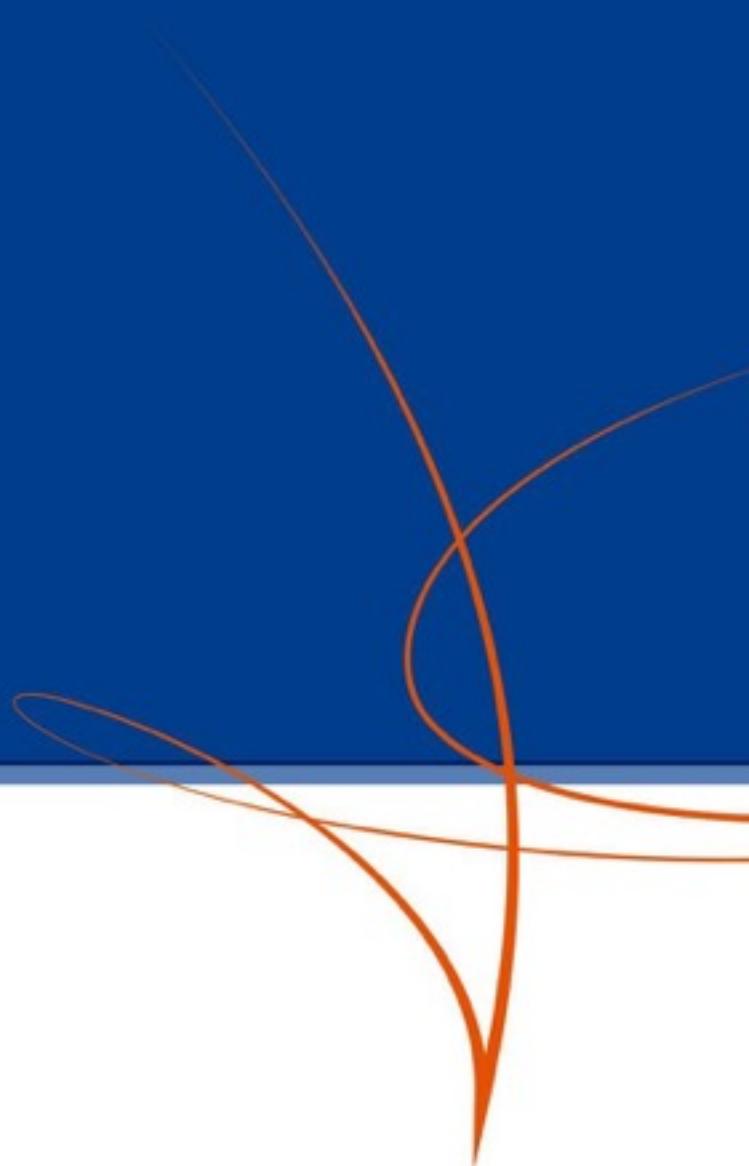


# 推荐系统



## ■ 推荐系统与评估

1. 推荐系统广泛应用
2. 推荐系统需求
3. 推荐系统结构与评估

## ■ 推荐算法串讲

1. 基于内容推荐 (content-based algorithms)
2. 协同过滤 (neighborhood-based algorithms)
3. 矩阵分解与隐语义模型 (LFM, FM)
4. word2vec在推荐系统中的简单应用

## □ 基于内容的推荐

- 基于用户喜欢的物品的属性/内容进行推荐
- 需要分析内容，无需考虑用户与用户之间的关联
- 通常使用在文本相关产品上进行推荐
- 物品通过内容(比如关键词)关联：
  - 电影题材：爱情/探险/动作/喜剧/悬疑
  - 标志特征：黄晓明/王宝强…
  - 年代：1995, 2016…
  - 关键词
- 基于比对物品内容进行推荐

## □ 基于内容的推荐

- 对于每个要推荐的内容，我们需要建立一份资料
  - 比如词 $k_j$ 在文件 $d_j$ 中的权重 $w_{ij}$
  - 常用的方法比如**TF-IDF**
- 需要对用户也建立一份资料：
  - 比如说定义一个权重向量 $(w_{c1}, \dots, w_{ck})$
  - 其中 $w_{ci}$ 表示第 $k_i$ 个词对用户 $c$ 的重要度
- 计算匹配度
  - 比如用余弦距离公式

$$u(c, s) = \cos(\vec{w}_c, \vec{w}_s) = \frac{\vec{w}_c \cdot \vec{w}_s}{\|\vec{w}_c\|_2 \times \|\vec{w}_s\|_2} = \frac{\sum_{i=1}^K w_{i,c} w_{i,s}}{\sqrt{\sum_{i=1}^K w_{i,c}^2} \sqrt{\sum_{i=1}^K w_{i,s}^2}}$$

