Modeling Interactions in Recommender Systems via Factorization Models

Presenter: Liang Hu

The $1 Million Netflix Challenge

- In October 2006, Netflix announced “The Netflix Prize” to award $1 million to improve the accuracy of its movie recommendation service.

- The Netflix Prize was an open competition for the best collaborative filtering algorithm to predict user ratings for films

- The mission: make the company's recommendation engine 10% more accurate

https://www.technologyreview.com/2006/10/06/273459/the-1-million-netflix-challenge/

Netflix Prize in 2009

- Matrix Factorization (which the community generally called SVD, Singular Value Decomposition) and Restricted Boltzmann Machines (RBM).

- MF by itself provided a 0.8914 RMSE, while RBM alone provided a competitive but slightly worse 0.8990 RMSE.

https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429

Outline

- Latent Factor Based Matrix Factorization
- Feature Based Matrix Factorization
- Tensor Factorization
- Factorization Machines

Outline

- Latent Factor Based Matrix Factorization
- Feature Based Matrix Factorization
- Tensor Factorization
- Factorization Machines

User-item rating

- A full rating matrix $\boldsymbol{Y} \in \mathbb{R}^{N \times M}$

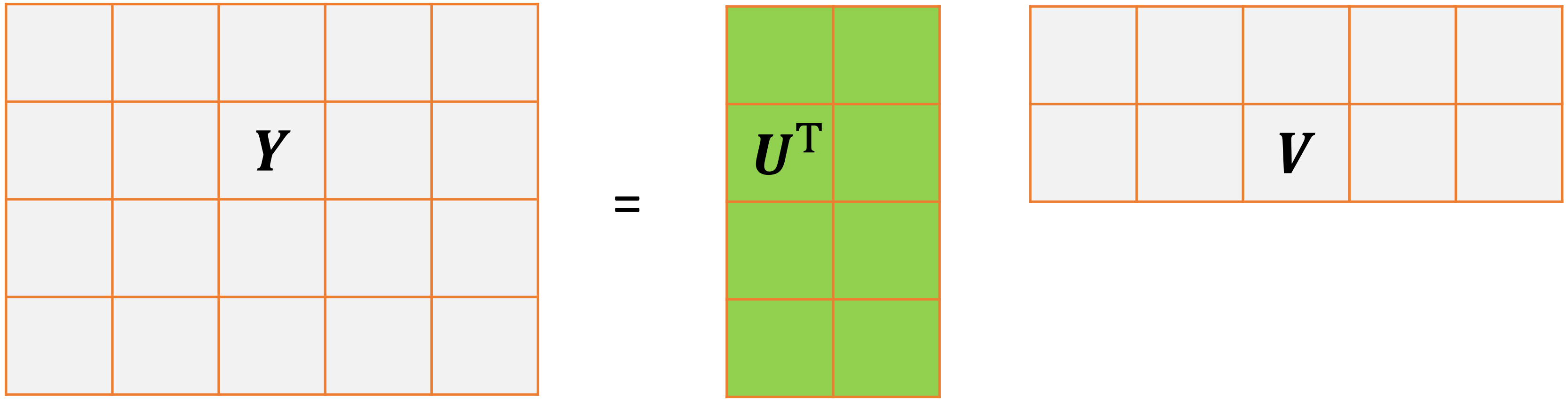
		5			4		2		
	5					5		1	
		4							
[illegible]			[illegible]						
	[illegible]			[illegible]		[illegible]			[illegible]
		4							
	2				4				
		3							5

Is there other way to represent rating table?

$O(NM)$, if N=100,000 users, M=50,000 items, each rating 4 bytes, then 20GB memory is needed.

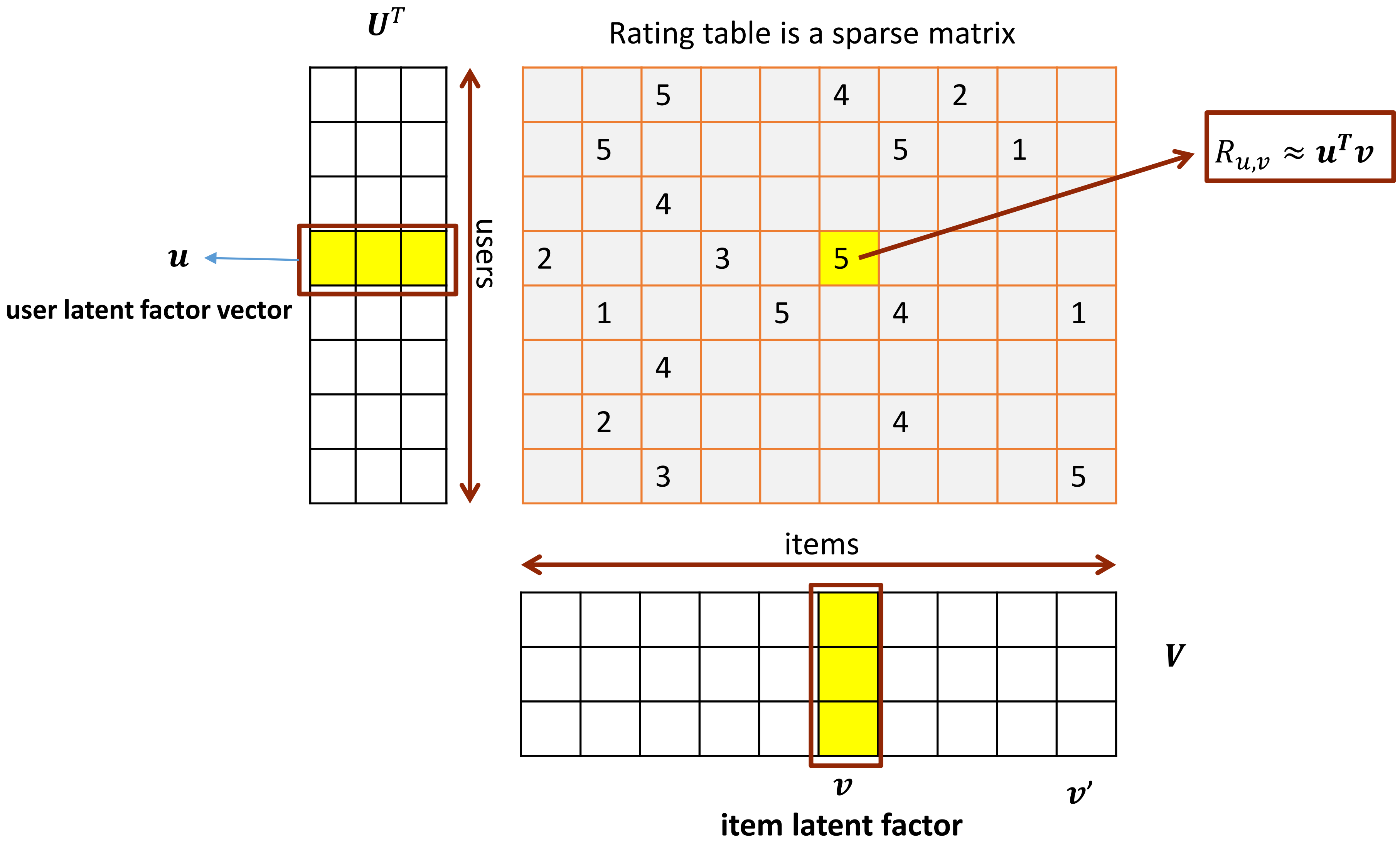
Matrix factorization: latent user/item factors

