i\bar{f}}\right)}{2d^2} V_{if} - \sum_{k \in \mathcal{I} \setminus \mathcal{I}^{\mathcal{P}}_u} \frac{s\left(d + U_{u\bar{f}} V^T_{k\bar{f}}\right)}{2d^2} V_{kf} + 2\alpha X_{uf} \\
& + \lambda \sum_{i \in \mathcal{I}^{\mathcal{P}}_u} \frac{c_i}{d} \sum_{j \in \mathcal{I}^{\mathcal{E}}_u} \left(2d\gamma_1 - U_{u\bar{f}} V^T_{i\bar{f}} + U_{u\bar{f}} V^T_{j\bar{f}}\right)\left(V_{if} - V_{jf}\right) \\
& + \lambda \sum_{i \in \mathcal{I}^{\mathcal{P}}_u} \frac{c_i}{d} \sum_{k \in \mathcal{I} \setminus (\mathcal{I}^{\mathcal{P}}_u \cup \mathcal{I}^{\mathcal{E}}_u)} \left(2d\gamma_2 - U_{u\bar{f}} V^T_{i\bar{f}} + U_{u\bar{f}} V^T_{k\bar{f}}\right)\left(V_{if} - V_{kf}\right) \\
& + (1-\lambda) \sum_{j \in \mathcal{I}^{\mathcal{E}}_u} \frac{c_j}{d} \sum_{k \in \mathcal{I} \setminus (\mathcal{I}^{\mathcal{P}}_u \cup \mathcal{I}^{\mathcal{E}}_u)} \left(2d\gamma_3 - U_{u\bar{f}} V^T_{j\bar{f}} + U_{u\bar{f}} V^T_{k\bar{f}}\right)\left(V_{jf} - V_{kf}\right).
\end{aligned}
\tag{9}
$$

Please refer to Appendix D of our paper for the detailed derivations of U^*_{uf}.

Personal Model Updating Module

The objective function and the update rule of X can be represented as follows,

$$\arg\max_{X} tr\left(UX^T\right), \text{s.t. } \mathbf{1}^T X = \mathbf{0}, X^T X = n\mathbf{I}, \tag{10}$$

$$X'^T = \sqrt{n}\begin{bmatrix} P_U & \hat{P}_U \end{bmatrix}\begin{bmatrix} Q_U & \hat{Q}_U \end{bmatrix}^T, \tag{11}$$

where P_U, $\hat{P}_U$ and Q_U are calculated based on the user-specific latent feature matrix U detailed in Appendix E.

Modules Overview

- Global Model Updating Module.
- Personal Model Updating Module.
- **Cache Updating Module**.

Cache Updating Module

For each item i's f-th bit V_{if}, all the clients need to send the corresponding parameters to the server.

As the number of items increases, the communication overhead on the clients increases linearly.

It is expected that the clients' communication overhead corresponds to the size of their interaction data, i.e., only the users who have interacted with item i need to communicate with the server.

Cache Updating Module

We rewrite V_{if}^{*} as follows,

$$\begin{aligned} V_{if}^{*} = & \sum_{u \in \mathcal{U}_i^{\mathcal{P}}} \mathrm{C}_{uif}^{\mathcal{P}} + \sum_{u \in \mathcal{U}_i^{\mathcal{E}}} \mathrm{C}_{uif}^{\mathcal{E}} + \sum_{u \in \mathcal{U}} \mathrm{C}_{uif} + 2\beta Y_{if} + \frac{\lambda}{2d^2} \sum_{t \in \mathcal{I}} \sum_{u \in \mathcal{U}_t^{\mathcal{P}}} U_{uf} U_{u\bar{f}} c_t V_{t\bar{f}}^{T} \\ & + \frac{(1-\lambda)}{2d^2} \sum_{t \in \mathcal{I}} \sum_{u \in \mathcal{U}_t^{\mathcal{E}}} U_{uf} U_{u\bar{f}} c_t V_{t\bar{f}}^{T} + \frac{\lambda}{2d^2} \sum_{t \in \mathcal{I}} c_t V_{tf} \left| \mathcal{U}_t^{\mathcal{P}} \right| + \frac{(1-\lambda)}{2d^2} \sum_{t \in \mathcal{I}} c_t V_{tf} \left| \mathcal{U}_t^{\mathcal{E}} \right|, \end{aligned} \quad (12)$$

The detailed derivations of $\mathrm{C}_{uif}^{\mathcal{P}}$, $\mathrm{C}_{uif}^{\mathcal{E}}$ and C_{uif} can be found in Appendix B.

Cache Updating Module

We can observe that the terms

$$\left(\frac{s}{2d}+\frac{\lambda\gamma_2}{d}\sum_{t\in\mathcal{I}_u^{\mathcal{P}}}c_t+\frac{(1-\lambda)\,\gamma_3}{d}\sum_{j\in\mathcal{I}_u^{\mathcal{E}}}c_j\right)U_{uf}$$

and

$$U_{uf}U_{u\bar{f}}\left(\frac{s}{2d^2}+\frac{\lambda}{2d^2}\sum_{t\in\mathcal{I}_u^{\mathcal{P}}}c_t+\frac{1-\lambda}{2d^2}\sum_{j\in\mathcal{I}_u^{\mathcal{E}}}c_j\right)$$

in C_{uif} are independent of the item i. Hence, client u can pre-calculate and send them to the server only once in each training round.

Cache Updating Module

Similarly,

$$\frac{\lambda}{2d^2}\sum_{t\in\mathcal{I}}\sum_{u\in\mathcal{U}_t^{\mathcal{P}}} U_{uf}U_{u\bar{f}}c_tV_{t\bar{f}}^T + \frac{(1-\lambda)}{2d^2}\sum_{t\in\mathcal{I}}\sum_{u\in\mathcal{U}_t^{\mathcal{E}}} U_{uf}U_{u\bar{f}}c_tV_{t\bar{f}}^T + \frac{\lambda}{2d^2}\sum_{t\in\mathcal{I}} c_tV_{tf}\left|\mathcal{U}_t^{\mathcal{P}}\right| + \frac{(1-\lambda)}{2d^2}\sum_{t\in\mathcal{I}} c_tV_{tf}\left|\mathcal{U}_t^{\mathcal{E}}\right|$$

can also be pre-calculated and sent to the server only once in each training round.

Cache Updating Module

We design a memorization strategy to calculate the terms of the summation of the parameters related to all items, such as $\sum_{k\in\mathcal{I}} V_{kf}$.

