北京交通大学

基于元矩阵分解的联邦评分预测任务

From Personalization to Privatization: Meta Matrix Factorization for Federated Rating Predictions

Honglei Zhang

2020-12-13

Yujie et al. Meta Matrix Factorization for Federated Rating Predictions, SIGIR, 2020.

Outline

北京交通大學

1. Motivation

2. Related Work

3. Model Architecture

4. Experimental Analysis

5. Discussions

Motivation

- Traditionally, recommender systems are organized in a **centralized fashion**, i.e., service providers hold all data and models at a data center.
- As even an anonymized centralized dataset still puts **user privacy at risk** via combinations with other datasets, **federated recommender systems** are increasingly being considered so to realize privacy-aware recommendations.
- Unlike fully centralized recommender systems at a data center, federated recommender systems that need to run on local devices have **stricter requirements** on the scale of the model, more space for storage, more RAM for running programs, more energy for calculation, and more communication bandwidth for downloading or updating.
- Moreover, existing federated approaches cannot **effectively exploit collaborative filtering** (CF) information among users/device, which limits the performance of existing federated recommendation methods.

Contributions

- We introduce a novel federated matrix factorization (MF) framework, MetaMF, that can **reduce the parameters of RP models and item embeddings** without loss in performance.
- We devise a **meta network**, including **collaborative memory and meta recommender modules**, to better exploit collaborative filtering in federated recommender systems.
- We propose a **rise-dimensional generation strategy** to reduce the parameters and calculation in generation.
- We conduct extensive experiments and analyses to **verify the effectiveness and efficiency** of MetaMF.

Related Work-Matrix Factorization

	Item1	Item2	Item3	…	Item n
User 1	5	?	4	…	?
User 2	?	1	?	…	3
User 3	?	?	4	…	5
…	…	…	…	…	…
User m	1	?	5	…	?

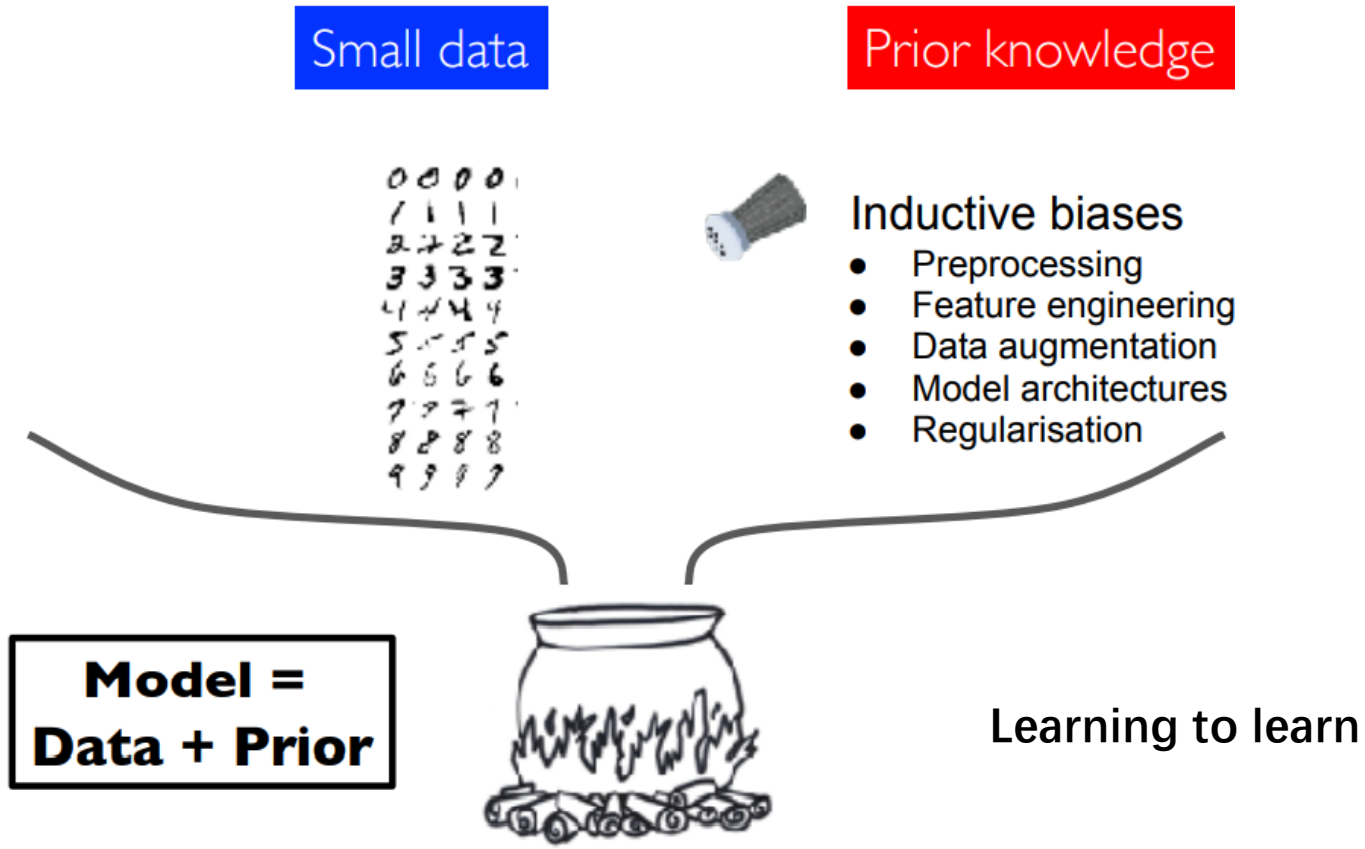
- Given $\mathbf{R} \in \mathbb{R}^{m \times n}$,
- We aim to approximate UI matrix by
- $\widehat{\mathbf{R}} = \mathbf{P}^{\mathbf{T}}\mathbf{Q}$
- where $\mathbf{P} \in \mathbb{R}^{f \times m}$ & $\mathbf{Q} \in \mathbb{R}^{f \times n}$