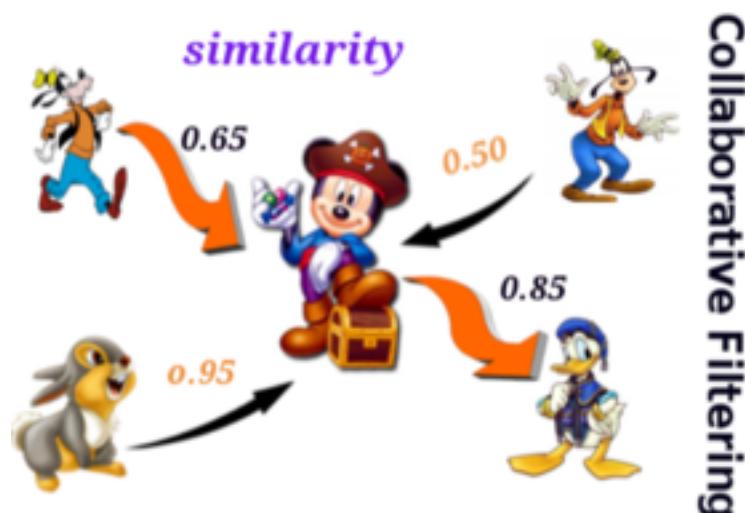
## □ *Neighborhood-based algorithm*

- 协同过滤是一种基于“近邻”的推荐算法
- 根据用户在物品上的行为找到物品或者用户的“近邻”



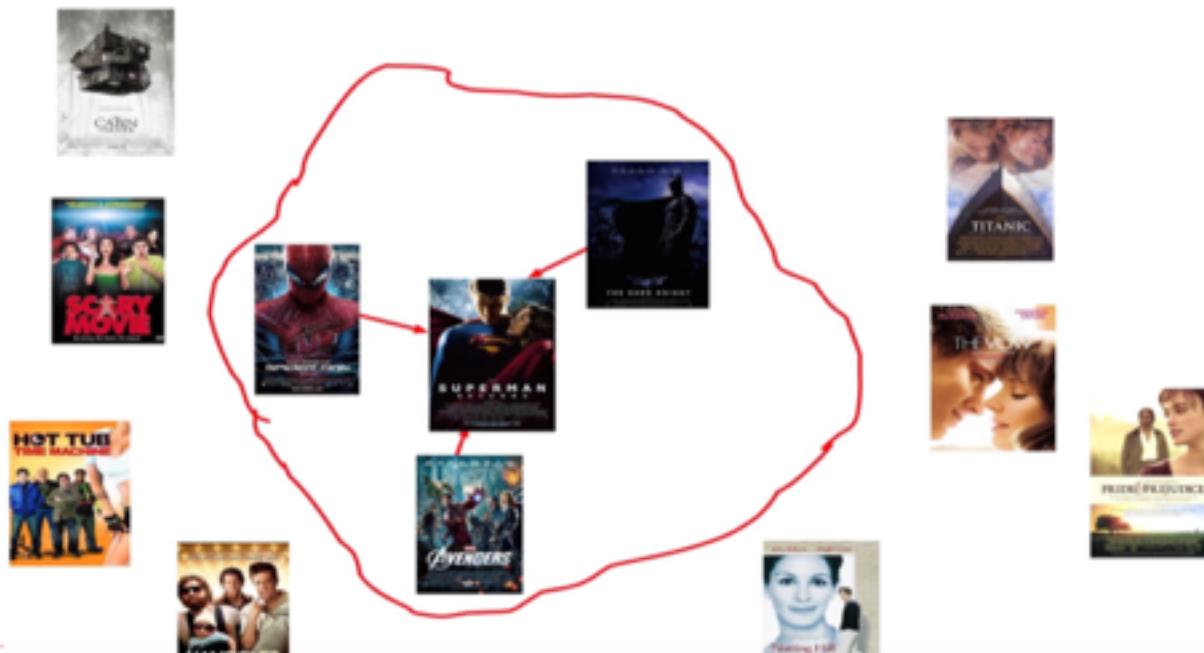
## □ 基于用户的协同过滤 (*user-based CF*)

- 基于用户有共同行为的物品，计算用户相似度
- 找到“近邻”，对近邻在新物品的评价(打分)加权推荐



## □ 基于物品的协同过滤 (*item-based CF*)

- 对于有相同用户交互的物品，计算物品相似度
- 找到物品“近邻”，进行推荐



## □ 相似度/距离定义

■ 欧氏距离

$$dist(X, Y) = \left( \sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}}$$

■ Jaccard相似度

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

■ 余弦相似度

$$\cos(\theta) = \frac{a^T b}{|a| \cdot |b|}$$

■ Pearson相似度

$$\frac{\sum_{i=1}^n (X_i - \mu_X)(Y_i - \mu_Y)}{\sqrt{\sum_{i=1}^n (X_i - \mu_X)^2} \sqrt{\sum_{i=1}^n (Y_i - \mu_Y)^2}}$$