For example, we can define $\mathrm{q}_1^f = \sum_{t\in\mathcal{I}} V_{tf}$ and $\mathbf{p}_1^f = \sum_{t\in\mathcal{I}} V_{tf}^T$. By leveraging the above memorization strategy in those terms, we can rewrite V_{if}^* as

$$V_{if}^* = \sum_{u\in\mathcal{U}_i^{\mathcal{P}}} \mathrm{C}_{uif}^{\mathcal{P}} + \sum_{u\in\mathcal{U}_i^{\mathcal{E}}} \mathrm{C}_{uif}^{\mathcal{E}} + \mathrm{q}_2^f + \mathbf{p}_2^f V_{if}^T + 2\beta Y_{if} + \frac{\lambda}{2d^2}\mathrm{h}_1^f + \frac{(1-\lambda)}{2d^2}\mathrm{h}_2^f + \frac{\lambda}{2d^2}\mathrm{h}_3^f + \frac{(1-\lambda)}{2d^2}\mathrm{h}_4^f, \tag{13}$$

where $\mathrm{C}_{uif}^{\mathcal{P}}$ and $\mathrm{C}_{uif}^{\mathcal{E}}$ have been rewritten through the memorization strategy. Similarly, we can rewrite U_{uf}^* with the above strategy.

The detailed derivations can be found in Appendix F.

Algorithm 1 The algorithm of DFMR in the server.

```
Input: The purchase matrix P ∈ {0,1}^{n×m} and the hyperparameters.
Initialize and pre-train U, V, X and Y.
Distribute U_{u·}, X_{u·} and the hyperparameters to client u.
Calculate |𝒰_i^ℰ|, i ∈ 1, 2, ⋯, m via secret sharing.
// Start training
for t = 1 to T do
    // Global model updating
    Update the cache q_1^f, p_1^f, f ∈ 1, 2, ⋯, d.
    Update the cache q_2^f, p_2^f, h_1^f, h_2^f, f ∈ 1, 2, ⋯, d via secret sharing.
    Update the cache h_3^f and h_4^f, f ∈ 1, 2, ⋯, d.
    for u = 1 to n do
        Distribute q_1^f and p_1^f, f ∈ 1, 2, ⋯, d to client u.
        Distribute V to client u.
    end for
    for i = 1 to m do
        for f = 1 to d do
```

Receive $C_{uif}^{\mathcal{P}\cup\mathcal{E}\cup\mathcal{F}}$ from client $u \in \mathcal{U}_i^{\mathcal{P}\cup\mathcal{E}\cup\mathcal{F}}$.
Update $V_{if} = sgn\left(K\left(V_{if}, V_{if}^{*}\right)\right)$.
if V_{if} changes **then**
Distribute V_{if} to client $u \in \mathcal{U}_i^{\mathcal{P}\cup\mathcal{E}\cup\mathcal{F}}$.
end if
end for
end for
Update Y.
// Personal model updating
Update the cache q_5^f, $\mathbf{p}_5^f$, q_6^f, $\mathbf{p}_6^f$ and $\mathbf{p}_7^f$, $f \in 1, 2, \cdots, d$.
for $u = 1$ **to** n **do**
Distribute q_5^f, $\mathbf{p}_5^f$, q_6^f, $\mathbf{p}_6^f$ and $\mathbf{p}_7^f$, $f \in 1, 2, \cdots, d$ to client u, $f \in 1, 2, \cdots, d$.
end for
end for

Algorithm 2 The algorithm of DFMR on the client u.

Input: A set of purchased items $\mathcal{I}_u^{\mathcal{P}}$, a set of examined items $\mathcal{I}_u^{\mathcal{E}}$, the sampling parameter ρ.
Receive $U_{u\cdot}$, $X_{u\cdot}$ and the hyperparameters from the server.
Randomly generated $\mathcal{I}_u^{\mathcal{F}}$, where $\left|\mathcal{I}_u^{\mathcal{F}}\right| = \rho\left(\left|\mathcal{I}_u^{\mathcal{P}}\right| + \left|\mathcal{I}_u^{\mathcal{E}}\right|\right)$.
Calculate $\left|\mathcal{U}_i^{\mathcal{E}}\right|$, $i \in \mathcal{I}_u^{\mathcal{E}} \cup \mathcal{I}_u^{\mathcal{F}}$ via secret sharing.
// Start training
for $t = 1$ **to** T **do**
    // Global model updating
    Update the cache q_2^f, $\mathbf{p}_2^f$, $\mathbf{p}_3^f$, $\mathbf{p}_4^f$, $f \in 1, 2, \cdots, d$ via secret sharing.
    Receive q_1^f and $\mathbf{p}_1^f$, $f \in 1, 2, \cdots, d$ from the server.
    Receive V from the server.
    for $i = 1$ **to** m **do**
        for $f = 1$ **to** d **do**
            if $i \in \mathcal{I}_u^{\mathcal{P}} \cup \mathcal{I}_u^{\mathcal{E}} \cup \mathcal{I}_u^{\mathcal{F}}$ **then**

$$C_{uif}^{\mathcal{I}_u^{\mathcal{P}} \cup \mathcal{I}_u^{\mathcal{E}} \cup \mathcal{I}_u^{\mathcal{F}}} = \begin{cases} C_{uif}^{\mathcal{P}} & \text{if } i \in \mathcal{I}_u^{\mathcal{P}} \\ C_{uif}^{\mathcal{E}} & \text{if } i \in \mathcal{I}_u^{\mathcal{E}} \\ 0 & \text{if } i \in \mathcal{I}_u^{\mathcal{F}} \end{cases}$$

Upload $C_{uif}^{\mathcal{I}_u^{\mathcal{P}} \cup \mathcal{I}_u^{\mathcal{E}} \cup \mathcal{I}_u^{\mathcal{F}}}$ to the server via secret sharing.
if V_{if} changes **then**
Update V_{if}.
Update the cache q_1^f and $\mathbf{p}_1^f$ locally.
end if
end if
end for
end for
// Personal model updating
Receive the cache q_5^f, $\mathbf{p}_5^f$, q_6^f, $\mathbf{p}_6^f$, and $\mathbf{p}_7^f$, $f \in 1, 2, \cdots, d$ from the server.
Update $U_{u\cdot}$.
end for

Space Complexity

- The server requires $d\,(5d+3)\,B$ bits for storing the cache, $2dm$ bits for storing V and Y.
- Each client u requires $d\,(4d-1)\,B$ bits for storing the cache, dm bits for storing the item-specific binary matrix and $2d$ bits for storing $U_{u\cdot}$ and $X_{u\cdot}$, where B is the bit-length of the data.
- We usually have $dB \ll m$, which means that the space complexity of the server and the clients are $\mathcal{O}\,(dm)$.