- Approximated by low-rank matrices
  - Given a matrix $\boldsymbol{Y} \in \mathbb{R}^{N \times M}$, we have
    - $\boldsymbol{Y} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{V}$ where $\boldsymbol{U} = [\boldsymbol{u}_{\boldsymbol{1}}, \dots, \boldsymbol{u}_{\boldsymbol{N}}]$, user latent factors (or user embedding in the terminology of deep learning), $\boldsymbol{V} = [\boldsymbol{v}_{\boldsymbol{1}}, \dots, \boldsymbol{v}_{\boldsymbol{M}}]$ item latent factors
    - $\boldsymbol{U} \in \mathbb{R}^{D \times N}$, $\boldsymbol{V} \in \mathbb{R}^{D \times M}$ where D denotes the dimensionality (rank)



$O(ND + MD)$, if N=100,000 users, M=50,000 items, D=100, each factor 4 bytes, only 20.1 MB memory is needed.

MF(Matrix factorization) with missing values



MF Model

- $r_{i,j}$: The rating on item j given by user i
- $\boldsymbol{u}_i = \mathbf{U}_{:,i}$: Latent factor vector of user i
- $\boldsymbol{v}_j = \mathbf{V}_{:,j}$: Latent factor vector of item j
- $e_{i,j}$: The error term
- $f(\cdot)$: Identity function

$$r_{i,j} = f\left(\boldsymbol{u}_i, \boldsymbol{v}_j\right) + e_{i,j} = \boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{v}_j + e_{i,j} = \boldsymbol{v}_j^{\mathrm{T}} \boldsymbol{u}_i + e_{i,j}$$

Alternative Least Square (ALS)

- Find the solution to one parameter with fixing all other parameters
  - E.g.

1. Find $\boldsymbol{v}_j$ by fixing $\boldsymbol{u}_i$, reducing MF to linear regression model

$$\arg\max_{\boldsymbol{v}_j}\left(r_{i,j} - \boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{v}_j\right)^2$$

we easily find optimal $\dot{\boldsymbol{v}}_j$

2. Find $\boldsymbol{u}_i$ by fixing $\dot{\boldsymbol{v}}_j$

$$\arg\max_{\boldsymbol{u}_i}\left(r_{i,j} - \dot{\boldsymbol{v}}_j^{\mathrm{T}}\boldsymbol{u}_i\right)^2$$

And we find the optimal $\dot{\boldsymbol{u}}_i$

ALS w.r.t. each user and each item

- Given user i and his/her rated item set $\boldsymbol{O}_i$, the loss function w.r.t. $\boldsymbol{u}_i$ is:

$$Loss = \frac{1}{|\boldsymbol{O}_i|}\sum_{j\in\boldsymbol{O}_i}\left(r_{ij} - \boldsymbol{v}_j^{\mathrm{T}}\boldsymbol{u}_i\right)^2 + \lambda_A\|\boldsymbol{u}_i\|^2 = \frac{1}{|\boldsymbol{O}_i|}\left(\boldsymbol{r}_i - \boldsymbol{V}_i^{\mathrm{T}}\boldsymbol{u}_i\right)^{\mathrm{T}}\left(\boldsymbol{r}_i - \boldsymbol{V}_i^{\mathrm{T}}\boldsymbol{u}_i\right) + \lambda_A\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{u}_i$$

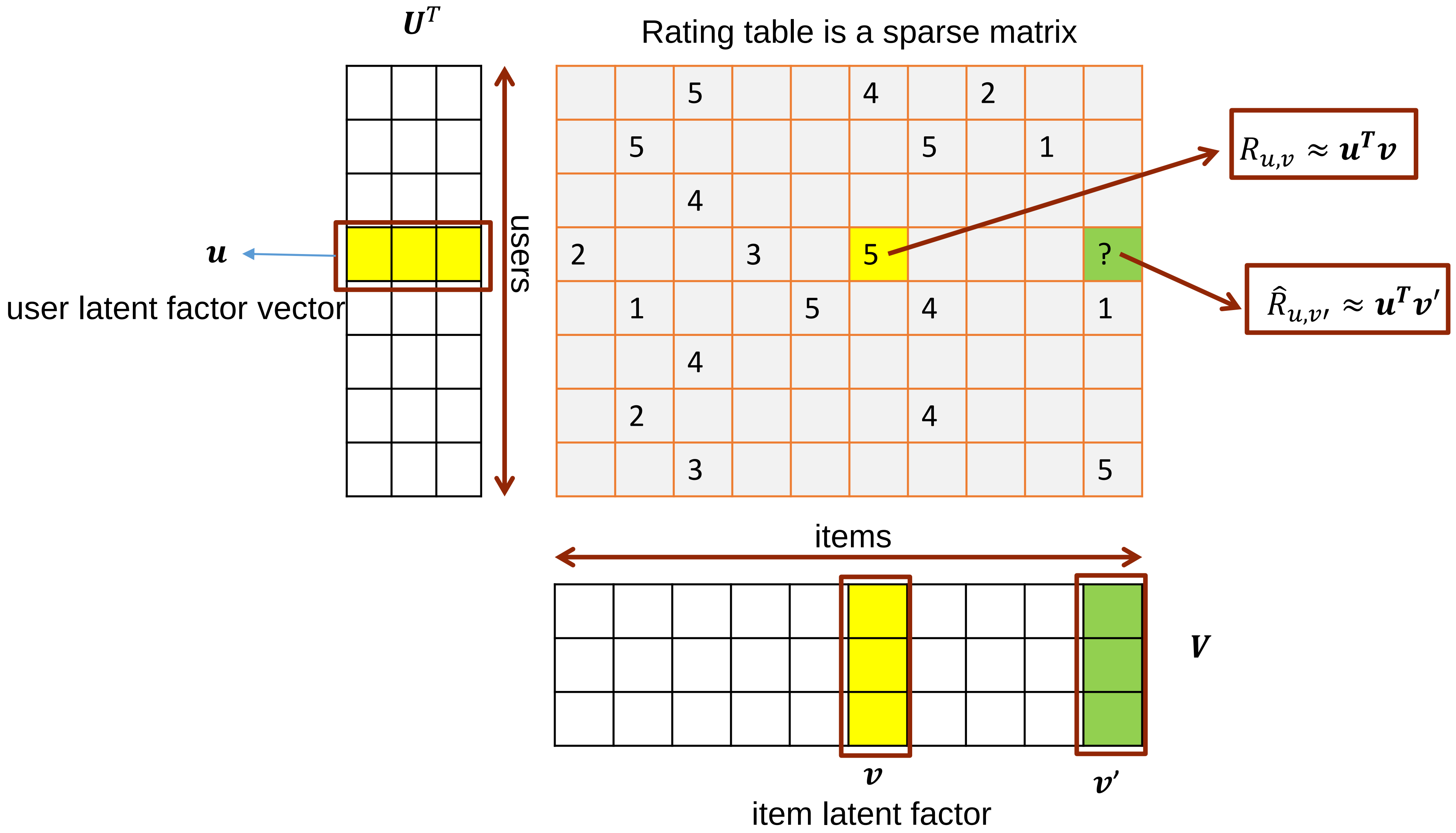
Where $\boldsymbol{r}_i = \left[r_{ij}\right]_{j\in\boldsymbol{O}_i}, \boldsymbol{V}_i = \left[\boldsymbol{v}_j\right]_{j\in\boldsymbol{O}_i}$