## □ 基于物品的协同过滤

- 一个用户序列  $u_i, i=1 \dots n$ , 一个物品序列  $p_j, j=1 \dots m$
- $n \times m$  得分矩阵  $v$ , 每个元素  $v_{ij}$  表示用户  $i$  对物品  $j$  的打分
- 计算物品  $i$  和物品  $j$  之间的相似度 / 距离

$$S(i, j) = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

$$\sqrt{\sum_{i=1}^n (X_i - \mu_X)^2} \sqrt{\sum_{i=1}^n (Y_i - \mu_Y)^2}$$

- 选取  $\text{Top } K$  推荐或者加权预测得分

$$r_{xi} = \frac{\sum_{j \in N(i; x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i; x)} s_{ij}}$$

$s_{ij}$  similarity of items  $i$  and  $j$   
 $r_{xj}$  rating of user  $x$  on item  $j$   
 $N(i; x)$  set items rated by  $x$  similar to  $i$

## □ 基于物品的协同过滤

	users											
	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

 - estimate rating of movie 1 by user 5

## □ 基于物品的协同过滤

	1	2	3	4	5	6	7	8	9	10	11	12	users
1	1		3		?	5			5		4		sim(1,m)
2			5	4			4			2	1	3	
3	2	4		1	2		3		4	3	5		
4		2	4		5			4			2		
5			4	3	4	2					2	5	
6	1		3		3			2			4		

## □ 基于物品的协同过滤

	users											
	1	2	3	4	5	6	7	8	9	10	11	12
movies	1	1		3		2.6	5			5		4
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Predict by taking weighted average:

$$r_{1,5} = (0.41 \cdot 2 + 0.59 \cdot 3) / (0.41 + 0.59) = 2.6$$

$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum s_{ii}}$$

## □ 基于用户的协同过滤

- 一个用户序列  $u_i, i=1\dots n$ , 一个物品序列  $p_j, j=1\dots m$
- $n \times m$  得分矩阵  $v$ , 每个元素  $v_{ij}$  表示用户  $i$  对物品  $j$  的打分
- 计算用户相似度 (距离)

$$u_{ik} = \frac{\sum_j (v_{ij} - v_i)(v_{kj} - v_k)}{\sqrt{\sum_j (v_{ij} - v_i)^2 \sum_j (v_{kj} - v_k)^2}} \quad \text{or} \quad \cos(u_i, u_j) = \frac{\sum_{k=1}^m v_{ik} v_{jk}}{\sqrt{\sum_{k=1}^m v_{ik}^2 \sum_{k=1}^m v_{jk}^2}}$$

- 预测得分

$$v_{ij}^* = K \sum_{v_{ki} \neq ?} u_{jk} v_{kj} \quad \text{or} \quad v_{ij}^* = v_i + K \sum_{v_{ki} \neq ?} u_{jk} (v_{kj} - v_k)$$

## □ *User-based CF vs Item-based CF*

	UserCF	ItemCF
性能	适用于用户较少的场合，如果用户很多，计算用户相似度矩阵代价很大	适用于物品数明显小于用户数的场合，如果物品很多（网页），计算物品相似度矩阵代价很大
领域	时效性较强，用户个性化兴趣不太明显的领域	长尾物品丰富，用户个性化需求强烈的领域
实时性	用户有新行为，不一定造成推荐结果的立即变化	用户有新行为，一定会导致推荐结果的实时变化
冷启动	在新用户对很少的物品产生行为后，不能立即对他进行个性化推荐，因为用户相似度表是每隔一段时间离线计算的  新物品上线后一段时间，一旦有用户对物品产生行为，就可以将新物品推荐给和对它产生行为的用户兴趣相似的其他用户	新用户只要对一个物品产生行为，就可以给他推荐和该物品相关的其他物品  但没有办法在不离线更新物品相似度表的情况下将新物品推荐给用户
推荐理由	很难提供令用户信服的推荐解释	利用用户的历史行为给用户做推荐解释，可以令用户比较信服

### □ 协同过滤优点

- 基于用户行为，因此对推荐内容无需先验知识
- 只需要用户和商品关联矩阵即可，结构简单
- 在用户行为丰富的情况下，效果好

### □ 协同过滤缺点

- 需要大量的显性/隐性用户行为
- 需要通过完全相同的商品关联，相似的不行
- 假定用户的兴趣完全取决于之前的行为，而和当前上下文环境无关
- 在数据稀疏的情况下受影响。可以考虑二度关联。