Communication Complexity

- The communication complexity of the server is $\mathcal{O}(dnmT)$.
- The communication complexity of the client u is $\mathcal{O}(dmT)$.

Computational Complexity

- The computational complexity of the server is $\mathcal{O}(dsT)$.
- The computational complexity of each client u is $\mathcal{O}\left(d\left(|\mathcal{I}_u^{\mathcal{P}}| + |\mathcal{I}_u^{\mathcal{E}}|\right)^2 T\right)$.

Privacy analysis

- Our DFMR protects a portion of the privacy-sensitive raw data without affecting the basic business.
- The personal parameter $U_{u\cdot}$, which is related to the user preference, is always kept on each client.
- All the uploaded intermediate parameters are protected via secret sharing and fake items techniques [Lin et al., 2022].

Research Questions

- RQ1: How does our DFMR perform compared with the baseline methods?
- RQ2: What is the superiority of our proposed cache updating module?
- RQ3: Comparing with real-valued methods, does the space overhead reduce significantly?

Datasets

- We conduct experiments on four public datasets including JD, Tmall, User Behavior (UB) and MovieLens 10M (ML10M).
- For the sake of simplicity, we only preserve two types of behaviors, i.e., examinations and purchases.
- For the simulation, we regard the rating behaviors with scores greater than or equal to 4 as purchase behaviors and the rest behaviors as examination behaviors on ML10M.

Datasets

We preprocess the datasets as follows:

- For duplicated (user, item, behavior) tuples in a sequence, we only retain the earliest one;
- We discard the cold-start items with fewer than 20 purchase interactions and the cold-start users with fewer than 5 purchase interactions;
- We remove the records from Tmall on the special sales promotion day November 11;
- We sort all the records according to the timestamp in ascending order, and then take the data front of the 80% timeline as the training set, that of 80%-90% as the validation set, and that after 90% as the test set;
- We discard the examination records which the same users have both examined and purchased.

Datasets

Table: Statistics of the processed datasets.

Dataset	JD	Tmall	UB	ML10M
#Users	10,690	17,202	20,443	3,625
#Items	13,465	16,177	30,947	3,633
P	60,967	201,424	107,489	152,013
E	217,977	799,352	511,020	932,155
P(val.)	5,892	23,758	13,088	23,124
P(te.)	5,013	15,646	13,131	14,154
P/E	1:3.58	1:3.97	1:4.75	1:6.13
Density	0.19%	0.36%	0.10%	8.23%

Baselines

Four centralized learning-based algorithms:

- LogMF [Johnson and C, 2014] is a classic one-class collaborative filtering (OCCF) algorithm that adopts a pointwise preference assumption, which learns the user and item vectors via the stochastic gradient descent (SGD) technique using a logistic loss function.
- eALS [He et al., 2016] is a non-sampling-based OCCF algorithm, which learns the user and item vectors via the element-wise alternative least square technique [He et al., 2016] using a root mean square error (RMSE) loss function.

Baselines

- VALS [Ding et al., 2018] is a non-sampling-based heterogeneous one-class collaborative filtering (HOCCF) algorithm that models the pairwise relations among the purchase data, the examination data and the un-interacted data, which learns the user and item vectors via the element-wise alternative least square technique using an RMSE loss function.
- DCF [Zhang et al., 2016] is a non-sampling-based discrete OCCF algorithm, which learns the user and item vectors via the discrete coordinate descent technique [Farsa and Rahnamayan, 2020] using an RMSE loss function.

Baselines

Two federated learning-based algorithms:

- MF_FedAVG is a federated version of LogMF, which uses the FedAVG algorithm [McMahan et al., 2017] to aggregate the items' gradients in the server.
- FedGNN [Wu et al., 2022] is a GNN-based federated OCCF algorithm, which proposes to use a trusted third-party server to construct similar user neighborhoods for each client to learn high-order graph information.

Performance Evaluation (RQ1)

Table: Recommendation performance of MF, eALS, VALS, MF_FedAVG, FedGNN, DCF and our DFMR on JD.

Metrics	MF	eALS	VALS	MF_FedAVG	FedGNN	DCF	DFMR
NDCG@5	$0.0538_{\pm 0.0044}$	$0.0556_{\pm 0.0007}$	$0.0581_{\pm 0.0005}$	$0.0510_{\pm 0.0013}$	$0.0598_{\pm 0.0014}$	$0.0512_{\pm 0.0017}$	$0.0539_{\pm 0.0008}$
Pre@5	$0.0310_{\pm 0.0020}$	$0.0308_{\pm 0.0009}$	$0.0326_{\pm 0.0003}$	$0.0302_{\pm 0.0001}$	$0.0304_{\pm 0.0012}$	$0.0290_{\pm 0.0021}$	$0.0324_{\pm 0.0011}$
Rec@5	$0.0651_{\pm 0.0052}$	$0.0684_{\pm 0.0004}$	$0.0727_{\pm 0.0012}$	$0.0601_{\pm 0.0001}$	$0.0733_{\pm 0.0034}$	$0.0633_{\pm 0.0042}$	$0.0701_{\pm 0.0031}$
NDCG@10	$0.0714_{\pm 0.0037}$	$0.0768_{\pm 0.0003}$	$0.0775_{\pm 0.0005}$	$0.0686_{\pm 0.0016}$	$0.0802_{\pm 0.0008}$	$0.0686_{\pm 0.0024}$	$0.0712_{\pm 0.0023}$
Pre@10	$0.0272_{\pm 0.0004}$	$0.0291_{\pm 0.0001}$	$0.0291_{\pm 0.0002}$	$0.0274_{\pm 0.0004}$	$0.0279_{\pm 0.0010}$	$0.0267_{\pm 0.0013}$	$0.0283_{\pm 0.0012}$
Rec@10	$0.1094_{\pm 0.0036}$	$0.1195_{\pm 0.0013}$	$0.1210_{\pm 0.0013}$	$0.1040_{\pm 0.0023}$	$0.1257_{\pm 0.0037}$	$0.1039_{\pm 0.0063}$	$0.1095_{\pm 0.0048}$

Table: Recommendation performance of MF, eALS, VALS, MF_FedAVG, FedGNN, DCF and our DFMR on Tmall.