- Alternately, given item j and user set $\boldsymbol{O}_j$, the loss function w.r.t. $\boldsymbol{v}_j$ is:

$$Loss = \frac{1}{|\boldsymbol{O}_j|}\sum_{i\in\boldsymbol{O}_j}\left(r_{ij} - \boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{v}_j\right)^2 + \lambda_B\left\|\boldsymbol{v}_j\right\|^2 = \frac{1}{|\boldsymbol{O}_j|}\left(\boldsymbol{r}_j - \boldsymbol{U}_j^{\mathrm{T}}\boldsymbol{v}_j\right)^{\mathrm{T}}\left(\boldsymbol{r}_j - \boldsymbol{U}_j^{\mathrm{T}}\boldsymbol{v}_j\right) + \lambda_B\boldsymbol{v}_j^{\mathrm{T}}\boldsymbol{v}_j$$

Where$\boldsymbol{r}_j = \left[r_{ij}\right]_{i\in\boldsymbol{O}_j}, \boldsymbol{U}_j = \left[\boldsymbol{u}_i\right]_{i\in\boldsymbol{O}_j}$

Rating prediction



Modeling user and item biases

- Some users may tend to give higher ratings, and some other users may tend to give lower ratings.

- We can add these biases into MF model:

$$r_{i,j} = \boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{v}_j + b_i + b_j + e_{i,j}$$

where b_i models the bias of user i and b_j models the bias of item j

Probabilistic MF (PMF)

- Distribution of parameters:

$$\boldsymbol{u}_i \sim N(\mathbf{0}, \sigma_u^2 \mathbf{I}), \qquad \boldsymbol{v}_j \sim N(\mathbf{0}, \sigma_v^2 \mathbf{I})$$

- Distribution of ratings :

$$r_{i,j} = \boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{v}_j + e_{i,j}$$

$$e_{i,j} \sim \boldsymbol{N}(0, \sigma_e^2) \Leftrightarrow r_{i,j} - \boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{v}_j \sim \boldsymbol{N}(0, \sigma_e^2) \Leftrightarrow r_{i,j} \sim \boldsymbol{N}\left(\boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{v}_j, \sigma_e^2\right)$$

- The posterior of parameters and user-item interaction $\boldsymbol{O}$

$$P\left(\{\boldsymbol{u}_i\}, \{\boldsymbol{v}_j\} | \boldsymbol{O}\right) \propto P\left(\boldsymbol{O} | \{\boldsymbol{u}_i\}, \{\boldsymbol{v}_j\}\right) P(\{\boldsymbol{u}_i\}) P\left(\{\boldsymbol{v}_j\}\right)$$

$$= \prod_{u,i \in \boldsymbol{O}} N\left(r_{i,j} | \boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{v}_j, \sigma_e^2\right) \prod_i N(\boldsymbol{u}_i | \mathbf{0}, \sigma_u^2 \mathbf{I}) \prod_j N\left(\boldsymbol{v}_j | \mathbf{0}, \sigma_v^2 \mathbf{I}\right)$$

Parameter Learning for PMF

- Log-posterior function

$$\log P\left(\{\boldsymbol{u}_i\},\{\boldsymbol{v}_j\}|\boldsymbol{O}\right) \propto \log \prod_{u,i\in\boldsymbol{O}} N\left(r_{i,j}|\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{v}_j,\sigma_e^2\right)\prod_i N(\boldsymbol{u}_i|\mathbf{0},\sigma_u^2\mathbf{I})\prod_j N\left(\boldsymbol{v}_j|\mathbf{0},\sigma_v^2\mathbf{I}\right)$$

$$= -\frac{1}{2\sigma_e^2}\left[\sum_{u,i\in\boldsymbol{O}}\left(y_{u,i}-\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{v}_j\right)^2 + \lambda_u\sum_i\|\boldsymbol{u}_i\|^2 + \lambda_v\sum_i\left\|\boldsymbol{v}_j\right\|^2\right]$$

Where $\lambda_u = \frac{1}{2\sigma_u^2}, \lambda_v = \frac{1}{2\sigma_v^2}$

- So, this is equivalent to solving a L2-norm regularized least square problem.

```python
from keras.layers import Input, Embedding, dot, Lambda
from keras.models import Model
from keras.regularizers import l2
import keras.backend as K
import numpy as np
import pandas as pd


# 读入文件
df_train = pd.read_csv('ua.base', sep='\t')
df_test = pd.read_csv('ua.test', sep='\t')

user_ids = set(df_train.uid).union(set(df_test.uid))
item_ids = set(df_train.mid).union(set(df_test.mid))
nb_user = len(user_ids)
nb_item = len(item_ids)
print('Number of users = %d | Number of movies = %d' % (nb_user, nb_item))

u_id2idx = dict(zip(user_ids, range(nb_user)))
i_id2idx = dict(zip(item_ids, range(nb_item)))

# 替换ID为Index
df_train.assign(uid=[u_id2idx[uid] for uid in df_train.uid])
df_train.assign(mid=[i_id2idx[iid] for iid in df_train.mid])
df_test.assign(uid=[u_id2idx[uid] for uid in df_test.uid])
df_test.assign(mid=[i_id2idx[iid] for iid in df_test.mid])

# 构建MF模型
reg = l2(1e-5)
nb_latent_factor = 100

user_idx = Input(shape=[1], name='user_idx')
item_idx = Input(shape=[1], name='item_idx')

user_emb = Embedding(input_dim=nb_user, output_dim=nb_latent_factor, input_length=1, embeddings_regularizer=reg, name='user_embedding')
item_emb = Embedding(input_dim=nb_item, output_dim=nb_latent_factor, input_length=1, embeddings_regularizer=reg, name='item_embedding')

u_lf = user_emb(user_idx)
i_lf = item_emb(item_idx)

SqueezeEmbed = Lambda(lambda x: K.squeeze(x, 1))
u_lf = SqueezeEmbed(u_lf)
```

Outline

- Latent Factor Based Matrix Factorization
- Feature Based Matrix Factorization
- Tensor Factorization
- Factorization Machines

