Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心

基于协同过滤的推荐算法

Recommendation Algorithm Based on Collaborative Filtering

赵惟肖　硕士研究生

2018年08月26日

内容提要

- 背景简介
- 基本概念
- 算法原理
- 优劣分析
- 难点总结
- 参考文献

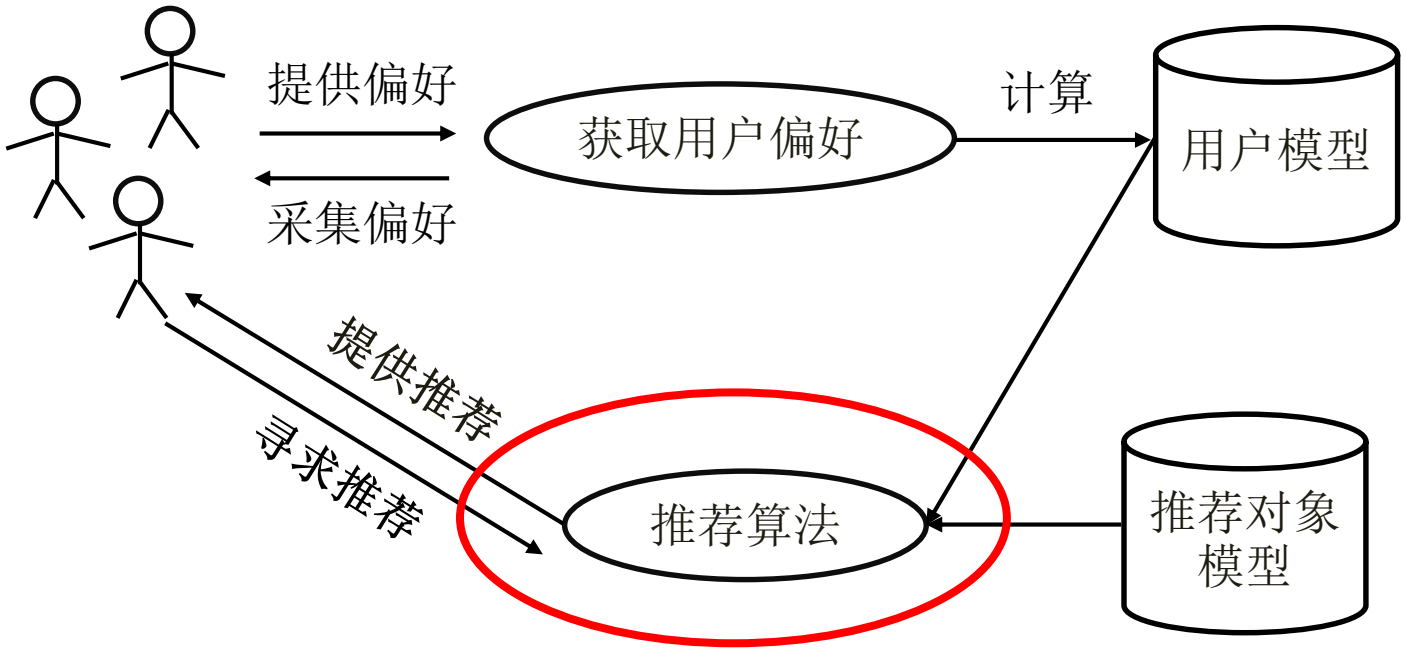
背景简介

背景简介

- **推荐系统**
- **前提：信息过载；用户需求的多样性。**
- **是一种信息过滤系统，用于向用户推荐用户感兴趣的信息和商品。**
- **应用于各行各业。推荐的对象包括：电影、音乐、新闻、书籍、搜索查询等。**