□ 对于新用户

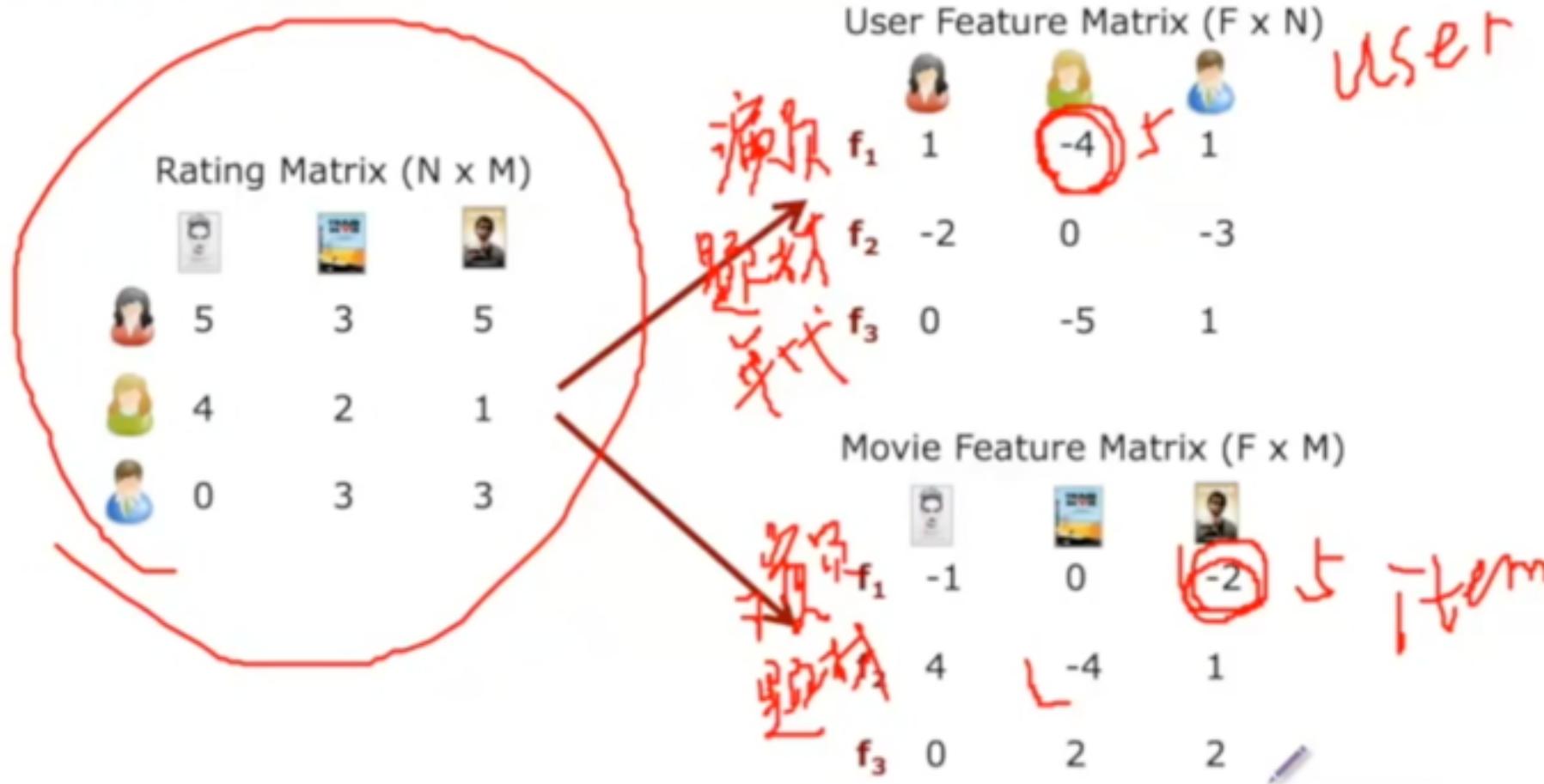
- 所有推荐系统对于新用户都有这个问题
- 推荐非常热门的商品，收集一些信息
- 在用户注册的时候收集一些信息
- 在用户注册完之后，用一些互动游戏等确定喜欢与不喜欢