2D feature interaction

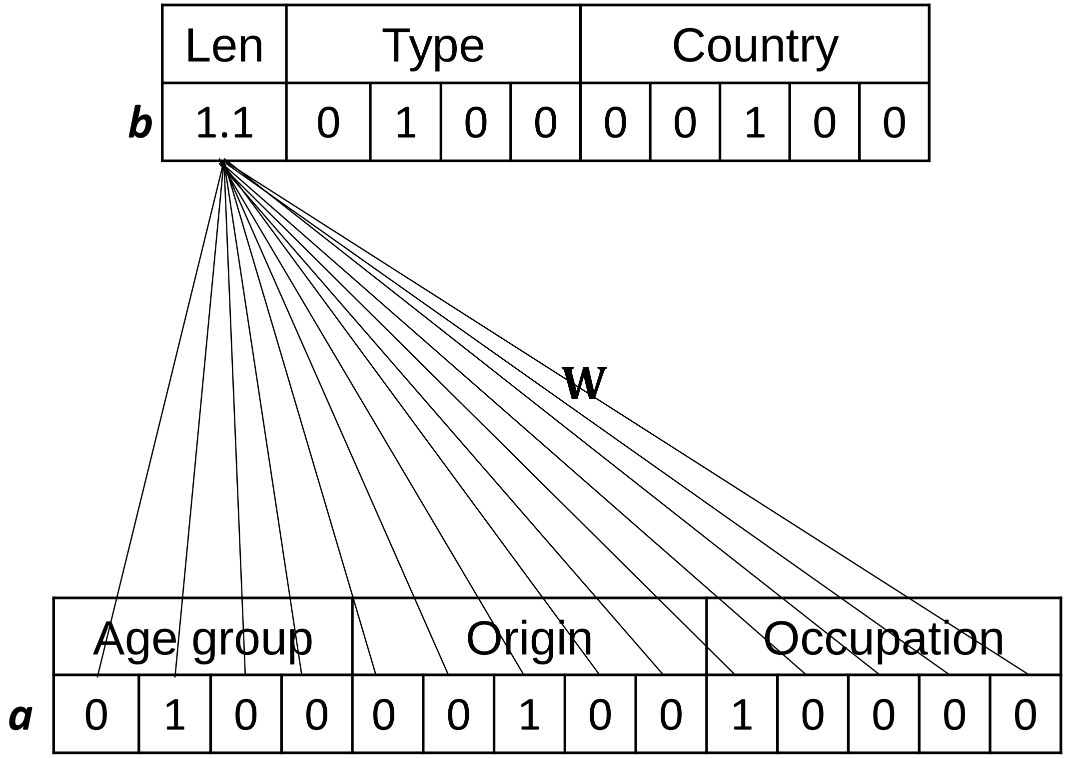
- Movie features:
  - Length: 120 Min, Type: Action, Country: U.S.

	Len	Type				Country				
a	0.8	0	1	0	0	0	0	1	0	0

- User features:
  - Age group: 20-30, Origin: China, Occupation: IT Engineer

	Age group				Origin					Occupation				
b	0	1	0	0	0	0	1	0	0	1	0	0	0	0

2D feature interaction



- The sum of all pairs of feature interaction

$$y = \sum_{i,j} a_i b_j w_{ij}$$

- Matrix form

$$y = \boldsymbol{a}^{\mathrm{T}}\mathbf{W}\mathbf{b}$$

- What's the problem?
  - If $|\boldsymbol{a}|$ and $|\boldsymbol{b}|$ is large, the parameter matrix $\mathbf{W}$ have the size $|\boldsymbol{a}| \times |\boldsymbol{b}|$.
  - TOO MANY parameters!

Feature-based matrix factorization

- Since the matrix $\mathbf{W}$ can be approximated by two low-rank matrices, i.e. $\mathbf{W} = \mathbf{U}^{\mathrm{T}}\mathbf{V}$, we have

$$y = \boldsymbol{a}^{\mathrm{T}}\mathbf{W}\mathbf{b} = \boldsymbol{a}^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}\mathbf{V}\mathbf{b}$$

  where $\mathbf{U} \in \mathbb{R}^{L\times|\boldsymbol{a}|}, \mathbf{V} \in \mathbb{R}^{L\times|\boldsymbol{b}|}$

- Let $\boldsymbol{m}^{\mathrm{T}} = \boldsymbol{a}^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}, \boldsymbol{n} = \mathbf{V}\mathbf{b}$, we have

$$y = \left(\boldsymbol{a}^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}\right)(\mathbf{V}\mathbf{b}) = \boldsymbol{m}^{\mathrm{T}}\boldsymbol{n}$$

- Normally, we often append the linear terms

$$y = \boldsymbol{m}^{\mathrm{T}}\boldsymbol{n} + \boldsymbol{a}^{\mathrm{T}}\boldsymbol{\beta}^{\boldsymbol{u}} + \boldsymbol{b}^{\mathrm{T}}\boldsymbol{\beta}^{\boldsymbol{v}}$$

Connect Feature-based MF to Latent factor based MF

- We represent users' IDs and items' IDs with one-hot encoding

	User 3									
$\boldsymbol{a}_3$	0	0	1	0	0	0	0	0	0	0

	Item 4									
$\boldsymbol{b}_4$	0	0	0	1	0	0	0	0	0	0

- Since $\mathbf{a}_3^{\mathrm{T}}\mathbf{U}^{\mathrm{T}} = \mathbf{U}_{:,3}^{\mathrm{T}}, \mathbf{V}\mathbf{b}_4 = \mathbf{V}_{:,4}, \mathbf{a}_3^{\mathrm{T}}\boldsymbol{\beta}^{\boldsymbol{u}} = \beta_3^u, \mathbf{b}_4^{\mathrm{T}}\boldsymbol{\beta}^{\boldsymbol{v}} = \beta_4^v$
- we have

$$y = \mathbf{a}_3^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}\mathbf{V}\mathbf{b}_4 + \mathbf{a}_3^{\mathrm{T}}\boldsymbol{\beta}^{\boldsymbol{u}} + \mathbf{b}_4^{\mathrm{T}}\boldsymbol{\beta}^{\boldsymbol{v}} = \mathbf{u}_3^{\mathrm{T}}\mathbf{v}_4 + \beta_3^u + \beta_4^v$$

Jointly modeling explicit features and latent factors

- Given users' IDs and users' features, items' IDs and items' features

	User i									
$\boldsymbol{u}_i$	0	0	1	0	0	0	0	0	0	0

	Age group				Origin					Occupation				
$\boldsymbol{a}_i$	0	1	0	0	0	0	1	0	0	1	0	0	0	0

	Item j									
$\boldsymbol{v}_j$	0	0	0	1	0	0	0	0	0	0

	Len	Type				Country				
$\boldsymbol{b}_j$	1.1	0	1	0	0	0	0	1	0	0

- We have

$$y_{ij} = \boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{v}_j + \boldsymbol{a}_i^{\mathrm{T}} \mathbf{P}^{\mathrm{T}} \mathbf{Q} \boldsymbol{b}_j + \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{\beta}^{\boldsymbol{u}} + \boldsymbol{b}_j^{\mathrm{T}} \boldsymbol{\beta}^{\boldsymbol{v}} + b_i + b_j$$

Bilinear model

- A function of two variables is bilinear if it is linear with respect to each of its variables

- For example:

$$y_{ij} = f(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^T \boldsymbol{v} + \boldsymbol{u} + \boldsymbol{v}$$

where $\boldsymbol{u} \in \mathbb{R}^{n\times 1}, \boldsymbol{v} \in \mathbb{R}^{n\times 1}$are independently linear in both those variables