背景简介

- 推荐系统
- 有3个重要的模块：用户建模模块、推荐对象建模模块、推荐算法模块。



背景简介

- 推荐系统算法
- 基于内容的推荐
- 协同过滤推荐
- 基于关联规则的推荐
- 基于知识的推荐

基本概念

基本概念

- 协同过滤
- 简单来说是利用某兴趣相投、拥有共同经验之群体的喜好来推荐用户感兴趣的信息。
- 分为以下几种：

1.基于邻域

- -基于用户的协同过滤算法（UserCF）
- -基于物品的协同过滤算法（ItemCF）

2.基于模型

- -基于SVD的隐语义模型算法

3.其他

算法原理

算法原理

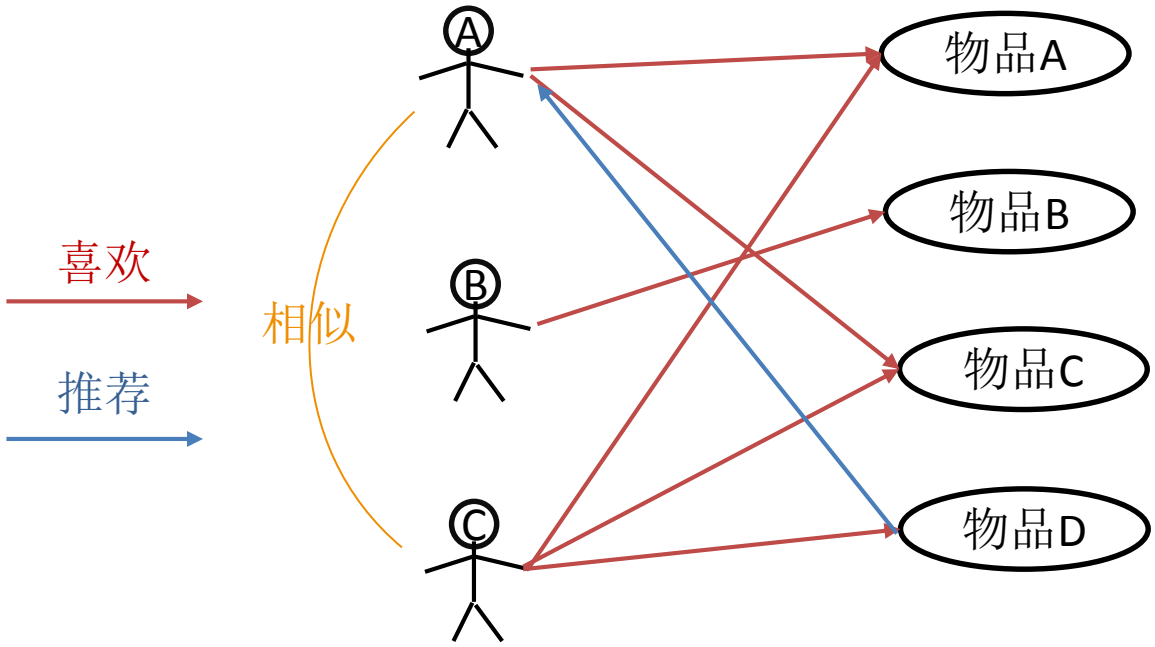
P	对用户进行个性化推荐
C	具备大量用户和物品间的行为信息
D	稀疏性、冷启动、可扩展、隐私保护
L	ACM A类会议(RecSys)

T	相似品味用户的喜好组织成一个有排序的目录
I	已有的用户对不同物品的喜好
P	收集用户偏好 找到相似的用户或物品 计算推荐
O	预测出的用户喜好

算法原理

- **基于用户的协同过滤（UserCF）**

用户/物品	物品A	物品B	物品C	物品D
用户A	√		√	推荐
用户B		√		
用户C	√		√	√



算法原理

- UserCF
- 1. 找到与目标用户兴趣相似的用户集合
- 2. 找到这个集合中用户喜欢的、并且目标用户没有听说过的物品推荐给目标用户

$$W_{AB} = \frac{|\{a,b,d\} \cap \{a,c\}|}{\sqrt{|\{a,b,d\}||\{a,c\}|}} = \frac{1}{\sqrt{6}}$$