Metrics	MF	eALS	VALS	MF_FEDAVG	FedGNN	DCF	DFMR
NDCG@5	$0.0023_{\pm 0.0006}$	$0.0024_{\pm 0.0001}$	$0.0027_{\pm 0.0000}$	$0.0018_{\pm 0.0004}$	$0.0024_{\pm 0.0004}$	$0.0011_{\pm 0.0003}$	$0.0018_{\pm 0.0002}$
Pre@5	$0.0013_{\pm 0.0003}$	$0.0014_{\pm 0.0001}$	$0.0015_{\pm 0.0000}$	$0.0011_{\pm 0.0002}$	$0.0013_{\pm 0.0002}$	$0.0008_{\pm 0.0002}$	$0.0011_{\pm 0.0001}$
Rec@5	$0.0030_{\pm 0.0006}$	$0.0030_{\pm 0.0004}$	$0.0032_{\pm 0.0000}$	$0.0022_{\pm 0.0006}$	$0.0031_{\pm 0.0005}$	$0.0017_{\pm 0.0004}$	$0.0023_{\pm 0.0004}$
NDCG@10	$0.0032_{\pm 0.0005}$	$0.0032_{\pm 0.0002}$	$0.0036_{\pm 0.0000}$	$0.0024_{\pm 0.0004}$	$0.0031_{\pm 0.0003}$	$0.0020_{\pm 0.0003}$	$0.0026_{\pm 0.0003}$
Pre@10	$0.0012_{\pm 0.0001}$	$0.0011_{\pm 0.0001}$	$0.0013_{\pm 0.0000}$	$0.0009_{\pm 0.0001}$	$0.0011_{\pm 0.0001}$	$0.0009_{\pm 0.0001}$	$0.0010_{\pm 0.0001}$
Rec@10	$0.0055_{\pm 0.0006}$	$0.0049_{\pm 0.0006}$	$0.0058_{\pm 0.0001}$	$0.0037_{\pm 0.0006}$	$0.0050_{\pm 0.0002}$	$0.0040_{\pm 0.0005}$	$0.0044_{\pm 0.0006}$

Performance Evaluation (RQ1)

Table: Recommendation performance of MF, eALS, VALS, MF_FedAVG, FedGNN, DCF and our DFMR on UB.

Metrics	MF	eALS	VALS	MF_FEDAVG	FedGNN	DCF	DFMR
NDCG@5	$0.0040_{\pm 0.0003}$	$0.0040_{\pm 0.0002}$	$0.0055_{\pm 0.0001}$	$0.0029_{\pm 0.0002}$	$0.0041_{\pm 0.0005}$	$0.0017_{\pm 0.0003}$	$0.0025_{\pm 0.0007}$
Pre@5	$0.0019_{\pm 0.0002}$	$0.0021_{\pm 0.0001}$	$0.0030_{\pm 0.0000}$	$0.0015_{\pm 0.0001}$	$0.0020_{\pm 0.0003}$	$0.0009_{\pm 0.0001}$	$0.0014_{\pm 0.0004}$
Rec@5	$0.0056_{\pm 0.0005}$	$0.0055_{\pm 0.0004}$	$0.0072_{\pm 0.0001}$	$0.0039_{\pm 0.0002}$	$0.0058_{\pm 0.0006}$	$0.0026_{\pm 0.0006}$	$0.0035_{\pm 0.0013}$
NDCG@10	$0.0054_{\pm 0.0001}$	$0.0058_{\pm 0.0002}$	$0.0072_{\pm 0.0001}$	$0.0040_{\pm 0.0002}$	$0.0053_{\pm 0.0006}$	$0.0026_{\pm 0.0003}$	$0.0035_{\pm 0.0008}$
Pre@10	$0.0016_{\pm 0.0001}$	$0.0019_{\pm 0.0001}$	$0.0024_{\pm 0.0000}$	$0.0014_{\pm 0.0001}$	$0.0016_{\pm 0.0001}$	$0.0009_{\pm 0.0001}$	$0.0012_{\pm 0.0002}$
Rec@10	$0.0096_{\pm 0.0003}$	$0.0104_{\pm 0.0007}$	$0.0123_{\pm 0.0001}$	$0.0071_{\pm 0.0006}$	$0.0090_{\pm 0.0007}$	$0.0048_{\pm 0.0005}$	$0.0062_{\pm 0.0014}$

Table: Recommendation performance of MF, eALS, VALS, MF_FedAVG, FedGNN, DCF and our DFMR on ML10M.

Metrics	MF	eALS	VALS	MF_FEDAVG	FedGNN	DCF	DFMR
NDCG@5	$0.0809_{\pm 0.0021}$	$0.0355_{\pm 0.0010}$	$0.0536_{\pm 0.0003}$	$0.0491_{\pm 0.0007}$	$0.0889_{\pm 0.0030}$	$0.0542_{\pm 0.0027}$	$0.0566_{\pm 0.0029}$
Pre@5	$0.0696_{\pm 0.0023}$	$0.0476_{\pm 0.0024}$	$0.0447_{\pm 0.0004}$	$0.0429_{\pm 0.0006}$	$0.0750_{\pm 0.0017}$	$0.0787_{\pm 0.0071}$	$0.0825_{\pm 0.0051}$
Rec@5	$0.0455_{\pm 0.0016}$	$0.0358_{\pm 0.0017}$	$0.0339_{\pm 0.0011}$	$0.0272_{\pm 0.0012}$	$0.0510_{\pm 0.0014}$	$0.0503_{\pm 0.0028}$	$0.0520_{\pm 0.0020}$
NDCG@10	$0.0876_{\pm 0.0005}$	$0.0485_{\pm 0.0008}$	$0.0578_{\pm 0.0006}$	$0.0522_{\pm 0.0013}$	$0.0950_{\pm 0.0027}$	$0.0763_{\pm 0.0016}$	$0.0770_{\pm 0.0026}$
Pre@10	$0.0635_{\pm 0.0003}$	$0.0407_{\pm 0.0010}$	$0.0391_{\pm 0.0007}$	$0.0385_{\pm 0.0009}$	$0.0676_{\pm 0.0041}$	$0.0743_{\pm 0.0023}$	$0.0744_{\pm 0.0029}$
Rec@10	$0.0827_{\pm 0.0009}$	$0.0609_{\pm 0.0006}$	$0.0568_{\pm 0.0008}$	$0.0470_{\pm 0.0006}$	$0.0871_{\pm 0.0024}$	$0.0878_{\pm 0.0002}$	$0.0847_{\pm 0.0026}$

Performance Evaluation (RQ1)

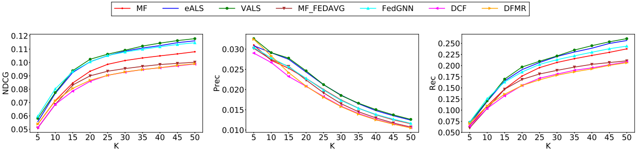


Figure: Recommendation performance (i.e., NDCG, Precision and Recall) with different values of K on JD.