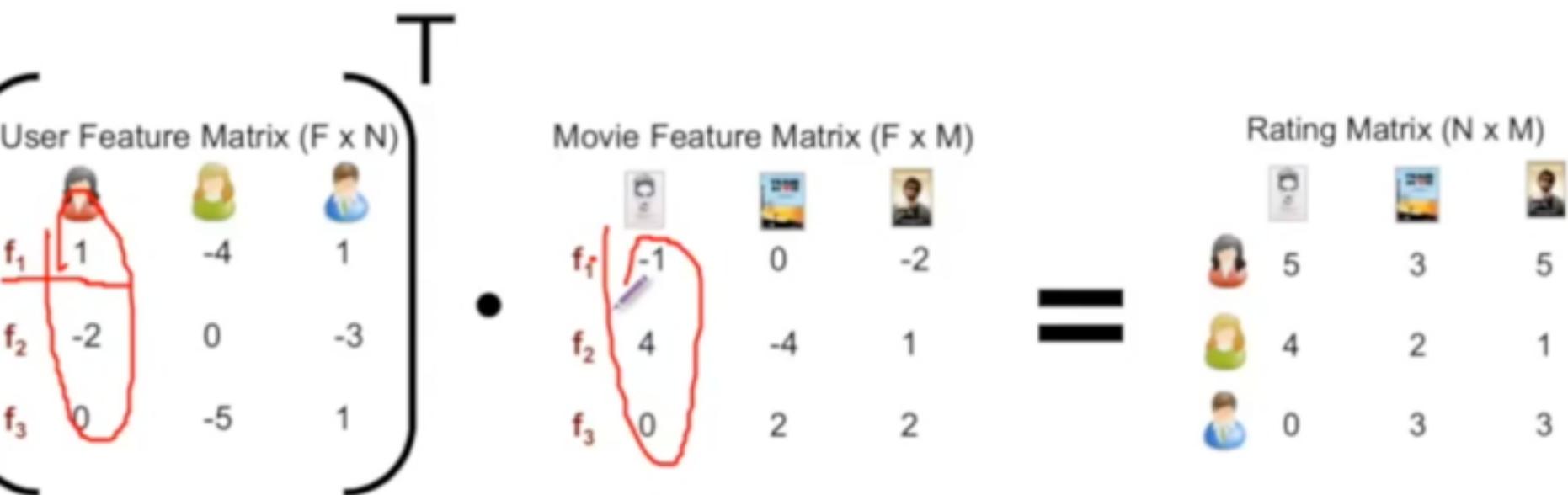
□ 对于新商品

- 根据本身的属性，求与原来商品的相似度。
- Item-based协同过滤可以推荐出去。

### □ 隐语义模型

- 我们有用户评分矩阵，其中部分位置是空着的（没打分）
- 我们希望尽量正确地填满未打分的项（预测得分）。
- 主要想法是，应该有一些隐藏的因素，影响用户的打分
  - 比如电影：演员、题材、主题、年代…
  - 不一定是人直接可理解的隐藏因子
  - 找到隐藏因子，可以对user和item进行关联
- 我们假定：
  - 隐藏因子的个数小于user和item数
  - 因为如果每个user都关联一个独立的隐藏因子，那就没法做预测了。