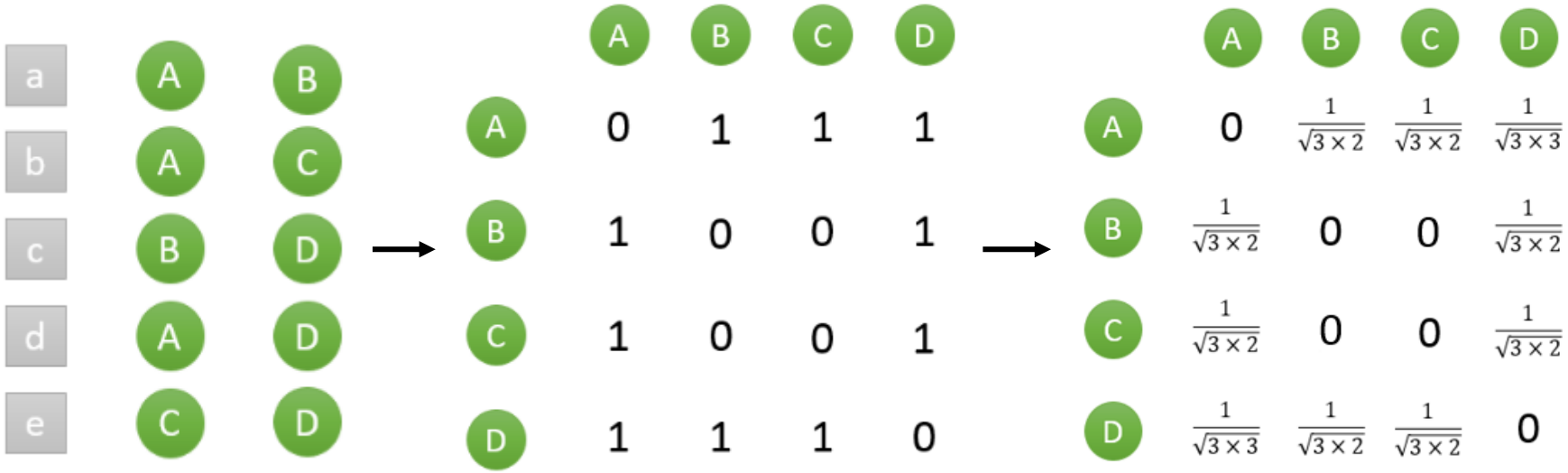
$$W_{AC} = \frac{|\{a,b,d\} \cap \{b,e\}|}{\sqrt{|\{a,b,d\}||\{b,e\}|}} = \frac{1}{\sqrt{6}}$$

$$W_{AD} = \frac{|\{a,b,d\} \cap \{c,d,e\}|}{\sqrt{|\{a,b,d\}||\{c,d,e\}|}} = \frac{1}{3}$$

$$W_{u,v} = \cos(r_u, r_v) = \frac{\sum_{i \in I} r_{u,i} \times r_{v,i}}{\sqrt{\sum_{i \in I} r_{u,i}^2} \sqrt{\sum_{i \in I} r_{v,i}^2}}$$

算法原理

- UserCF
- 建立物品到用户的倒查表T，表示该物品被哪些用户产生过行为。
- 然后对于每个物品，喜欢它的用户，在矩阵中他们两两加1。
- 计算用户两两之间的相似度。



算法原理

- UserCF
- 从矩阵中找出与目标用户 u 最相似的 K 个用户，用集合 N(u, K) 表示。
- 将 N 中用户喜欢的物品全部提取出来，并去除 u 已经喜欢的物品。
- 举例：假设我们要给 A 推荐物品，选取 K = 3 个相似用户，相似用户则是：B、C、D，他们喜欢过并且 A 没有喜欢过的物品有：c、e。

$$r(A,c) = r_{AB} + r_{AD} = \frac{1}{\sqrt{6}} + \frac{1}{\sqrt{9}} = 0.7416$$

$$r(A,e) = r_{AC} + r_{AD} = \frac{1}{\sqrt{6}} + \frac{1}{\sqrt{9}} = 0.7416$$

- 看样子用户 A 对 c 和 e 的喜欢程度可能是一样的。

算法原理

- **UserCF**
- 余弦相似度公式：$$W_{u,v} = \cos(r_u, r_v) = \frac{\sum_{i \in I} r_{u,i} \times r_{v,i}}{\sqrt{\sum_{i \in I} r_{u,i}^2} \sqrt{\sum_{i \in I} r_{v,i}^2}}$$
- 对当前用户评分的预测值：

$$\hat{r}_{u,i} = \frac{\sum_{v \in N_i(u)} w_{v,u} \times r_{v,i}}{\sum_{v \in N_i(u)} |w_{v,u}|}$$