- $\min_{\mathbf{P},\mathbf{Q}} \frac{1}{2}\left\|\mathbf{R} - \mathbf{P}^{\mathbf{T}}\mathbf{Q}\right\|_F^2$
- $\mathbf{P}$ & $\mathbf{Q}$ can be explained as the latent factor matrices of users and items
- Both *Users* & *items* are placed in a common subspace

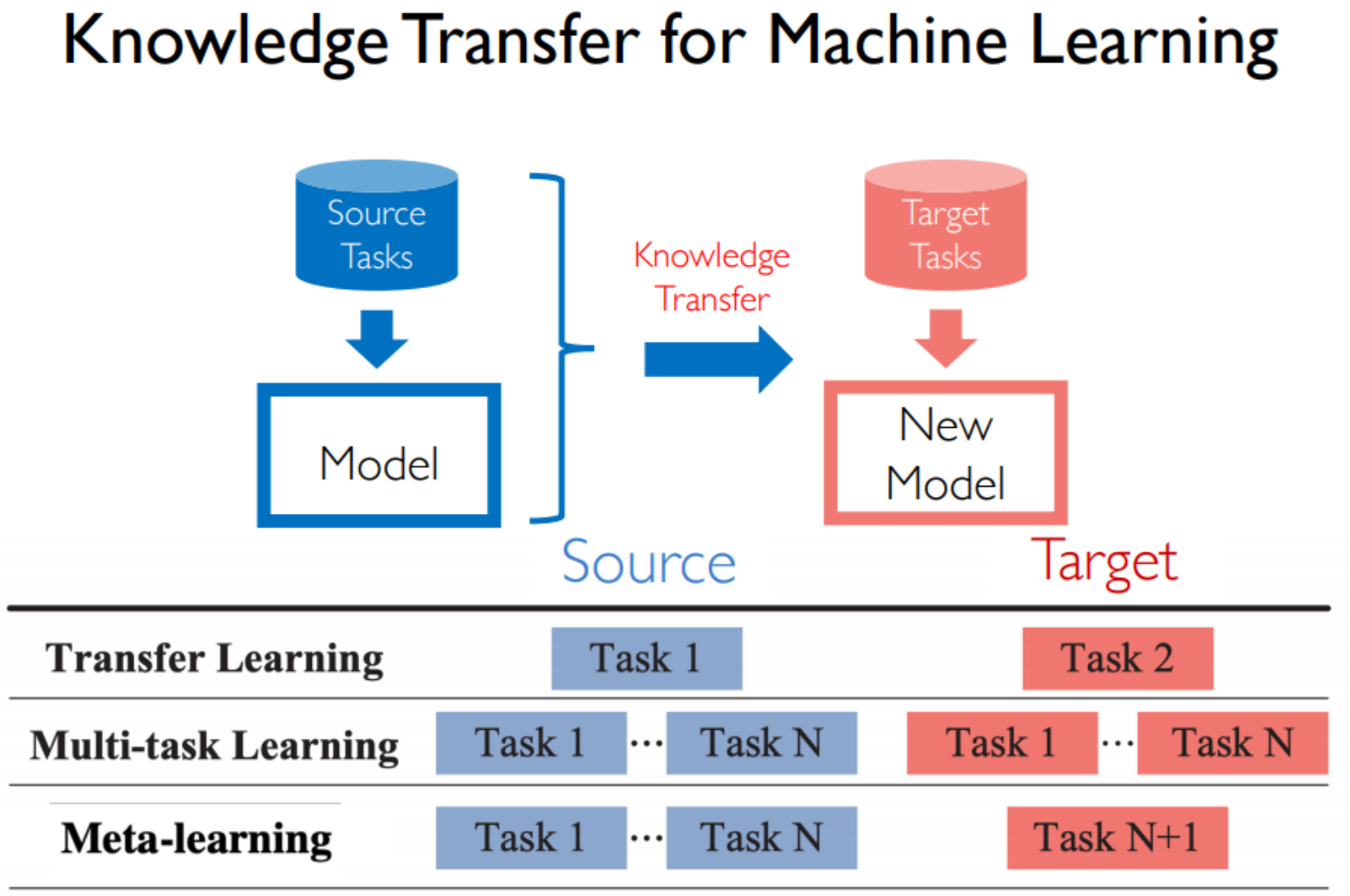
邬俊. 推荐系统导论. 北京交通大学. 2020

推荐系统之矩阵分解家族-https://zhuanlan.zhihu.com/p/35262187

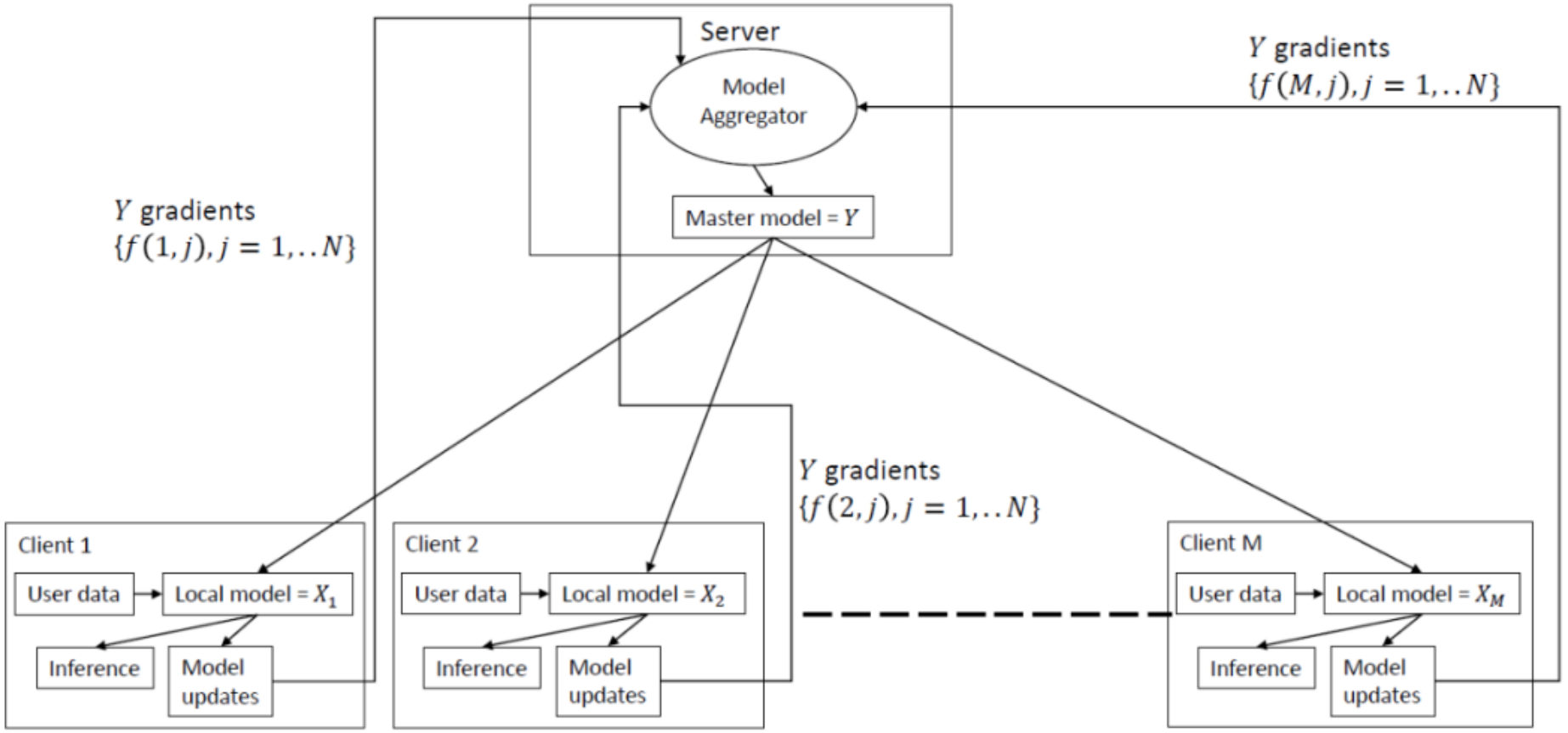
Related Work-Learning from Small data



Related Work-Learning from Small data

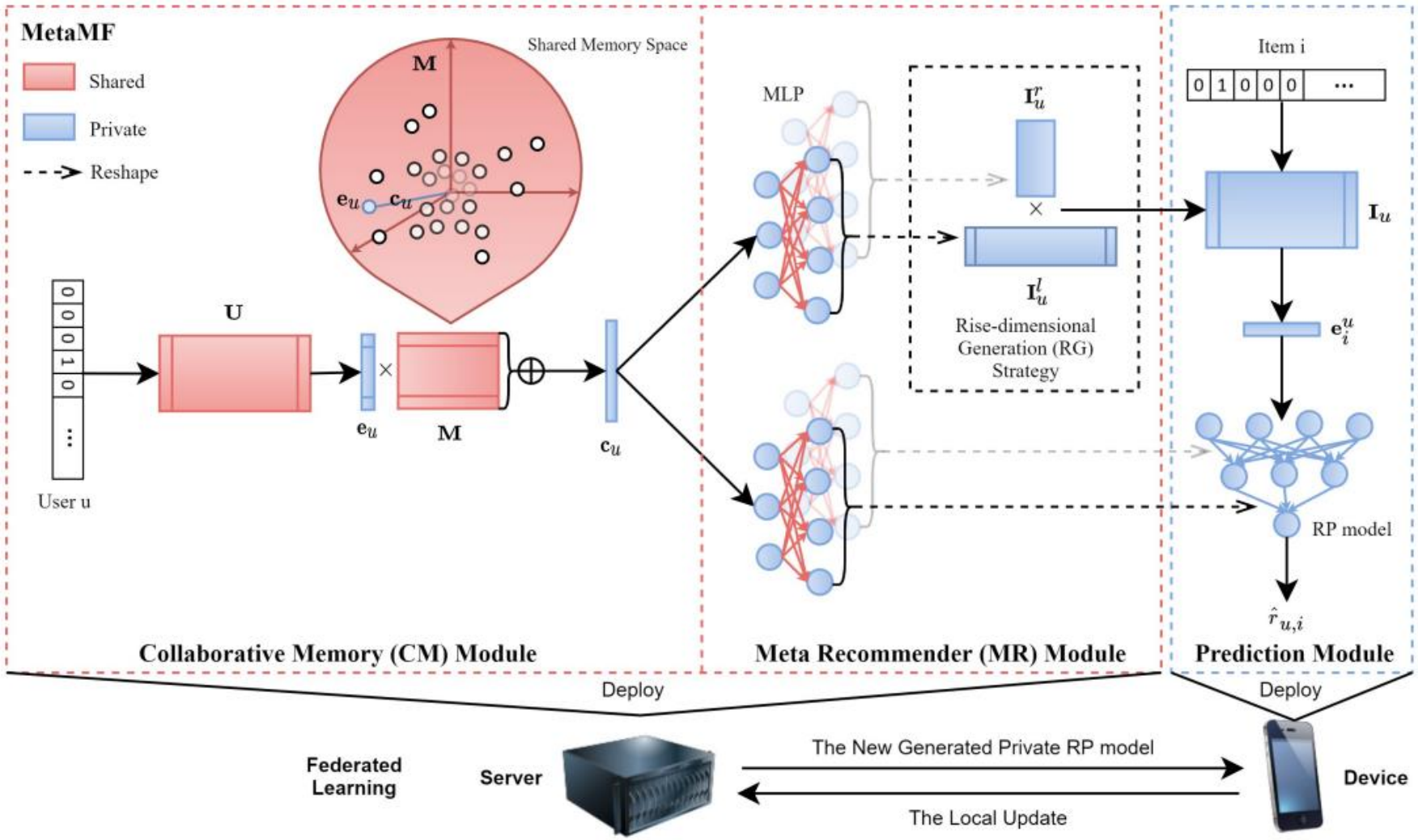


Related Work-Federated Learning



Muhammad et al. Federated Collaborative Filtering for Privacy-Preserving Personalized Recommendation System, arXiv, 2019.