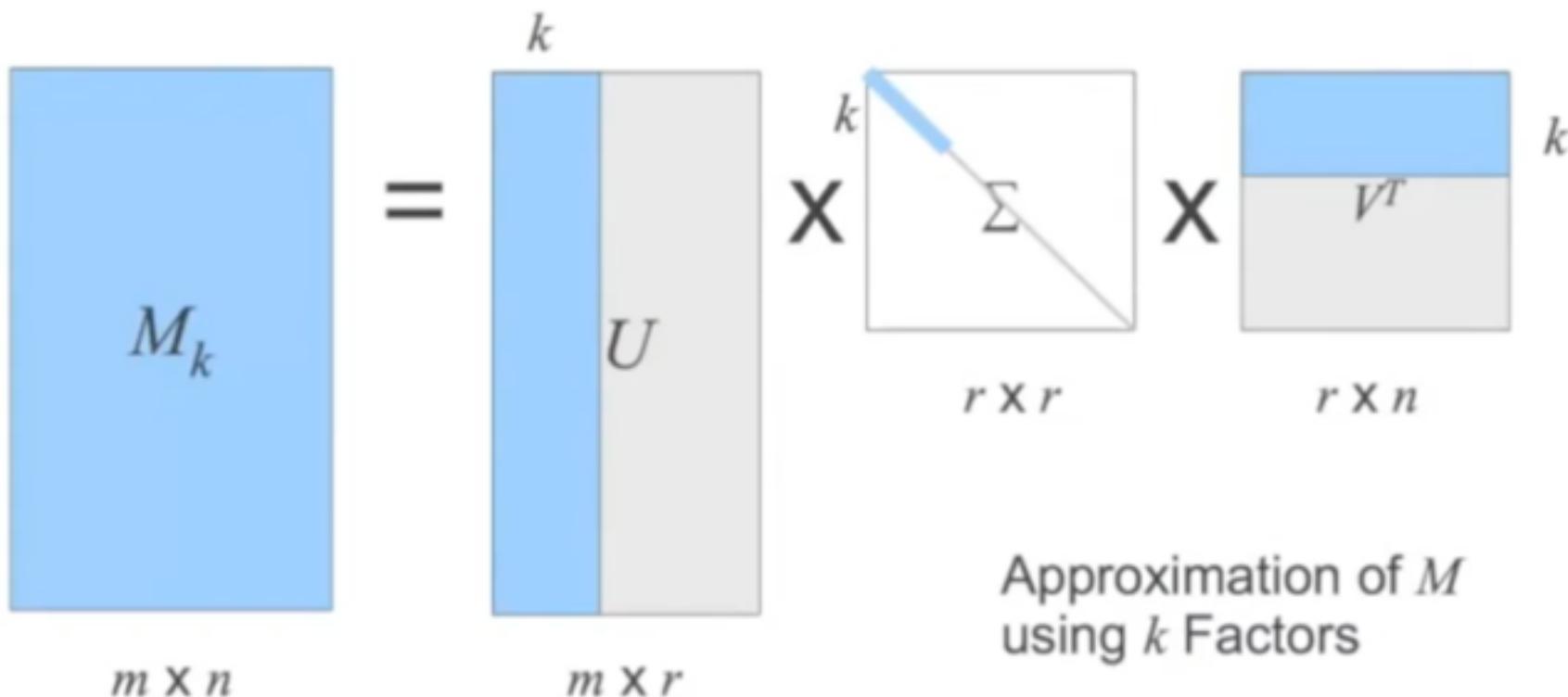




□ 简单使用**SVD**不好吗

□ **SVD**的时间复杂度是  $O(m^3)$

□ 原矩阵缺省值较多，不宜用0填补



## □ 隐语义模型

- 相比之下，CF简单、直接、可解释性强。
- 隐语义模型能更好地挖掘用户和物品隐藏关联关系。
- 隐语义模型覆盖度更好。

	D1	D2	D3	D4
U1	5	3	-	1
U2	4	-	-	1
U3	1	1	-	5
U4	1	-	-	4
U5	-	1	5	4

## 求解用户/物品特征矩阵 (*user/item feature matrix*)

- 假定有  $U$  个用户,  $D$  个物品,  $R$  为打分矩阵
- 假定有  $K$  个隐含变量, 我们需要找到矩阵  $P(U*K)$  和  $Q(D*K)$ :

$$R \approx P \times Q^T = \hat{R}$$

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^K p_{ik} q_{kj}$$

- 如何才能找到最佳的  $P$  和  $Q$  呢?

□ 梯度下降

① 定义损失函数

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^K p_{ik} q_{kj})^2$$

② 求解梯度

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2e_{ij}q_{kj}$$

$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2e_{ij}p_{ik}$$

求解用户/物品特征矩阵 (*user/item feature matrix*)

- 别忘了正则化:

$$e_{ij}^2 = (r_{ij} - \sum_{k=1}^K p_{ik} q_{kj})^2 + \frac{\beta}{2} \sum_{k=1}^K (\|P\|^2 + \|Q\|^2)$$

- 再次求梯度/偏导，更新迭代公式:

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + \alpha (2e_{ij} q_{kj} - \beta p_{ik})$$

$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + \alpha (2e_{ij} p_{ik} - \beta q_{kj})$$

- 用  $P * Q$  还原矩阵补充未打分项

- 为保证可解释性，通常会限定分解得到的  $P$  和  $Q$  中的元素都是非负的，即 **非负矩阵分解** (*NMF, Nonnegative Matrix Factorization*)。

➤ 因为不存在减法操作，因此可以看做对隐变量特征的贡献之和(加权)，完成原矩阵拟合。

□ 求解用户/物品特征矩阵 (*user/item feature matrix*)

□ 如何找到合适的矩阵  $P(U*K)$  和  $Q(D*K)$ :

□ 梯度下降

① 定义损失函数

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^K p_{ik}q_{kj})^2$$

② 求解梯度

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2e_{ij}q_{kj}$$

$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2e_{ij}p_{ik}$$

③ 迭代更新

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + 2\alpha e_{ij}q_{kj}$$

$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + 2\alpha e_{ij}p_{ik}$$

求解用户/物品特征矩阵 (*user/item feature matrix*)

- 别忘了正则化：

$$e_{ij}^2 = (r_{ij} - \sum_{k=1}^K p_{ik}q_{kj})^2 + \frac{\beta}{2} \sum_{k=1}^K (||P||^2 + ||Q||^2)$$

- 再次求梯度/偏导，更新迭代公式：

$$\begin{aligned} p'_{ik} &= p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + \alpha(2e_{ij}q_{kj} - \beta p_{ik}) \\ q'_{kj} &= q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + \alpha(2e_{ij}p_{ik} - \beta q_{kj}) \end{aligned}$$

- 用  $P * Q$  还原矩阵补充未打分项

- 为保证可解释性，通常会限定分解得到的  $P$  和  $Q$  中的元素都是非负的，即 **非负矩阵分解 (NMF, Nonnegative Matrix Factorization)**。

➤ 因为不存在减法操作，因此可以看做对隐变量特征的贡献之和（加权），完成原矩阵拟合。

## □ 隐语义模型进一步优化

$$e_{ij}^2 = (r_{ij} - \sum_{k=1}^K p_{ik}q_{kj})^2 + \frac{\beta}{2} \sum_{k=1}^K (||P||^2 + ||Q||^2)$$



### Baseline predictor

- Separates users and movies
- Benefits from insights into user's behavior
- Among the main practical contributions of the competition

### User-Movie interaction

- Characterizes the matching between users and movies
- Attracts most research in the field
- Benefits from algorithmic and mathematical innovations

- $\mu$  = overall mean rating
- $b_x$  = bias of user  $x$
- $b_i$  = bias of movie  $i$

## □ 加“偏置”的隐语义模型

$$r_{xi} = \mu + b_x + b_i + q_i \cdot p_x$$

Overall mean rating      Bias for user  $x$       Bias for movie  $i$   
 User-Movie Interaction

### ■ Example:

- Mean rating:  $\mu = 3.7$
- You are a critical reviewer: your ratings are 1 star lower than the mean:  $b_x = -1$
- Star Wars gets a mean rating of 0.5 higher than average movie:  $b_i = +0.5$
- Predicted rating for you on Star Wars:  
 $= 3.7 - 1 + 0.5 = 3.2$