- 考虑到不同用户对物品的评分标准是不一样的，进行归一化：

$$\hat{r}_{u,i} = \bar{r_u} + \frac{\sum_{v \in N_i(u)} w_{v,u} \times (r_{v,i} - \bar{r_v})}{\sum_{v \in N_i(u)} |w_{v,u}|}$$

- 有的用户评分波动大，有的用户相对稳定：

$$\hat{r}_{u,i} = \bar{r_u} + \sigma_u \frac{\sum_{v \in N_i(u)} w_{v,u} \times (r_{v,i} - \bar{r_u}) / \sigma_v}{\sum_{v \in N_i(u)} |w_{v,u}|}$$

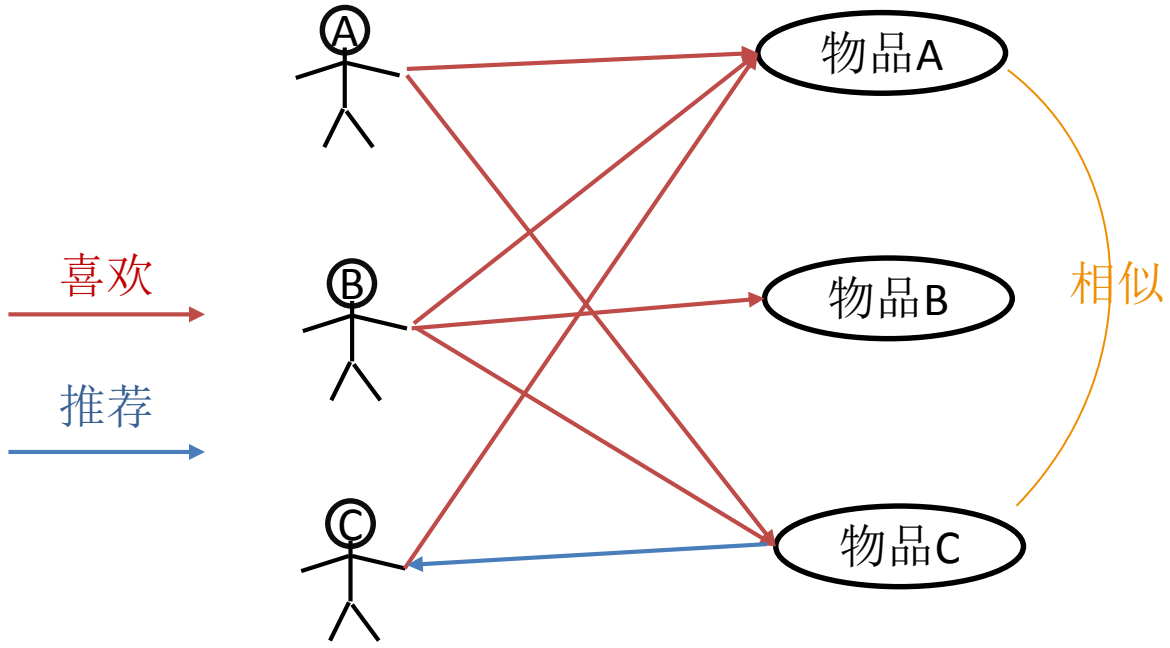
算法原理

- UserCF要解决的问题
- 用户矩阵Matrix R一般是非常稀疏的。
- 数百万的用户计算。
- 人的善变性。

算法原理

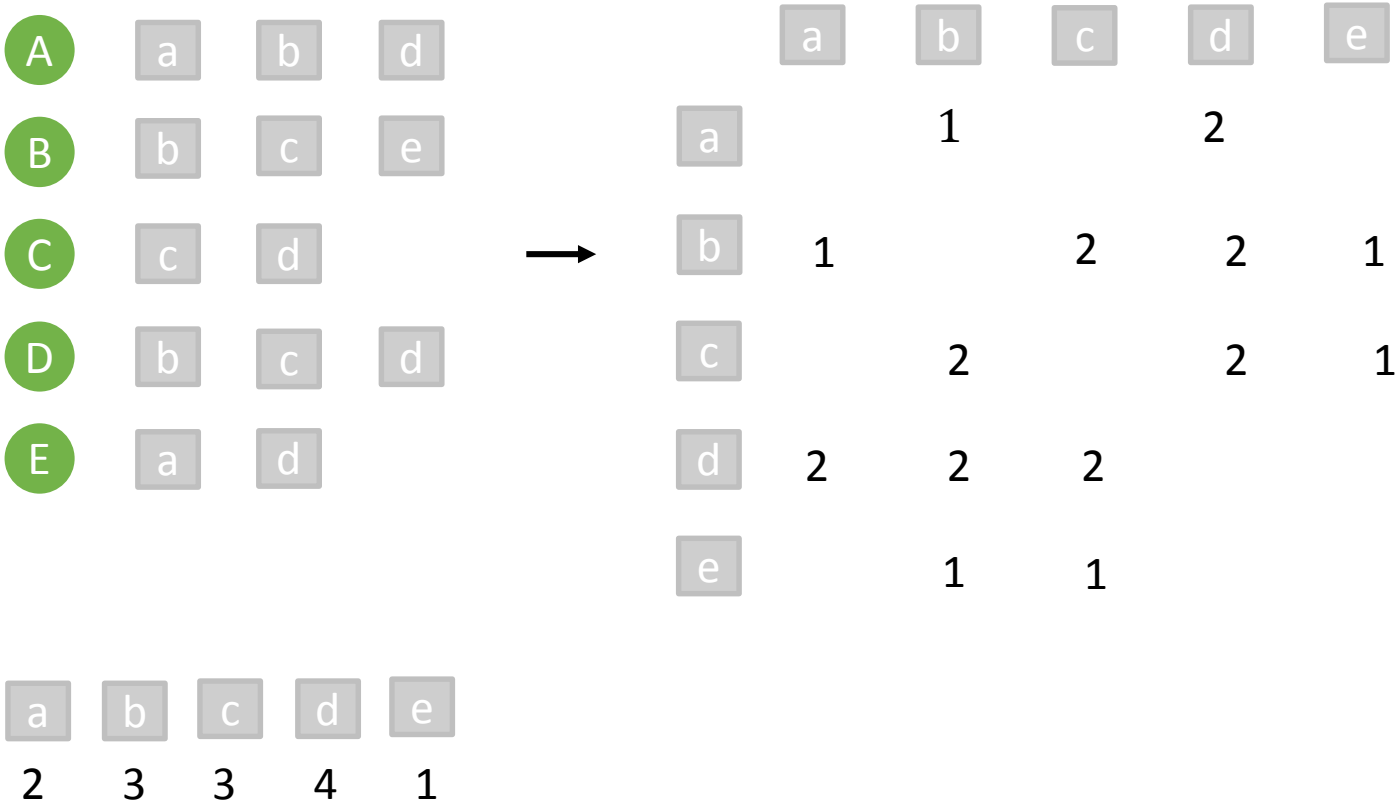
- 基于物品的协同过滤（ItemCF）

用户/物品	物品A	物品B	物品C
用户A	√		√
用户B	√	√	√
用户C	√		推荐



算法原理

- ItemCF
- 给用户推荐和他之前喜欢物品相似的物品



算法原理

- ItemCF
- 给用户推荐和他之前喜欢物品相似的物品

	a	b	c	d	e
a		1		2	
b	1		2	2	1
c		2		2	1
d	2	2	2		
e		1	1		

→

	a	b	c	d	e
a		$\frac{1}{\sqrt{2\times3}}$		$\frac{2}{\sqrt{2\times4}}$	
b	$\frac{1}{\sqrt{2\times3}}$		$\frac{2}{\sqrt{3\times3}}$	$\frac{2}{\sqrt{3\times4}}$	$\frac{1}{\sqrt{3\times1}}$