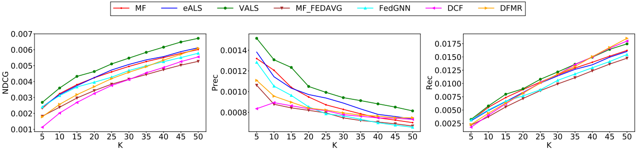


Figure: Recommendation performance (i.e., NDCG, Precision and Recall) with different values of K on Tmall.

Performance Evaluation (RQ1)

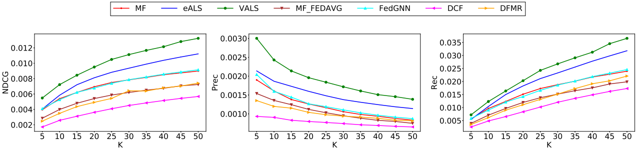


Figure: Recommendation performance (i.e., NDCG, Precision and Recall) with different values of K on UB.

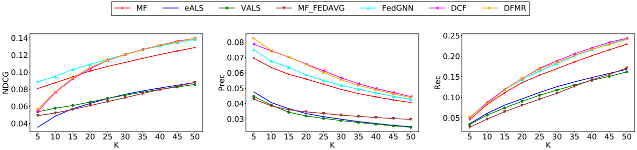


Figure: Recommendation performance (i.e., NDCG, Precision and Recall) with different values of K on ML10M.

Performance Evaluation (RQ1)

We have the following observations:

- For the three baselines exploiting homogeneous one-class feedback (i.e., MF, eALS and DCF), we can observe that DCF does not perform well compared with the two real valued-based baselines, i.e., MF and eALS, in most cases. It indicates that there is a performance gap between real valued-based models and binary-based models. Indeed, it is acceptable to sacrifice some accuracy for efficiency, but the hope is that the performance will not drop significantly.
- Moreover, our DFMR achieves the better performance than DCF on all the four datasets, which clearly shows the advantage of our generic solution by modeling different preference levels for the users' purchase, examination and un-interacted behaviors.
- Additionally, our DFMR can approach or even outperform MF and eALS on both JD and ML10M, which demonstrates the effectiveness of our DFMR.

Performance Evaluation (RQ1)

- For the baseline VALS exploiting heterogeneous one-class feedback, it outperforms all OCCF-based centralized baselines in most cases, which clearly indicates that the use of the examination behaviors can improve the recommendation performance. Moreover, it outperforms our DFMR on Tmall and UB. Instead, our DFMR can approach or even outperform VALS on JD and ML10M. It clearly indicates that in the case of binarizing the user and item vectors our DFMR does not sacrifice much in terms of recommendation performance.
- Compared with the two federated learning-based baselines (i.e., MF_FedAVG and FedGNN), we can observe that DFMR achieves better performance than MF_FedAVG on four datasets. And at the same time, it can approach and even outperform FedGNN. It clearly indicates that although our DFMR sacrifices some accuracy for efficiency, it is still a state-of-the-art federated learning-based recommendation method due to the exploitation for examination behaviors.

Superiority of the Cache Updating Module (RQ2)

Table: Time cost of the server and each client of one round with different numbers of dimensions on JD.

	Model	2	4	8	16	32
Server	eALS	$1.84e^{-2}$	$1.92e^{-2}$	$2.13e^{-2}$	$2.86e^{-2}$	$4.33e^{-2}$
	VALS	$1.34e^{-1}$	$2.89e^{-1}$	$8.09e^{-1}$	2.64	9.60
	DCF	$3.52e^{-2}$	$3.67e^{-2}$	$4.12e^{-2}$	$5.28e^{-2}$	$8.44e^{-2}$
	DFMR (w/ cache)	$2.20e^{-1}$	$2.66e^{-1}$	$5.17e^{-1}$	1.61	6.28
	DFMR (w/o cache)	$2.72e^{1}$	$5.45e^{2}$	$3.39e^{3}$	$2.82e^{4}$	$2.11e^{5}$
Client	eALS	$9.74e^{-8}$	$1.63e^{-7}$	$3.49e^{-7}$	$8.73e^{-7}$	$2.83e^{-6}$
	VALS	$2.91e^{-4}$	$3.03e^{-4}$	$3.41e^{-4}$	$4.77e^{-4}$	$9.86e^{-4}$
	DCF	$1.74e^{-6}$	$1.91e^{-6}$	$2.31e^{-6}$	$3.32e^{-6}$	$6.20e^{-6}$
	DFMR (w/ cache)	$3.33e^{-5}$	$9.32e^{-5}$	$3.02e^{-4}$	$1.07e^{-3}$	$4.33e^{-3}$
	DFMR (w/o cache)	$3.48e^{-3}$	$9.34e^{-3}$	$2.87e^{-2}$	$2.17e^{-1}$	$8.40e^{-1}$

Superiority of the Cache Updating Module (RQ2)

Table: Time cost of the server and each client of one round with different numbers of dimensions on Tmall.

	Model	2	4	8	16	32
Server	eALS	$2.39e^{-2}$	$2.59e^{-2}$	$3.10e^{-2}$	$5.09e^{-2}$	$1.18e^{-1}$
	VALS	$3.32e^{-1}$	$6.12e^{-1}$	1.42	3.99	$1.31e^{1}$
	DCF	$4.40e^{-2}$	$4.85e^{-2}$	$5.91e^{-2}$	$8.47e^{-2}$	$1.55e^{-1}$
	DFMR (w/ cache)	1.15	1.28	1.66	3.48	10.1
	DFMR (w/o cache)	$4.42e^{2}$	$3.02e^{3}$	$1.56e^{4}$	$1.17e^{5}$	$8.73e^{5}$
Client	eALS	$4.37e^{-7}$	$5.34e^{-7}$	$8.33e^{-7}$	$1.99e^{-6}$	$5.87e^{-6}$
	VALS	$1.20e^{-3}$	$1.21e^{-3}$	$1.25e^{-3}$	$1.40e^{-3}$	$1.92e^{-3}$
	DCF	$1.70e^{-6}$	$1.93e^{-6}$	$2.61e^{-6}$	$4.18e^{-6}$	$8.29e^{-6}$
	DFMR (w/ cache)	$1.01e^{-4}$	$2.64e^{-4}$	$8.49e^{-4}$	$2.99e^{-3}$	$1.21e^{-2}$
	DFMR (w/o cache)	$1.04e^{-2}$	$2.84e^{-2}$	$9.47e^{-2}$	$6.95e^{-1}$	3.33

Superiority of the Cache Updating Module (RQ2)

Table: Time cost of the server and each client of one round with different numbers of dimensions on UB.