Model Architecture - MetaMF



Model Architecture - MetaMF

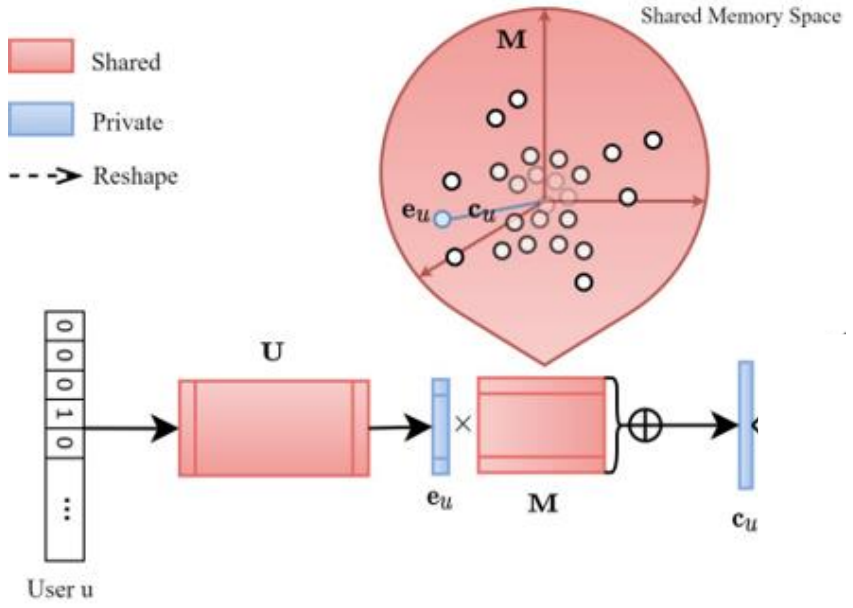
协同记忆模块CMM

Specifically, we assign each user u and each item i the indicator vectors, $\mathbf{i}_u \in \mathbb{R}^m$ and $\mathbf{i}_i \in \mathbb{R}^n$ respectively, where m is the number of users and n is the number of items. Note that $\mathbf{i}_u$ and $\mathbf{i}_i$ are one-hot vectors with each dimension corresponding to a particular user or item. For the given user u, we first get the user embedding $\mathbf{e}_u$ by Eq. 1:

$$\mathbf{e}_u = \mathbf{U}\mathbf{i}_u, \tag{1}$$

where $\mathbf{e}_u \in \mathbb{R}^{d_u}$, $\mathbf{U} \in \mathbb{R}^{d_u \times m}$ is the user embedding matrix, and d_u is the size of user embeddings. Then we proceed to obtain a collaborative vector for u. Specifically, we use a shared memory matrix $\mathbf{M} \in \mathbb{R}^{d_u \times k}$ to store the basis vectors which span a space of all collaborative vectors, where k is the dimension of basis vectors and collaborative vectors. And we consider the user embedding $\mathbf{e}_u$ as the coordinates of u in the shared memory space. So the collaborative vector $\mathbf{c}_u \in \mathbb{R}^k$ for u is a linear combination of the basis vectors in $\mathbf{M}$ by $\mathbf{e}_u$, as shown in Eq. 2:

$$\mathbf{c}_u = \sum_i \mathbf{M}(i,:)\mathbf{e}_u(i), \tag{2}$$



Model Architecture - MetaMF

元推荐模型 Meta Recommender Module

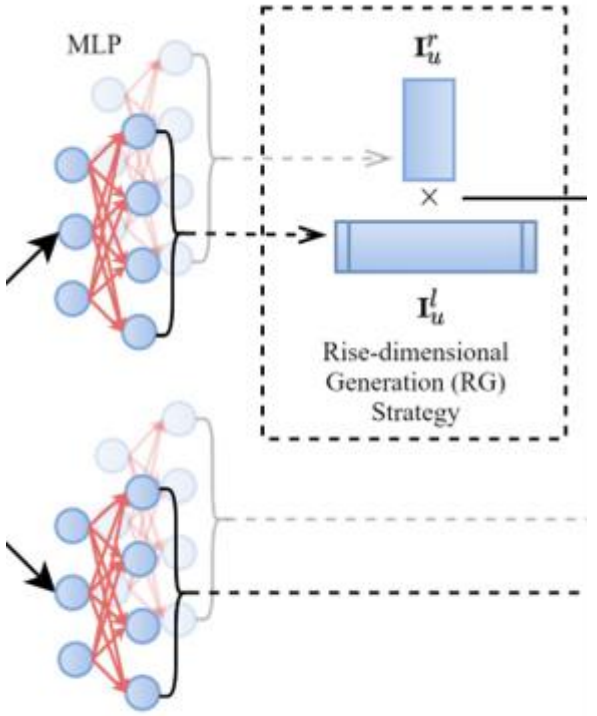
3.4.1 Private Item Embeddings. We propose to generate the private item embedding matrix $\mathbf{I}_u \in \mathbb{R}^{d_i \times n}$ for each user u, where d_i is the size of item embeddings. However, it is a challenge to directly generate the whole item embedding matrix when there are a large number of items with relatively high-dimensional item embeddings (instead of extremely small ones). Therefore, we propose a rise-dimensional generation (RG) strategy to decompose the generation into two parts: a low-dimensional item embedding matrix $\mathbf{I}_u^l \in \mathbb{R}^{s \times n}$ and a rise-dimensional matrix $\mathbf{I}_u^r \in \mathbb{R}^{d_i \times s}$, where s is the size of low-dimensional item embeddings and $s \ll d_i$. Specifically, we first follow Eq. 3 to generate $\mathbf{I}_u^l \in \mathbb{R}^{sn}$ and $\mathbf{I}_u^r \in \mathbb{R}^{d_i s}$ (in the form of vectors):

$$\begin{aligned} \mathbf{h}_i^l &= \mathrm{ReLU}(\mathbf{W}_i^l \mathbf{c}_u + \mathbf{b}_i^l), \quad \mathbf{I}_u^l = \mathbf{U}_i^l \mathbf{h}_i^l, \\ \mathbf{h}_i^r &= \mathrm{ReLU}(\mathbf{W}_i^r \mathbf{c}_u + \mathbf{b}_i^r), \quad \mathbf{I}_u^r = \mathbf{U}_i^r \mathbf{h}_i^r, \end{aligned} \tag{3}$$

where $\mathbf{W}_i^l$ and $\mathbf{W}_i^r \in \mathbb{R}^{o \times k}$, $\mathbf{U}_i^l \in \mathbb{R}^{sn \times o}$ and $\mathbf{U}_i^r \in \mathbb{R}^{d_i s \times o}$ are weights; $\mathbf{b}_i^l$ and $\mathbf{b}_i^r \in \mathbb{R}^o$ are biases; $\mathbf{h}_i^l$ and $\mathbf{h}_i^r \in \mathbb{R}^o$ are hidden states; o is the hidden size. Then we reshape $\mathbf{I}_u^l$ to a matrix whose shape is $s \times n$, and reshape $\mathbf{I}_u^r$ to a matrix whose shape is $d_i \times s$. Finally, we multiply $\mathbf{I}_u^l$ and $\mathbf{I}_u^r$ to get $\mathbf{I}_u$:

$$\mathbf{I}_u = \mathbf{I}_u^r \mathbf{I}_u^l. \tag{4}$$

Compared to directly generating $\mathbf{I}_u$, which needs $O(d_i \times n)$ parameters, the RG strategy needs $O(s \times n + d_i \times s)$ parameters which reduces the cost of generating $\mathbf{I}_u$. For different users, the generated item embedding matrices are different.



Meta Recommender (MR) Module

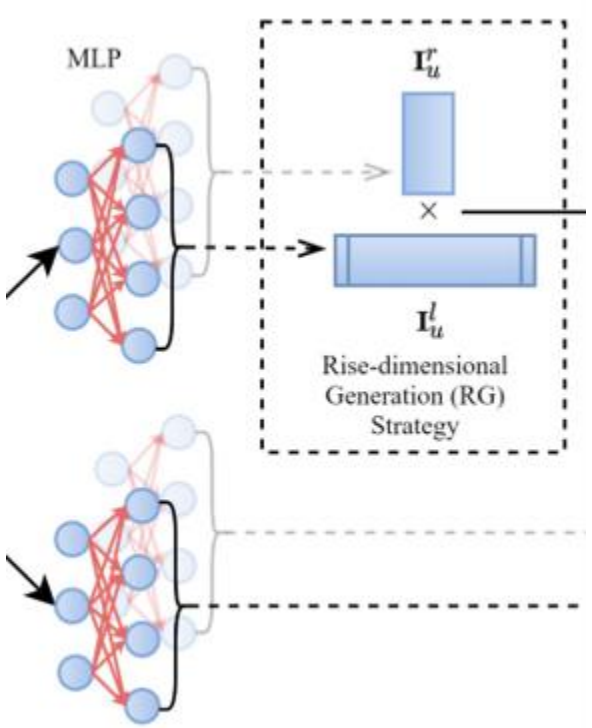
Model Architecture - MetaMF

元推荐模型 Meta Recommender Module

3.4.2 Private RP Model. We also propose to generate a private RP model for each user u. We use a MLP as the RP model, so we need to generate the weights and biases for each layer of MLP. Specifically, for layer l, we denote its weights and biases as $\mathbf{W}_l^u \in \mathbb{R}^{f_{out} \times f_{in}}$ and $\mathbf{b}_l^u \in \mathbb{R}^{f_{out}}$ respectively, where f_{in} is the size of its input and f_{out} is the size of its output. Then $\mathbf{W}_l^u$ and $\mathbf{b}_l^u$ are calculated as follows:

$$\begin{aligned} \mathbf{h}_g &= \mathrm{ReLU}(\mathbf{W}_g^h \mathbf{c}_u + \mathbf{b}_g^h), \\ \mathbf{W}_l^u &= \mathbf{U}_g^w \mathbf{h}_g + \mathbf{b}_g^w, \\ \mathbf{b}_l^u &= \mathbf{U}_g^b \mathbf{h}_g + \mathbf{b}_g^b, \end{aligned} \tag{5}$$