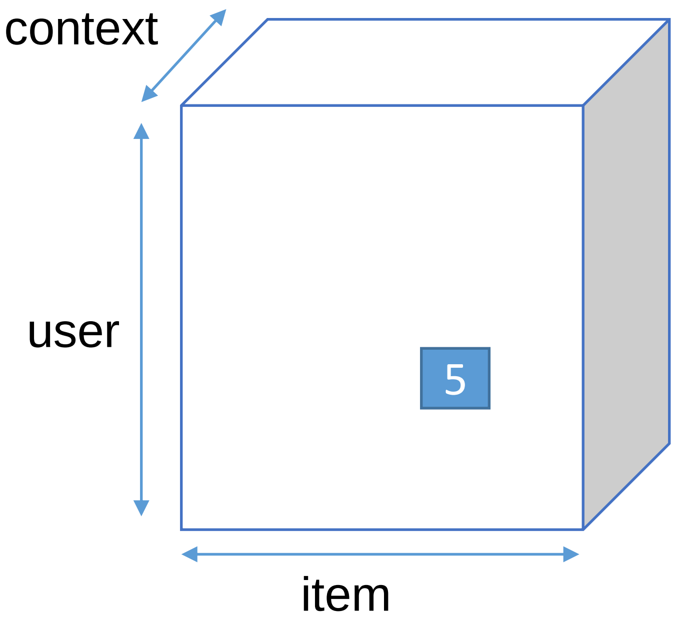
Outline

- Latent Factor Based Matrix Factorization
- Feature Based Matrix Factorization
- Tensor Factorization
- Factorization Machines

Higher order interaction

- Rating mapping without context
  - $User \times Item \rightarrow R$

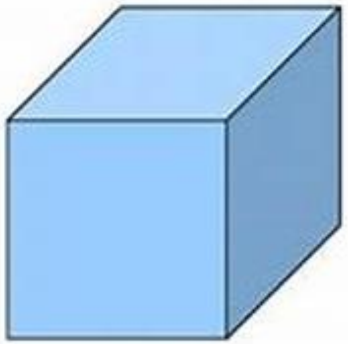
- Rating mapping with context
  - $User \times Item \times Context \rightarrow R$



Tensors

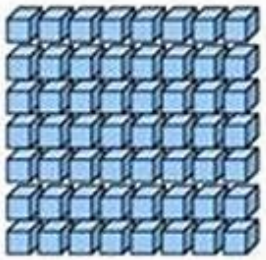
1d-tensor

2d-tensor

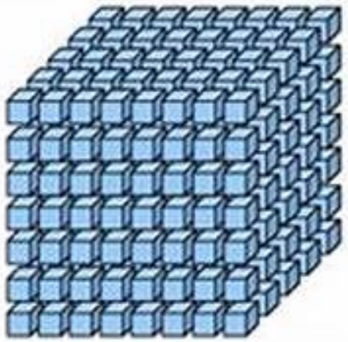


3d-tensor

4d-tensor



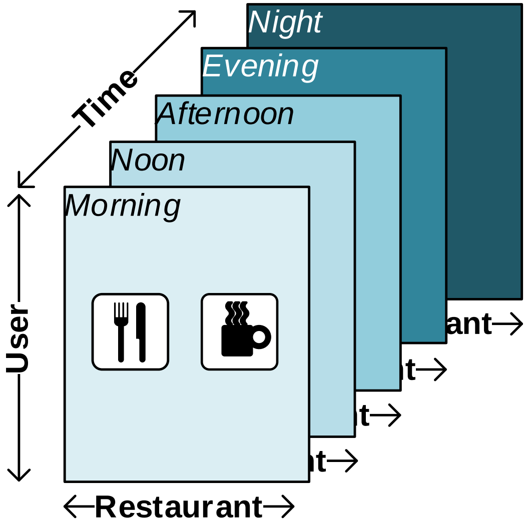
5d-tensor



6d-tensor

Example: Restaurant recommendation

- The recommendation varies with the time, so we have the triplet $\langle user, restaurant, time \rangle$.



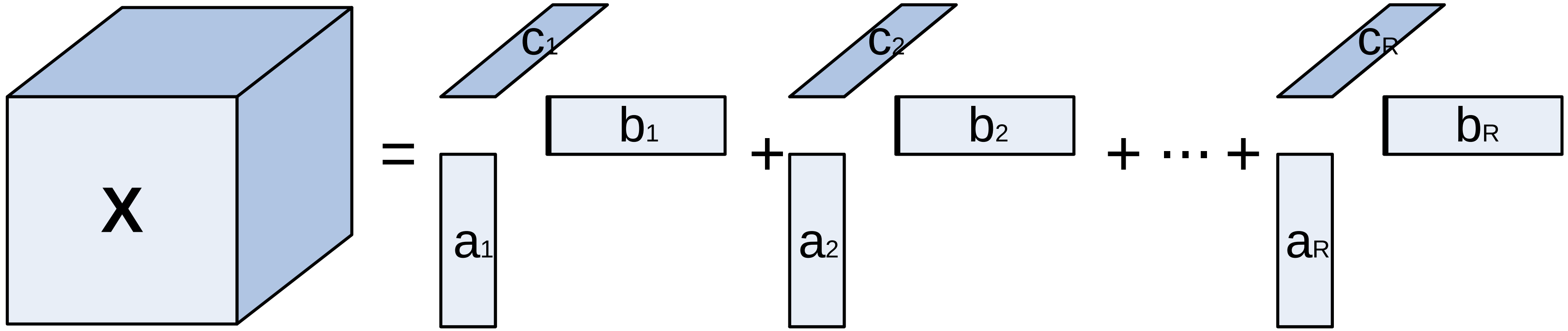
- Recommendation problem:

  Which restaurant should be recommended to a user in the afternoon?

Tensor Factorization over Triadic Relation

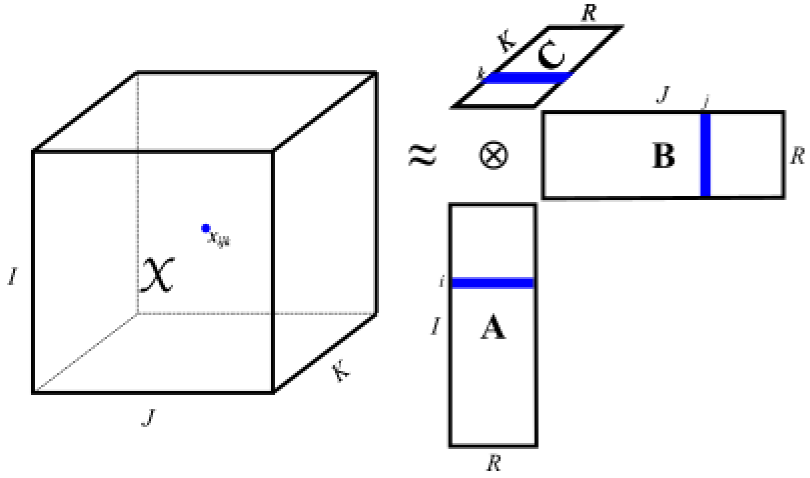
- **CP (CANDECOMP/PARAFAC) Model:** decompose a tensor into a sum of rank-one components

$$\mathcal{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = \sum_{r=1}^{R} \mathbf{A}_{r,:} \circ \mathbf{B}_{r,:} \circ \mathbf{C}_{r,:}, \qquad i.e., x_{i,j,k}^{r} = \mathbf{A}_{r,i}\mathbf{B}_{r,j}\mathbf{C}_{r,k}$$