c		$\frac{2}{\sqrt{3\times3}}$		$\frac{2}{\sqrt{3\times4}}$	$\frac{1}{\sqrt{3\times1}}$
d	$\frac{2}{\sqrt{2\times4}}$	$\frac{2}{\sqrt{3\times4}}$	$\frac{2}{\sqrt{3\times4}}$		
e		$\frac{1}{\sqrt{3\times1}}$	$\frac{1}{\sqrt{3\times1}}$		

a	b	c	d	e
2	3	3	4	1

算法原理

- ItemCF
- 从矩阵中找出与目标物品 i 最相似的 K 个物品，用集合 N(i, K) 表示。
- 将 N 中喜欢该物品的用户全部提取出来。
- 举例：假设我们要给用户A推荐物品，用户A喜欢物品a,b和c，兴趣度为1, 2,2。选取 K = 2 个相似物品，相似物品是：d、e。

$$\mathrm{r}(A,d) = r_{\mathrm{ad}} + r_{\mathrm{bd}} + r_{\mathrm{cd}} = 1 * \frac{1}{\sqrt{2}} + 2 * \frac{1}{\sqrt{3}} + 2 * \frac{1}{\sqrt{3}} = 3.03$$

$$\mathrm{r}(A,e) = r_{\mathrm{ae}} + r_{\mathrm{be}} + r_{ce} = 2 * \frac{1}{\sqrt{3}} + 2 * \frac{1}{\sqrt{3}} = 2.32$$

- 可以把物品d推荐给用户A。

算法原理

- ItemCF

- 余弦相似度公式：$W_{i,j} = \cos(r_i, r_j) = \frac{\sum_{u \in U} r_{u,i} \times r_{u,j}}{\sqrt{\sum_{u \in U} r_{u,i}^2} \sqrt{\sum_{u \in U} r_{u,j}^2}}$