where $\mathbf{W}_g^h \in \mathbb{R}^{o \times k}$, $\mathbf{U}_g^w \in \mathbb{R}^{f_{out} f_{in} \times o}$ and $\mathbf{U}_g^b \in \mathbb{R}^{f_{out} \times o}$ are weights; $\mathbf{b}_g^h \in \mathbb{R}^o$, $\mathbf{b}_g^w \in \mathbb{R}^{f_{out} f_{in}}$ and $\mathbf{b}_g^b \in \mathbb{R}^{f_{out}}$ are biases; $\mathbf{h}_g \in \mathbb{R}^o$ is hidden state. Finally, we reshape $\mathbf{W}_l^u$ to a matrix whose shape is $f_{out} \times f_{in}$. Note that $\mathbf{W}_g^h$, $\mathbf{b}_g^h$, $\mathbf{U}_g^w$, $\mathbf{b}_g^w$, $\mathbf{U}_g^b$ and $\mathbf{b}_g^b$ are not shared by different layers of the RP model. And f_{in} and f_{out} also vary with different layers. Detailed settings can be found in the experimental setup. Also, MetaMF returns different parameters of the MLP to each user.



Meta Recommender (MR) Module

Model Architecture - MetaMF

预测模块 Prediction Module

The prediction module estimates the user's rating for a given item i using the generated item embedding matrix $\mathbf{I}_u$ and RP model from the CM module.

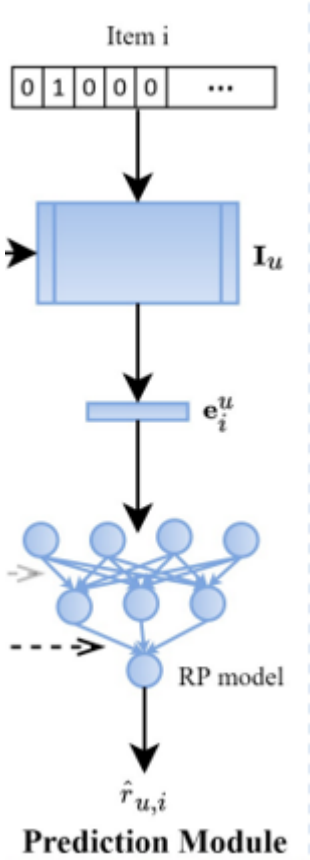
First, we get the private item embedding $\mathbf{e}_i^u \in \mathbb{R}^{d_i}$ of i from $\mathbf{I}_u$ by Eq. 6:

$$\mathbf{e}_i^u = \mathbf{I}_u \mathbf{i}_i. \tag{6}$$

Then we follow Eq. 7 to predict $r_{u,i}$ based on the RP model:

$$\begin{aligned} \mathbf{h}_1 &= \text{ReLU}(\mathbf{W}_1^u \mathbf{e}_i^u + \mathbf{b}_1^u), \\ \mathbf{h}_2 &= \text{ReLU}(\mathbf{W}_2^u \mathbf{h}_1 + \mathbf{b}_2^u), \\ &\vdots \\ \mathbf{h}_{L-1} &= \text{ReLU}(\mathbf{W}_{L-1}^u \mathbf{h}_{L-2} + \mathbf{b}_{L-1}^u), \\ \hat{r}_{u,i} &= \mathbf{W}_L^u \mathbf{h}_{L-1} + \mathbf{b}_L^u, \end{aligned} \tag{7}$$

where L is the number of layers of the RP model. The weights $\{\mathbf{W}_1^u, \mathbf{W}_2^u, \ldots, \mathbf{W}_{L-1}^u, \mathbf{W}_L^u\}$ and biases $\{\mathbf{b}_1^u, \mathbf{b}_2^u, \ldots, \mathbf{b}_{L-1}^u, \mathbf{b}_L^u\}$ are generated by the CM module. The last layer L is the output layer, which returns a scalar as the predicted rating $\hat{r}_{u,i}$.



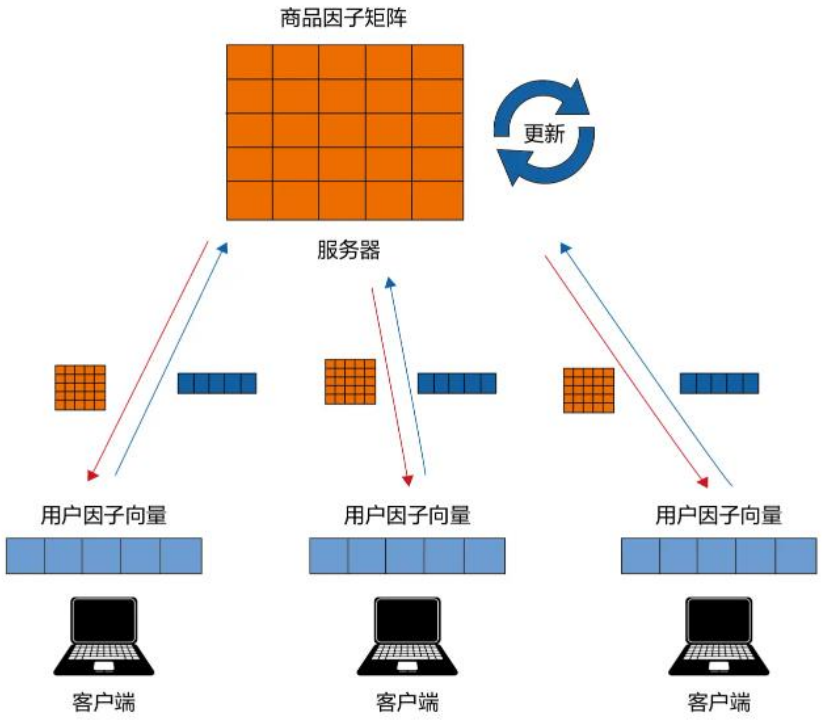
Model Architecture - MetaMF

Algorithm 1 MetaMF

Input: All trainable parameters Θ, which are stored in the server; The user set $\mathcal{U}$, where one user per device; For user u, her local data D_u stored in her device, and $D_{train} = \sum_{u \in \mathcal{U}} D_u$; † means the code is executed in the device.