Tensor Factorization:
Latent factor representation for high-order relation

- **CP Model:** $x_{i,j,k} = \sum_{r=1}^{R} \mathbf{A}_{r,i}\boldsymbol{B}_{r,j}\boldsymbol{C}_{r,k} = \langle \mathbf{A}_{:,i}, \boldsymbol{B}_{:,j}, \boldsymbol{C}_{:,k} \rangle$

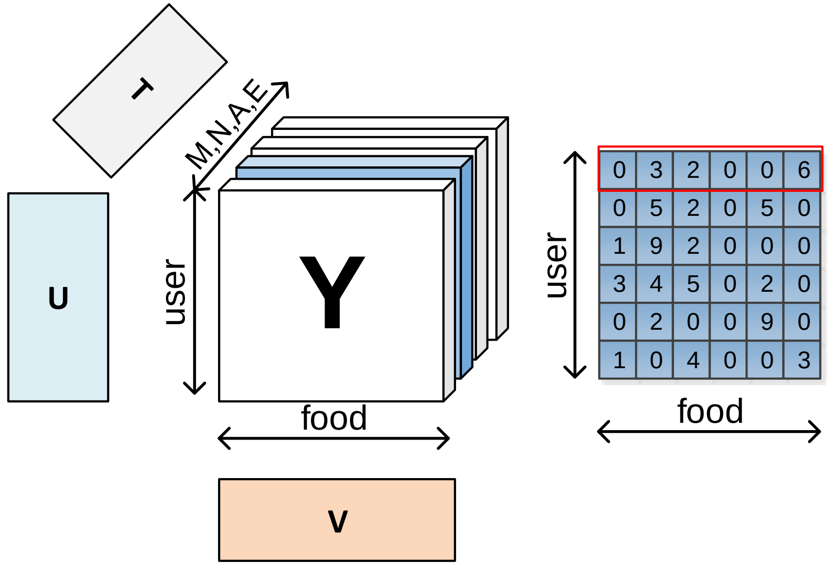


Full Storage: $O(\prod_i N_i)$, if N_i=100,000, each entry 4 bytes, 4PB (4×10^{15} bytes) memory is needed

Low-rank Storage: $O(D \sum_i N_i)$, if D=100, each factor 4 bytes, only 120MB memory is needed

TF for Recommendation

- $\mathbf{U}$: user factor matrix, $\mathbf{V}$: food factor matrix, $\mathbf{T}$: time factor matrix



- **Approximate the 3D tensor: $\mathcal{Y} \approx [\![\mathbf{U}, \mathbf{V}, \mathbf{T}]\!]$**

Objective setup for triadic interaction

- r_{ijt}: The rating on item j given by user i at time t
- $\boldsymbol{u}_i$: Latent factor vector of user i
- $\boldsymbol{v}_j$: Latent factor vector of item j
- $\boldsymbol{z}_t$: Latent factor vector of time t
- $e_{i,j,t}$: The error term

$$r_{ijt} = \langle \boldsymbol{u}_i * \boldsymbol{v}_j * \boldsymbol{z}_t \rangle + e_{i,j,t}$$

- Objective:

$$\arg \min_{\boldsymbol{u}_i, \boldsymbol{v}_j, \boldsymbol{z}_t} \left(r_{ijt} - \langle \boldsymbol{u}_i * \boldsymbol{v}_j * \boldsymbol{z}_t \rangle \right)^2$$

ALS

- Given user i and his/her rated item set $\boldsymbol{O}_i$

$$Loss = \frac{1}{\boldsymbol{O}_i}\sum_{j,t\in\boldsymbol{O}_i}\left(r_{ijt} - \langle \boldsymbol{u}_i * \boldsymbol{v}_j * \boldsymbol{z}_t \rangle\right)^2 + \lambda_A\|\boldsymbol{u}_i\|^2 + \lambda_B\left\|\boldsymbol{v}_j\right\|^2 + \lambda_C\|\boldsymbol{z}_t\|^2$$

- The loss function w.r.t. $\boldsymbol{u}_i$ is:

$$Loss = \frac{1}{\boldsymbol{O}_i}\sum_{j,t\in\boldsymbol{O}_i}\left(r_{ijt} - \boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{w}_{j,t}\right)^2 + \lambda_A\|\boldsymbol{u}_i\|^2$$

Where $\boldsymbol{w}_{j,t} = \boldsymbol{v}_j * \boldsymbol{z}_t$, the element-wise product

Probabilistic TF (PTF)

- Distribution of parameters:

$$\boldsymbol{u}_i \sim N(\mathbf{0}, \sigma_u^2 \mathbf{I}), \qquad \boldsymbol{v}_j \sim N(\mathbf{0}, \sigma_v^2 \mathbf{I}), \qquad \boldsymbol{z}_t \sim N(\mathbf{0}, \sigma_z^2 \mathbf{I})$$

- Distribution of ratings :

$$r_{i,j,t} = \langle \boldsymbol{u}_i * \boldsymbol{v}_j * \boldsymbol{z}_t \rangle + e_{i,j,t}$$

$$e_{i,j,t} \sim \boldsymbol{N}(0, \sigma_e^2) \Leftrightarrow r_{i,j,t} - \langle \boldsymbol{u}_i * \boldsymbol{v}_j * \boldsymbol{z}_t \rangle \sim \boldsymbol{N}(0, \sigma_e^2) \Leftrightarrow r_{i,j,t} \sim \boldsymbol{N}\left(\langle \boldsymbol{u}_i * \boldsymbol{v}_j * \boldsymbol{z}_t \rangle, \sigma_e^2\right)$$

- The joint log-likelihood of parameters and user-item-time interaction $\boldsymbol{O}$

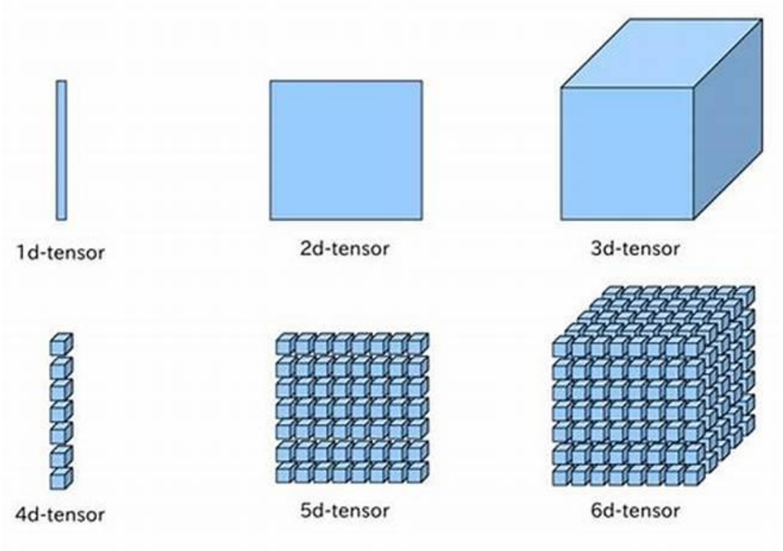
$$L = \log \prod_{u,i,t \in \boldsymbol{O}} N\left(r_{i,j,t} | \langle \boldsymbol{u}_i * \boldsymbol{v}_j * \boldsymbol{z}_t \rangle, \sigma_e^2\right) \prod_i N(\boldsymbol{u}_i | \mathbf{0}, \sigma_u^2 \mathbf{I}) \prod_j N\left(\boldsymbol{v}_j | \mathbf{0}, \sigma_v^2 \mathbf{I}\right) \prod_t N(\boldsymbol{z}_t | \mathbf{0}, \sigma_z^2 \mathbf{I})$$