## □ 加“偏置”的隐语义模型

- 依旧使用梯度下降优化，新加的参数同样需要正则化

$$\min_{Q, P} \sum_{(x, i) \in R} \left( r_{xi} - (\mu + b_x + b_i + q_i p_x) \right)^2$$

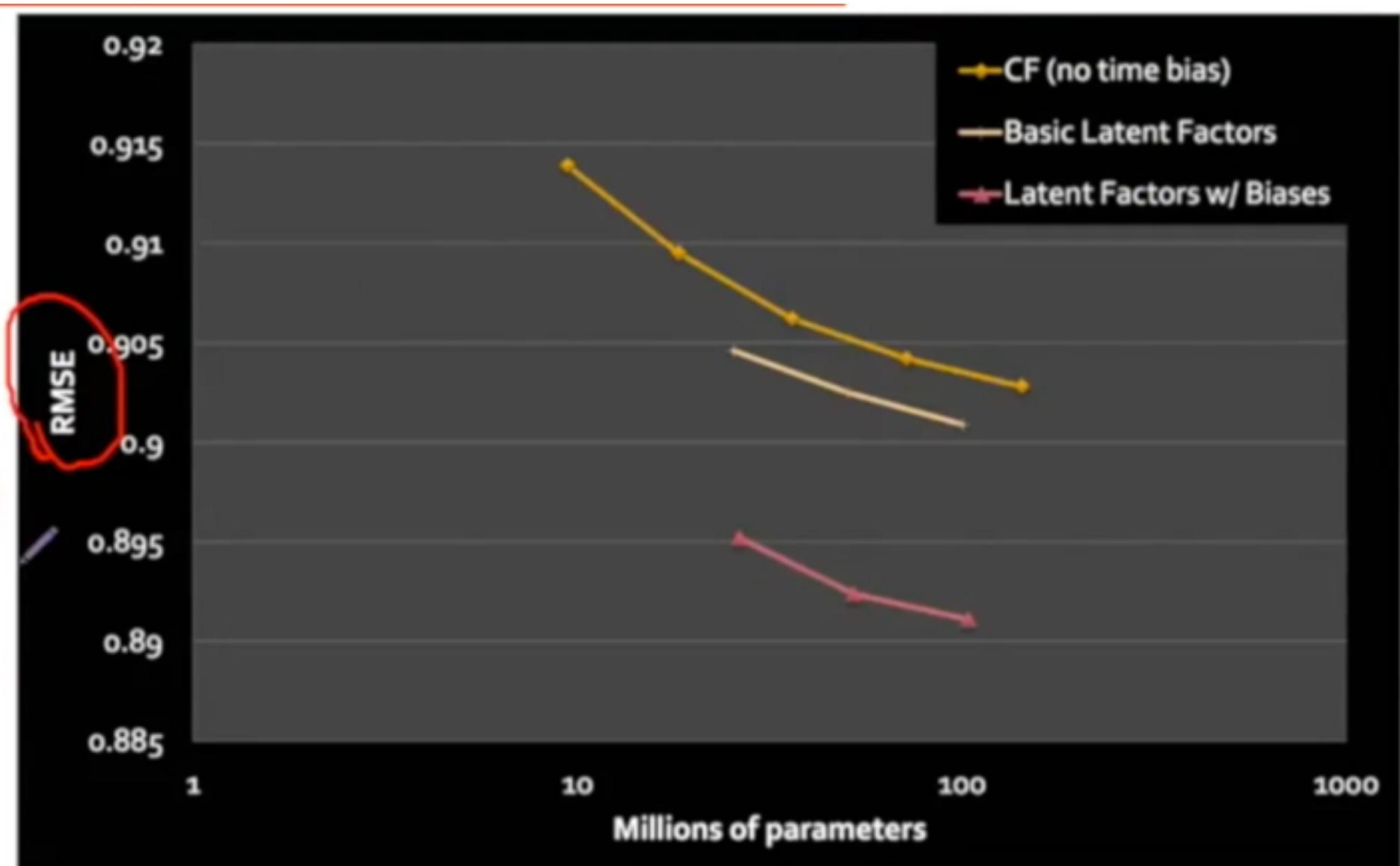
goodness of fit

$$+ \left( \lambda_1 \sum_i \|q_i\|^2 + \lambda_2 \sum_x \|p_x\|^2 + \lambda_3 \sum_x \|b_x\|^2 + \lambda_4 \sum_i \|b_i\|^2 \right)$$