	Model	2	4	8	16	32
Server	eALS	$2.30e^{-2}$	$2.45e^{-2}$	$2.95e^{-2}$	$4.86e^{-2}$	$1.26e^{-1}$
	VALS	$3.11e^{-1}$	$6.75e^{-1}$	1.85	6.03	$2.19e^{1}$
	DCF	$4.23e^{-2}$	$4.60e^{-2}$	$5.53e^{-2}$	$8.09e^{-2}$	$1.56e^{-1}$
	DFMR (w/ cache)	$4.96e^{-1}$	$6.62e^{-1}$	1.35	4.00	14.5
	DFMR (w/o cache)	$8.84e^{2}$	$3.66e^{3}$	$2.87e^{4}$	$1.94e^{5}$	$1.43e^{6}$
Client	eALS	$1.64e^{-7}$	$2.34e^{-7}$	$5.37e^{-7}$	$1.64e^{-6}$	$3.00e^{-6}$
	VALS	$1.78e^{-4}$	$1.90e^{-4}$	$2.31e^{-4}$	$3.71e^{-4}$	$9.01e^{-4}$
	DCF	$1.21e^{-6}$	$1.37e^{-6}$	$1.82e^{-6}$	$2.90e^{-6}$	$5.76e^{-6}$
	DFMR (w/ cache)	$3.72e^{-5}$	$9.48e^{-5}$	$2.68e^{-4}$	$1.04e^{-3}$	$3.88e^{-3}$
	DFMR (w/o cache)	$1.13e^{-2}$	$3.12e^{-2}$	$1.72e^{-1}$	$6.88e^{-1}$	3.56

Superiority of the Cache Updating Module (RQ2)

Table: Time cost of the server and each client of one round with different numbers of dimensions on ML10M.

	Model	2	4	8	16	32
Server	eALS	$9.26e^{-3}$	$9.45e^{-3}$	$1.23e^{-2}$	$1.91e^{-2}$	$3.42e^{-2}$
	VALS	$9.63e^{-2}$	$1.47e^{-1}$	$3.40e^{-1}$	$9.49e^{-1}$	3.06
	DCF	$111e^{-6}$	$111e^{-6}$	$111e^{-6}$	$111e^{-6}$	$111e^{-6}$
	DFMR (w/ cache)	1.36	1.17	1.58	2.87	7.78
	DFMR (w/o cache)	$2.00e^{2}$	$1.34e^{3}$	$1.07e^{4}$	$7.96e^{4}$	$6.76e^{5}$
Client	eALS	$1.22e^{-6}$	$1.44e^{-6}$	$2.32e^{-6}$	$4.14e^{-6}$	$9.04e^{-6}$
	VALS	$1.41e^{-3}$	$1.43e^{-3}$	$1.48e^{-3}$	$1.64e^{-3}$	$2.20e^{-3}$
	DCF	$3.21e^{-6}$	$3.96e^{-6}$	$5.79e^{-6}$	$9.80e^{-6}$	$1.95e^{-5}$
	DFMR (w/ cache)	$2.91e^{-3}$	$8.94e^{-3}$	$3.054e^{-2}$	$1.11e^{-1}$	$4.23e^{-1}$
	DFMR (w/o cache)	$1.74e^{-2}$	$4.52e^{-2}$	$1.34e^{-1}$	$4.40e^{-1}$	1.54

Superiority of the Cache Updating Module (RQ2)

We have the following observations:

- Our DFMR with cache is significantly faster than that without cache on four datasets. It indicates that our designed cache updating module can effectively reduce the computational overhead by pre-calculating the summation over the parameters of all the items or users independently for a corresponding specific item or user.
- For the OCCF-based methods, eALS and DCF, their computational time is of the same order of magnitude approximately, while the computational time of VALS and our DFMR with cache is also of the same order. It indicates that the discrete-based methods can be comparable to the real value-based methods in terms of the computational time.

Superiority of the Cache Updating Module (RQ2)

- When the dimension number increases, the computational time of our DFMR with cache increases linearly while that of our DFMR without cache increases exponentially. This result indicates that if the dimension number is large, our designed cache updating module can play a more important role in reducing the computational overhead.
- Even if we set a small dimension number in our DFMR without cache (e.g., $d = 2$), the computational time of the server is still larger than that of a large dimension number in our DFMR with cache (e.g., $d = 32$), which again shows the superiority of our designed cache updating module.

Superiority of the Cache Updating Module (RQ2)

Table: Communication cost (MB) of each client of one round with different numbers of dimensions on four datasets.

		8	16	32	64	128	256
JD	MF_FEDAVG	0.0329	0.0657	0.1315	0.2630	0.5260	1.0520
	FedGNN	0.0329	0.0657	0.1315	0.2630	0.5260	1.0520
	DFMR (w/ cache)	0.0138	0.0525	0.2047	0.8076	3.2083	12.789
	DFMR (w/o cache)	0.0166	0.0331	0.0663	0.1325	0.2650	0.5301
Tmall	MF_FEDAVG	0.0395	0.0790	0.1580	0.3160	0.6319	1.2638
	FedGNN	0.0395	0.0790	0.1580	0.3160	0.6319	1.2638
	DFMR (w/ cache)	0.0288	0.1092	0.4245	1.6734	6.6445	26.480
	DFMR (w/o cache)	0.0199	0.0398	0.0796	0.1592	0.3184	0.6369
UB	MF_FEDAVG	0.0756	0.1511	0.3022	0.6044	1.2089	2.4177
	FedGNN	0.0756	0.1511	0.3022	0.6044	1.2089	2.4177
	DFMR (w/ cache)	0.0171	0.0646	0.2510	0.9889	3.9255	15.642
	DFMR (w/o cache)	0.0381	0.0761	0.1523	0.3046	0.6092	1.2183
ML10M	MF_FEDAVG	0.0089	0.0177	0.0355	0.0710	0.1419	0.2838
	FedGNN	0.0089	0.0177	0.0355	0.0710	0.1419	0.2838
	DFMR (w/ cache)	0.1658	0.6267	2.4339	9.5902	38.070	151.70
	DFMR (w/o cache)	0.0045	0.0089	0.0179	0.0358	0.0715	0.1430

Superiority of the Cache Updating Module (RQ2)

We have the following observations:

- We can see that the communication cost of our DFMR with cache is higher than that without cache, which indicates that the use of the cache updating module will increase the communication overhead.
- As the dimension number increases, the communication cost of our DFMR without cache increases linearly while that with cache increases sharply.