Outline

- Latent Factor Based Matrix Factorization
- Feature Based Matrix Factorization
- Tensor Factorization
- Factorization Machines

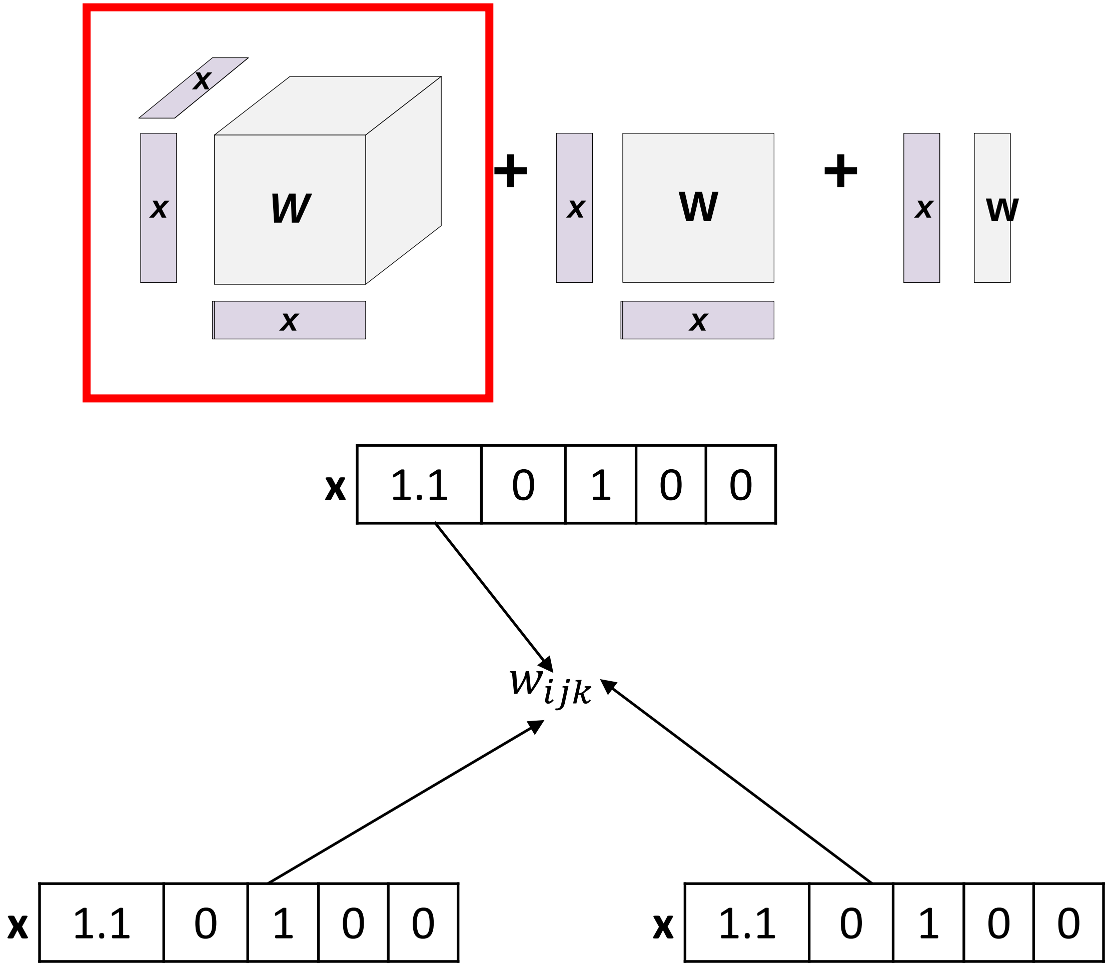
We need a framework to unify MF, FBMF, TF

- Factorization machines
  - The idea behind FMs is to model interactions between features using factorized parameters. The FM model has the ability to the estimate all interactions between features even with extreme sparse data.



Rendle, S., Gantner, Z., Freudenthaler, C., and Schmidt-Thieme, L. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 635-644, 2011.

3D feature interaction



- The sum of all triadic feature interaction

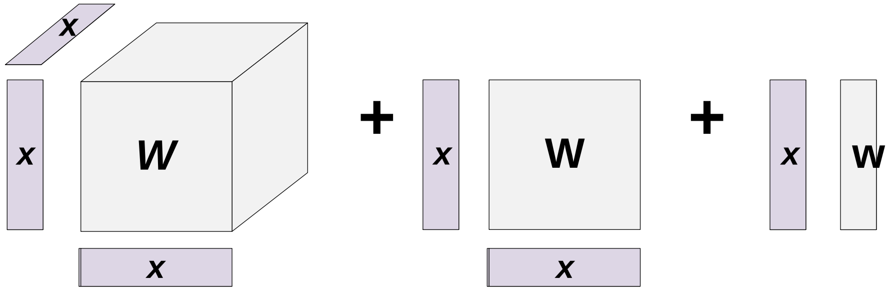
$$y = \sum_{i,j,k} x_i x_j x_k w_{ijk}$$

- Tensor form

$$y = \boldsymbol{\mathcal{W}} \times \boldsymbol{x}^{\mathrm{T}} \times \boldsymbol{x}^{\mathrm{T}} \times \boldsymbol{x}^{\mathrm{T}}$$

Model 3rd interactions

- $\boldsymbol{x}$: A feature vector with the length L
- y: The response
- The 3rd order interaction is:



$$y = \sum_{x_i, x_j, x_k \in \boldsymbol{x}} w_{i,j,k} x_i x_j x_k + \sum_{x_i, x_j \in \boldsymbol{x}} w_{i,j} x_i x_j + \sum_{x_i \in \boldsymbol{x}} w_i x_i + b + e_{i,j,k}$$

$$y = \boldsymbol{\mathcal{W}} \times \boldsymbol{x}^{\mathrm{T}} \times \boldsymbol{x}^{\mathrm{T}} \times \boldsymbol{x}^{\mathrm{T}} + \boldsymbol{x}^{\mathrm{T}} \mathbf{W} \boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}} \mathbf{w} + b + e_{i,j,k}$$

Where $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{L \times L \times L}, \mathbf{W} \in \mathbb{R}^{L \times L}, \mathbf{w} \in \mathbb{R}^{L}$

- What's the problem?
  - The number of parameters exponentially increases with the order.