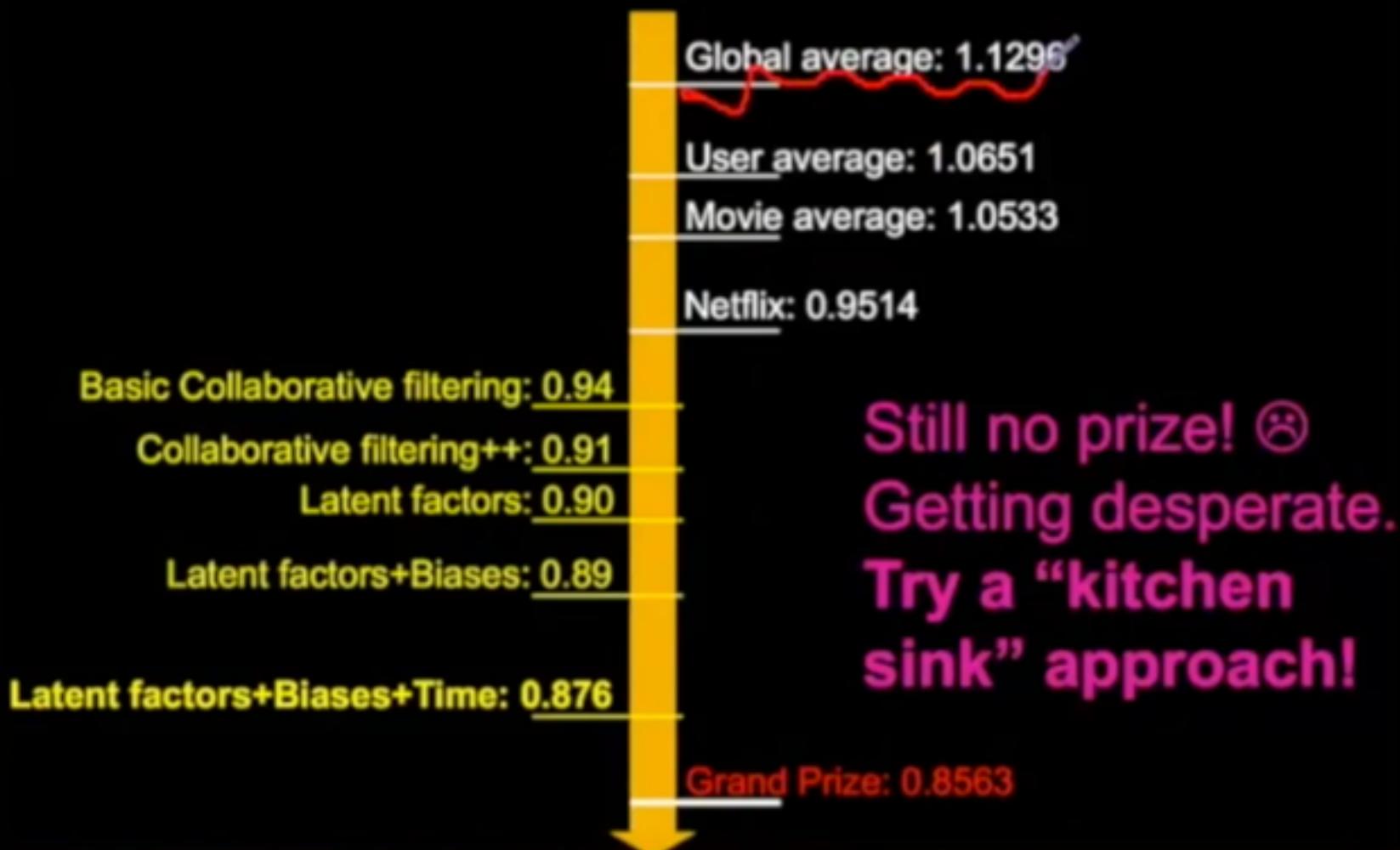
regularization

$\lambda$  is selected via grid-search on a validation set





# Performance of Various Methods



## □ Word2vec与用户行为序列

- 我们给定中文分词后的文本，使用word2vec能得到每个词(phrase)在高维空间的特征向量。
- 向量和向量之间的距离远近，表示2个词的关联度高低。
- 和“北京”最近的词为“东京”“柏林”“巴黎”“伦敦”

## □ 在推荐里怎么用？

- 把用户的行为序列当做分词过后的phrase
- 送给word2vec学习
- 根据商品映射得到的特征向量去找相似的商品

## □ 本质上也是体现商品关联，但是比协同过滤的覆盖度高

# Python Surprise



A Python scikit for  
recommender systems.

```
from surprise import SVD
from surprise import Dataset
from surprise.model_selection import cross_validate

# Load the movielens-100k dataset (download it if needed).
data = Dataset.load_builtin('ml-100k')

# Use the famous SVD algorithm.
algo = SVD()

# Run 5-fold cross-validation and print results.
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

Movielens 100k	RMSE	MAE	Time
SVD	0.934	0.737	0:00:11
SVD++	0.92	0.722	0:09:03
NMF	0.963	0.758	0:00:15
Slope One	0.946	0.743	0:00:08
k-NN	0.98	0.774	0:00:10
Centered k-NN	0.951	0.749	0:00:10
k-NN Baseline	0.931	0.733	0:00:12
Co-Clustering	0.963	0.753	0:00:03
Baseline	0.944	0.748	0:00:01
Random	1.514	1.215	0:00:01