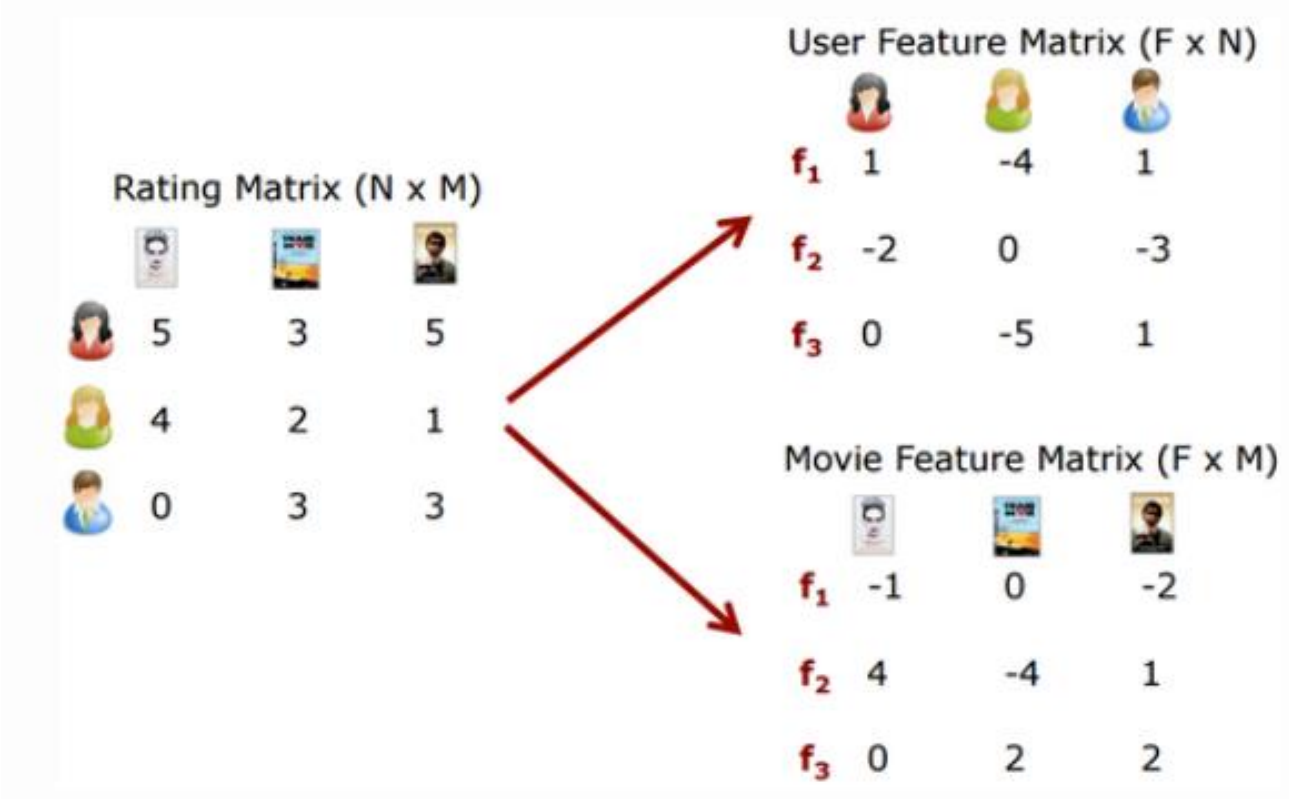
- 对当前用户评分的预测值：

$$\hat{r}_{u,i} = \frac{\sum_{j \in N_u(i)} w_{j,i} \times r_{u,j}}{\sum_{j \in N_u(i)} |w_{j,i}|}$$

- 通常用户数量远大于物品数量
- 可预先计算保留，物品并不善变

算法原理

- **基于模型的协同过滤算法（LFM）**