Solution: factorization

- Factorize parameter tensors

$$\boldsymbol{\mathcal{W}} = [\![\mathbf{U}, \mathbf{U}, \mathbf{U}]\!], \qquad \mathbf{W} = \mathbf{V}^{\mathrm{T}}\mathbf{V}$$

- Replace parameter tensors $\boldsymbol{\mathcal{W}}$ and $\mathbf{W}$ with the factorization form

$$y = \sum_{x_i, x_j, x_k \in \boldsymbol{x}} \langle \mathbf{u}_i, \mathbf{u}_j, \mathbf{u}_{\boldsymbol{k}} \rangle x_i x_j x_k + \sum_{x_i, x_j \in \boldsymbol{x}} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j + \sum_{x_i \in \boldsymbol{x}} w_i x_i + b + e_{i,j,k}$$

$$\Rightarrow \quad y = \langle \boldsymbol{x}^{\mathrm{T}}\mathbf{U}, \boldsymbol{x}^{\mathrm{T}}\mathbf{U}, \boldsymbol{x}^{\mathrm{T}}\mathbf{U} \rangle + \langle \boldsymbol{x}^{\mathrm{T}}\mathbf{V}, \boldsymbol{x}^{\mathrm{T}}\mathbf{V} \rangle + \boldsymbol{x}^{\mathrm{T}}\mathbf{w} + b + e_{i,j,k}$$

- Regularized Loss function

$$\arg\min_{\mathbf{U},\mathbf{V},\mathbf{w}} \left[\left(\langle \boldsymbol{a}, \boldsymbol{a}, \boldsymbol{a} \rangle + \langle \boldsymbol{b}, \boldsymbol{b} \rangle + \boldsymbol{x}^{\mathrm{T}}\mathbf{w} + b \right) - y \right]^2 + \lambda_U \|\mathbf{U}\|^2 + \lambda_V \|\mathbf{V}\|^2 + \lambda_w \|\mathbf{w}\|^2$$

where $\boldsymbol{a} = \boldsymbol{x}^{\mathrm{T}}\mathbf{U}, \boldsymbol{b} = \boldsymbol{x}^{\mathrm{T}}\mathbf{V}$

Connect FM to latent factor MF

- Given user 2 and item 3, and the rating y

	user				item				
x	0	1	0	0	0	0	1	0	0

$$y = \sum_{x_i, x_j \in \boldsymbol{x}} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j + \sum_{x_i \in \boldsymbol{x}} w_i x_i$$

$$= \langle \mathbf{v}_2, \mathbf{v}_7 \rangle + {\color{red}\langle \mathbf{v}_2, \mathbf{v}_2 \rangle} + {\color{red}\langle \mathbf{v}_7, \mathbf{v}_7 \rangle} + w_2 + w_7$$

$$= \mathbf{v}_2^{\mathrm{T}} \mathbf{v}_7 + {\color{red}\mathbf{v}_2^{\mathrm{T}} \mathbf{v}_2} + {\color{red}\mathbf{v}_7^{\mathrm{T}} \mathbf{v}_7} + w_2 + w_7$$

More examples

Recommender Data

(A,TI,H,{C},5)
(A,NH,S,{},3)
(A,SW,N,{B,C},1)
(B,SW,N,{A,C},4)
(B,ST,H,{},5)
(C,TI,S,{A},1)
(C,SW,H,{A,B},5)

→

Feature vector x

	User			Movie				Mood			Watched with		
	A	B	C	TI	NH	SW	ST	S	N	H	A	B	C
$\mathbf{x}^{(1)}$	1	0	0	1	0	0	0	0	0	1	0	0	1
$\mathbf{x}^{(2)}$	1	0	0	0	1	0	0	1	0	0	0	0	0
$\mathbf{x}^{(3)}$	1	0	0	0	0	1	0	0	1	0	0	0.5	0.5
$\mathbf{x}^{(4)}$	0	1	0	0	0	1	0	0	1	0	0.5	0	0.5
$\mathbf{x}^{(5)}$	0	1	0	0	0	0	1	0	0	1	0	0	0
$\mathbf{x}^{(6)}$	0	0	1	1	0	0	0	1	0	0	1	0	0
$\mathbf{x}^{(7)}$	0	0	1	0	0	1	0	0	0	1	0.5	0.5	0

Target y

5	$y^{(1)}$
3	$y^{(2)}$
1	$y^{(2)}$
4	$y^{(3)}$
5	$y^{(4)}$
1	$y^{(5)}$
5	$y^{(6)}$

Four variables: *User, Movie, Mood and Watched with*

Trade-off between efficiency and efficacy

- Modeling high-order interaction is not always necessary
  - 4th order interaction: $\langle User, Movie, Mood, Watched\ with\rangle$
  - 3rd order interactions: $\langle User, Movie, Mood\rangle$, $\langle User, Movie, Watched\ with\rangle$, $\langle Mood, Watched\ with\rangle$
  - 2nd order interactions: $\langle User, Movie,\rangle$, $\langle User, Mood\rangle$, $\langle User, Watched\ with\rangle$, $\langle Movie, Mood\rangle$, $\langle Movie, Watched\ with\rangle$, $\langle Mood, Watched\ with\rangle$

- Normally, FM only models 2nd-order interactions, including nested ones (1st-order)

$$y = \sum_{x_i, x_j \in \boldsymbol{x}} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j + \sum_{x_i \in \boldsymbol{x}} w_i x_i$$

Deal with different types of outputs using FM

$$\hat{y} = \sum_{x_i, x_j, x_k \in \boldsymbol{x}} \langle \mathbf{u}_i, \mathbf{u}_j, \mathbf{u}_{\boldsymbol{k}} \rangle x_i x_j x_k + \sum_{x_i, x_j \in \boldsymbol{x}} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j + \sum_{x_i \in \boldsymbol{x}} w_i x_i$$

- For continuous data (e.g. rating)
  - MSE: $L = (\hat{y} - y)^2$
- For binary data (e.g. like/dislike)
  - Binary cross entropy: $L = y \log p + (1 - y) \log(1 - p)$
  - Where $p = sigmoid(\hat{y})$
- For unitary data (e.g. click)
  - Binary cross entropy: $L = y \log p + (1 - y) \log(1 - p)$
  - Where $\mathrm{y} = \delta(m \succ n), p = sigmoid\big(U(\hat{y}_m) - U(\hat{y}_n)\big)$
- For categorical data (e.g. multiple choice)
  - Multinomial cross entropy: $L = \sum y_k softmax(\hat{y})$

Summary

- To model 2nd order interaction, we study the latent factor based matrix factorization (LFMF) for rating prediction.

- Feature-based matrix factorization extends LFMF to allow modeling the dyadic interaction between two feature vectors.

- To model higher order interactions, we introduce the tensor factorization in terms of CP model.

- In the last section, we introduce a general factorization framework, namely factorization machines.

Assignment I

MovieLens 1M dataset contains 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 users.

Experimental dataset: https://grouplens.org/datasets/movielens/1m/

Task 1: Using ALS MF to predict movie ratings.

- Evaluation metrics: MAE, RMSE

Task 2: Using feature based MF or FM to predict movie ratings in terms of user and item attributes.

- Evaluation metrics: MAE, RMSE

Assignment II

Restaurant & consumer data for context-aware recommendation.

The tasks were to generate a top-n list of venue according to the consumer preferences at the given time.

Experimental dataset: https://www.kaggle.com/chetanism/foursquare-nyc-and-tokyo-checkin-dataset

Task 1: Using FM to model the user-item interaction $\langle User\ ID, Venue\ ID\rangle$.

- Evaluation metrics: Recall, MAP@5, MAP@10, nDCG@5, nDCG@10

Task 2: Using FM to model context-aware user-item interactions $\langle User\ ID, Venue\ ID, Venue\ Category, Checkin\ Time\rangle$.

- Evaluation metrics: Recall, MAP@5, MAP@10, nDCG@5, nDCG@10
- Note: the features are suggested to be encoded using sparse matrix