Output: Θ; For user u, the parameters of her RP model and item embeddings Φ_u, which are stored in her device;

Initialize Θ randomly in the server;
for u in $\mathcal{U}$ **do**
    Generate Φ_u based on Θ;
    Send Φ_u to u's device;
end for
while not convergent **do**
    Sample a batch S from $\mathcal{U}$;
    for u in S **do**
        Sample a batch B_u from D_u;†
        Calculate the gradient of Φ_u based on B_u;†
        Upload the gradient to the server;†
        Calculate the gradient of Θ based on the gradient of Φ_u;
    end for
    Accumulate the gradients of Θ gathered from S;
    Update Θ based on the accumulated gradient;
    for u in $\mathcal{U}$ **do**
        Regenerate Φ_u based on new Θ;
        Send Φ_u to u's device;
    end for
end while



Experimental Setup

- **Conventional methods:**
  - **NMF** [54]: uses non-negative matrix factorization to decompose rating matrices.
  - **PMF** [34]: applies Gaussian distributions to model the latent factors of users and items.
  - **SVD++** [24]: extends SVD by considering implicit feedback for modeling latent factors.
  - **LLORMA** [28]: uses a number of low-rank sub-matrices to compose rating matrices.
- **Deep learning-based methods:**
  - **RBM** [38]: employs restricted Boltzmann machine (RBM) to model the generation process of ratings.
  - **AutoRec** [40]: proposes autoencoders (AEs) to model interactions between users and items. AutoRec has two variants, with one taking users' ratings as input, denoted by AutoRec-U, and the other taking items' ratings as input, denoted by AutoRec-I.
  - **NCF** [18]: the state-of-the-art MF method that combines generalized matrix factorization and MLP to model user-item interactions. We adapt NCF for the RP task by dropping the sigmoid activation function on its output layer and replacing its loss function with Eq. 8.
- **Federated methods:**
  - **FedRec** [7]: a federated recommendation method, which employs MAML [11] to learn a shared RP model in the server and update the model for each device. In our experiments, the shared RP model is a MLP with two layers (layer sizes are 16 and 1 respectively), and its user/item embedding size is 64.

Table 1: Statistics of the datasets, where *#avg* means the average number of user ratings, Hetrec-ML is the short name of Hetrec-movielens.

Datasets	*#users*	*#items*	*#ratings*	*#avg*	*#sparsity (%)*
Douban	2,509	39,576	894,887	357	0.9
Hetrec-ML	2,113	10,109	855,599	405	4
Movielens1M	6,040	3,706	1,000,209	166	4.5
Ciao	7,375	105,096	282,619	38	0.04

$$\text{MAE} = \frac{1}{|D_{test}|} \sum_{r_{u,i} \in D_{test}} |r_{u,i} - \hat{r}_{u,i}|. \tag{11}$$

MSE is defined as:

$$\text{MSE} = \frac{1}{|D_{test}|} \sum_{r_{u,i} \in D_{test}} (r_{u,i} - \hat{r}_{u,i})^2. \tag{12}$$

Experimental Analysis

Table 2: Comparison results of MetaMF and baselines on the four datasets. A superscript $^{\approx}$ indicates that there is no statistically significant difference between MetaMF and NCF (two-sided paired t-test, $p < 0.01$).

Method	Douban		Hetrec-movielens		Movielens1M		Ciao	
	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
NMF	0.602	0.585	0.625	0.676	0.727	0.848	0.750	1.039
PMF	0.639	0.701	0.617	0.644	0.703	0.788	1.501	3.970
SVD++	0.593	0.570	0.579	0.590	0.671	0.740	0.738	0.963
LLORMA	0.610	0.623	0.588	0.603	0.675	0.748	1.349	3.396
RBM	1.058	1.749	1.124	1.947	1.122	2.078	1.132	2.091
AutoRec-U	0.709	0.911	0.660	0.745	0.678	0.775	1.673	5.671
AutoRec-I	0.704	0.804	0.633	0.694	0.663	**0.715**	0.792	1.038
NCF	**0.583**	**0.547**	0.572	**0.575**	0.675	0.739	**0.735**	**0.937**
FedRec	0.760	0.927	0.846	1.265	0.907	1.258	0.865	1.507
MetaMF	$0.584^{\approx}$	0.549	$\mathbf{0.571}^{\approx}$	$0.578^{\approx}$	0.687	0.760	0.774	1.043

MetaMF outperforms most baselines despite the fact that it is federated while most baselines are centralized.
MetaMF is comparable to NCF on these two datasets.
MetaMF significantly outperforms FedRec on all datasets, which can flexibly take advantage of CF among users/devices by the meta network.