Superiority of the Cache Updating Module (RQ2)

- For our DFMR with cache, its communication cost is higher than MF_FedAVG and FedGNN when the size of the dimension is large. The reason is that the clients in the latter need to spend some additional cost to receive the cache which is proportional to the square of the dimension number.
- The dimension number is often fixed and it is thus unnecessary to set an especially large value. For example, we can learn a good recommendation model when $d = 32$ in the experiments. In this case, the communication costs sacrificed for the cache are acceptable.

Comparing of the Space Overhead (RQ3)

Table: Space overhead (MB) of one round with different numbers of dimensions on JD.

		8	16	32	64	128	256
Server	MF	0.8218	1.6437	3.2874	6.5747	13.149	26.299
	eALS	0.8223	1.6456	3.2952	6.6060	13.274	26.800
	VALS	0.8240	1.6520	3.3196	6.7017	13.653	28.307
	FedGNN	0.8218	1.6437	3.2874	6.5747	13.149	26.299
	DCF	0.0262	0.0534	0.1108	0.2372	0.5369	1.3238
	DFMR (w/ cache)	0.0283	0.0615	0.1425	0.3632	1.0388	3.3277
	DFMR (w/o cache)	0.0257	0.0514	0.1027	0.2055	0.4109	0.8218
Client	MF	0.8219	1.6438	3.2876	6.5752	13.150	26.301
	eALS	0.8224	1.6458	3.2954	6.6064	13.275	26.801
	VALS	0.8230	1.6479	3.3035	6.6382	13.401	27.303
	FedGNN	0.8219	1.6438	3.2876	6.5752	13.150	26.301
	DCF	0.0134	0.0278	0.0594	0.1345	0.3315	0.9129
	DFMR (w/ cache)	0.0147	0.0333	0.0824	0.2273	0.7045	2.4090
	DFMR (w/o cache)	0.0128	0.0257	0.0514	0.1027	0.2055	0.4110

Comparing of the Space Overhead (RQ3)

Table: Space overhead (MB) of one round with different numbers of dimensions on Tmall.

		8	16	32	64	128	256
Server	MF	0.9874	1.9747	3.9495	7.8989	15.798	31.596
	eALS	0.9879	1.9767	3.9573	7.9302	15.923	32.096
	VALS	0.9896	1.9830	3.9817	8.0259	16.302	33.604
	FedGNN	0.9874	1.9747	3.9495	7.8989	15.798	31.596
	DCF	0.0314	0.0638	0.1315	0.2786	0.6197	1.4893
	DFMR (w/ cache)	0.0335	0.0718	0.1632	0.4046	1.1216	3.4932
	DFMR (w/o cache)	0.0309	0.0617	0.1234	0.2468	0.4937	0.9874
Client	MF	0.9874	1.9749	3.9497	7.8994	15.799	31.598
	eALS	0.9879	1.9768	3.9575	7.9307	15.924	32.098
	VALS	0.9885	1.9789	3.9656	7.9624	16.050	32.600
	FedGNN	0.9874	1.9749	3.9497	7.8994	15.799	31.598
	DCF	0.0160	0.0329	0.0698	0.1552	0.3728	0.9957
	DFMR (w/ cache)	0.0173	0.0385	0.0927	0.2479	0.7459	2.4918
	DFMR (w/o cache)	0.0154	0.0309	0.0617	0.1234	0.2469	0.4937

Comparing of the Space Overhead (RQ3)

Table: Space overhead (MB) of one round with different numbers of dimensions on UB.

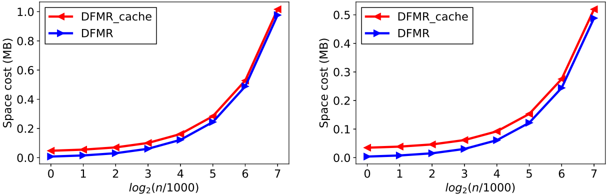
		8	16	32	64	128	256
Server	MF	1.8889	3.7777	7.5554	15.111	30.222	60.443
	eALS	1.8893	3.7797	7.5632	15.142	30.347	60.943
	VALS	1.8911	3.7860	7.5876	15.238	30.726	62.451
	FedGNN	1.8889	3.7777	7.5554	15.111	30.222	60.443
	DCF	0.0596	0.1201	0.2442	0.5040	1.0704	2.3908
	DFMR (w/ cache)	0.0617	0.1282	0.2759	0.6299	1.5724	4.3947
	DFMR (w/o cache)	0.0590	0.1181	0.2362	0.4722	0.9444	1.8889
Client	MF	1.8899	3.7778	7.5557	15.111	30.223	60.445
	eALS	1.8894	3.7798	7.5635	15.143	30.348	60.945
	VALS	1.8900	3.7819	7.5716	15.174	30.474	62.447
	FedGNN	1.8899	3.7778	7.5557	15.111	30.223	60.445
	DCF	0.0301	0.0611	0.1261	0.2679	0.5982	1.4464
	DFMR (w/ cache)	0.0314	0.0667	0.1491	0.3606	0.9713	2.9425
	DFMR (w/o cache)	0.0295	0.0590	0.1181	0.2361	0.4722	0.9445

Comparing of the Space Overhead (RQ3)

Table: Space overhead (MB) of one round with different numbers of dimensions on ML10M.

		8	16	32	64	128	256
Server	MF	0.2217	0.4435	0.8870	1.7739	3.5479	7.0957
	eALS	0.2222	0.4454	0.8948	1.8052	3.6729	7.5957
	VALS	0.2239	0.4518	0.9192	1.9009	4.0518	9.1035
	FedGNN	0.2217	0.4435	0.8870	1.7739	3.5479	7.0957
	DCF	0.0075	0.0159	0.0358	0.0872	0.2368	0.7237
	DFMR (w/ cache)	0.0096	0.0240	0.0675	0.2132	0.7388	2.7276
	DFMR (w/o cache)	0.0069	0.0139	0.0277	0.0554	0.1109	0.2217
Client	MF	0.2218	0.4436	0.8872	1.7744	3.5488	7.0977
	eALS	0.2223	0.4456	0.8950	1.8057	3.6738	7.5977
	VALS	0.2229	0.4477	0.9031	1.8374	3.7998	8.0996
	FedGNN	0.2218	0.4436	0.8872	1.7744	3.5488	7.0977
	DCF	0.0040	0.0090	0.0219	0.0595	0.1814	0.6129
	DFMR (w/ cache)	0.0054	0.0146	0.0449	0.1522	0.5545	2.1090
	DFMR (w/o cache)	0.0035	0.0069	0.0139	0.0277	0.0555	0.1109

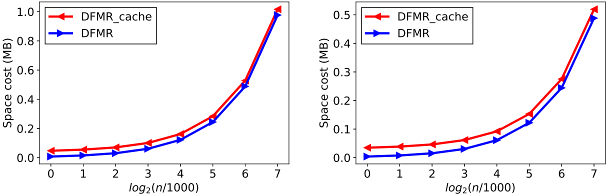
Comparing of the Space Overhead (RQ3)



(a) Server (b) Client

Figure: Space overhead (MB) of the server and each client of one round with different numbers of items on JD.

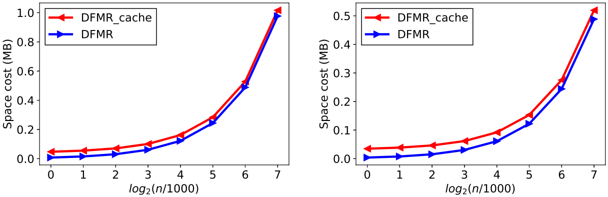
Comparing of the Space Overhead (RQ3)



(a) Server (b) Client

Figure: Space overhead (MB) of the server and each client of one round with different numbers of items on Tmall.

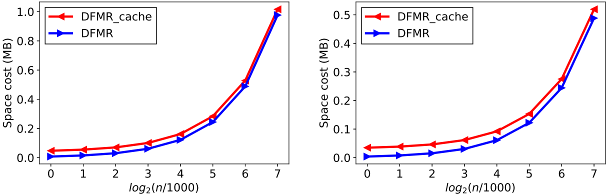
Comparing of the Space Overhead (RQ3)



(a) Server (b) Client

Figure: Space overhead (MB) of the server and each client of one round with different numbers of items on UB.

Comparing of the Space Overhead (RQ3)



(a) Server (b) Client

Figure: Space overhead (MB) of the server and each client of one round with different numbers of items on ML10M.

Comparing of the Space Overhead (RQ3)

We have the following observations:

- Taking MF and DFMR (w/o cache) as an example, we can see that the discrete hashing techniques can compress the storage space to 1.56% of the real-valued parameters when $B = 64$.
- Similar to the experimental results for the communication cost, the cache will bring some additional storage overhead.
- Both lines gradually approach as the number of items increases. Hence, the additional storage overhead can be ignored when the number of items is large.

Conclusions

- To ensure the basic business imperatives, we assume that the purchase behaviors can be collected, and then propose a novel framework called discrete federated multi-behavior recommendation (DFMR).
- We use discrete hashing techniques to binarize the user and item vectors, which reduces the storage overhead significantly.
- We design a global model updating module and a personal model updating module to update the parameters.
- We design a cache updating module to break through the computational bottlenecks.

Future Work

- We are interested in studying how to improve the accuracy of the discrete hashing methods in federated recommendation.
- We are also interested in extending our method to privacy-preserving sequential recommendation with different types of behaviors.
- We are also interested in studying how to forget the private data from the recommendation systems, which lies in the fields of machine unlearning and federated unlearning.

Thank you!

- We thank the handling editors and reviewers for their effort and constructive expert comments, and the support of National Natural Science Foundation of China (Nos. 62172283 and 62272315), Guangdong Basic and Applied Basic Research Foundation (Grant No. 2024A1515010122) and National Key Research and Development Program of China (Grant No. 2023YFF0725100).
- We thank Ms. Qianzhen Rao and Mr. Yang Liu for their data preprocessing scripts and Dr. Dugang Liu for his helpful discussions.

Ding, J., Yu, G., He, X., Quan, Y., Li, Y., Chua, T., Jin, D., and Yu, J. (2018).
Improving implicit recommender systems with view data.
In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3343–3349.

Farsa, D. Z. and Rahnamayan, S. (2020).
Discrete coordinate descent (DCD).
In *Proceedings of the 2020 IEEE International Conference on Systems, Man, and Cybernetics*, pages 184–190.

Han, J., Ma, Y., Mei, Q., and Liu, X. (2021).
DeepRec: On-device deep learning for privacy-preserving sequential recommendation in mobile commerce.
In *Proceedings of the Web Conference 2021*, pages 900–911.

He, X., Zhang, H., Kan, M., and Chua, T. (2016).
Fast matrix factorization for online recommendation with implicit feedback.
In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 549–558. ACM.

Johnson and C, C. (2014).
Logistic matrix factorization for implicit feedback data.
Advances in Neural Information Processing Systems, 27(78):1–9.

Lin, Z., Pan, W., Yang, Q., and Ming, Z. (2022).
A generic federated recommendation framework via fake marks and secret sharing.
ACM Transactions on Information Systems, 41(2):1–37.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017).
Communication-efficient learning of deep networks from decentralized data.
In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pages 1273–1282.

Selek, I., Vasara, J., and Ikonen, E. (2022).
Generalized orthogonalization: A unified framework for gram-schmidt orthogonalization, SVD and PCA.
In *Proceedings of the 2022 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1754–1759.

Wu, C., Wu, F., Lyu, L., Qi, T., Huang, Y., and Xie, X. (2022).
A federated graph neural network framework for privacy-preserving personalization.
Nature Communications, 13(1):3091.

Zhang, H., Luo, F., Wu, J., He, X., and Li, Y. (2022).
LightFR: Lightweight federated recommendation with privacy-preserving matrix factorization.
ACM Transactions on Information Systems.

Zhang, H., Shen, F., Liu, W., He, X., Luan, H., and Chua, T. (2016).
Discrete collaborative filtering.
In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 325–334.