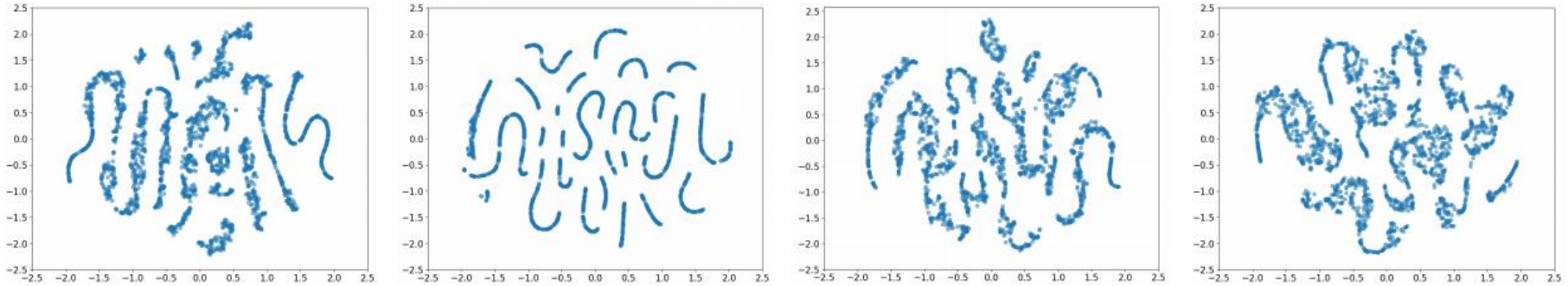
Experimental Analysis

Table 3: Rating prediction results of MetaMF, MetaMF-SI and MetaMF-SM on the four datasets. MetaMF-SI shares item embeddings for all users; MetaMF-SM shares the parameters of prediction module for all users.

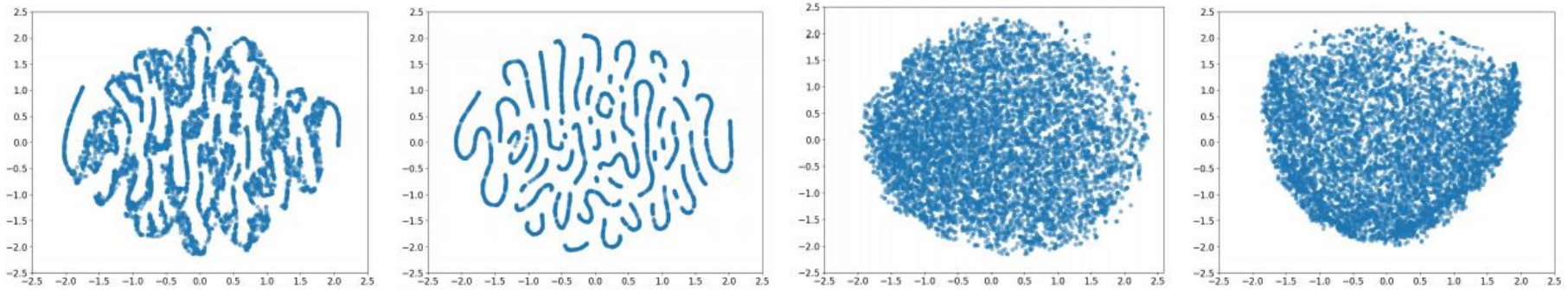
Method	Douban		Hetrec-movielens		Movielens1M		Ciao	
	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
MetaMF	**0.584**	**0.549**	**0.571**	**0.578**	**0.687**	**0.760**	0.774	1.043
MetaMF-SI	0.586	0.552	0.590	0.615	0.696	0.784	**0.732**	**0.925**
MetaMF-SM	0.595	0.571	0.595	0.622	0.697	0.788	0.789	1.061

We conclude that generating private item embeddings for each user can improve the performance of MetaMF.
Generating private RP models for users is able to improve the performance of MetaMF too.

Experimental Analysis



(a) The weights of the hidden layer on the Douban dataset.
(b) The embeddings of item 16716 on the Douban dataset.
(c) The weights of the hidden layer on the Hetrec-movielens dataset.
(d) The embeddings of item 1931 on the Hetrec-movielens dataset.



(e) The weights of the hidden layer on the Movielens1M dataset.
(f) The embeddings of item 482 on the Movielens1M dataset.
(g) The weights of the hidden layer on the Ciao dataset.
(h) The embeddings of item 8271 on the Ciao dataset.

Figure 2: The generated weights and item embeddings reduced dimension by t-SNE and normalized by mean and standard deviation on the four datasets, where one point corresponds to one user.

MetaMF **generates different weights and item embeddings** for different users on most datasets, which indicates that MetaMF has the ability to capture private factors for users.

References

- https://github.com/chaoyanghe/Awesome-Federated-Learning
- https://github.com/hongleizhang/RSPapers
- https://bitbucket.org/HeavenDog/metamf/src/master/

Questions

- 元学习在业界的应用情况?
- 联邦学习在业界的应用情况?

北京交通大學

thank you
danke
謝謝
ngiyabonga
рахмат
teşekkür ederim
Баярлалаа
спасибо
faafetai lava
mersi
kia ora
barka
welalin
tack
spas
dank je
misaotra
matondo
paldies
grazzi
mahalo
tapadh leat
gracias
vinaka
спасибі
blagodaram
kiitos
dankie
nanni
nandri
dhanyavad
mauruuru
köszönöm
bedankt
hvala
gràçie
bayarlalaa
enkosi
dziękuję
aciu
dankon
akun
хвала
asante
manana
obrigada
tenki
murakoze
chokrane
djiere dieuf
tau
mochchakkeram
дякую
mamnun
chnorakaloutioun
gratias ago
gràcies
sulpáy
go raibh maith agat
sobodi
děkuji
obrigado
sagolun
sukriya
kop khun krap
taiku
arigatô
takk
dakujem
trugarez
mèsi
didi madloba
najis tuke
ありがとう
tanemirt
grazie
shukriya
mercé
мерси
kam sah hamnida
terima kasih
rahmat
rahmet
diolch
dhanyavadagalu
তোমাকে ধন্যবাদ
감사합니다
xièxie
ευχαριστώ
merci