- 在推荐系统中，用户和物品之间没有直接关系。但是我们可以通过 feature 把它们联系在一起。



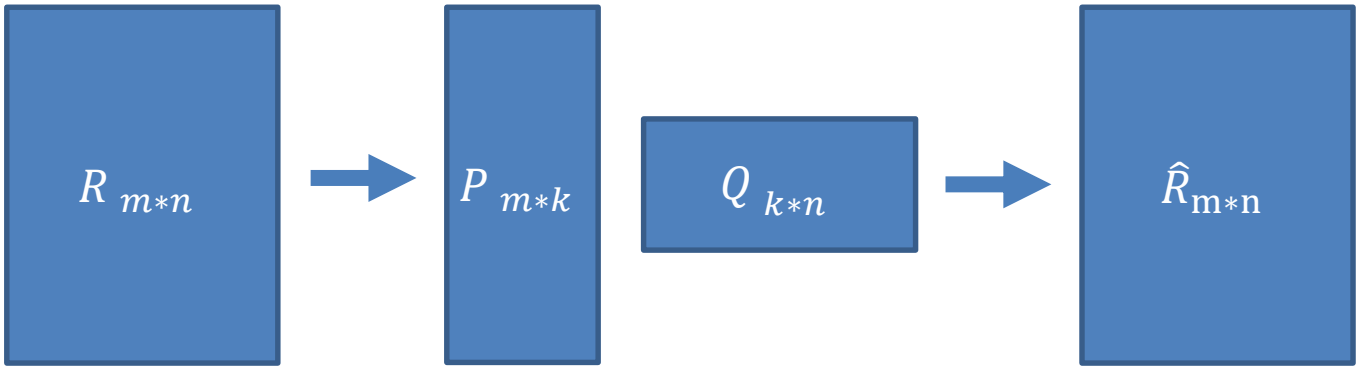
算法原理

- SVD
- SVD矩阵理论：假设 R 是一个m*n的矩阵，其中的元素全部属于数域 K（实数域 R 或复数域 C）。那么，存在 m×m 的酉矩阵 U 和 n×n 的酉矩阵 V 使得：

$$R = U \sum V^H$$

其中 Σ 是 m×n 的非负实数对角矩阵。

- 新的SVD模型：2006年Netflix电影推荐系统竞赛 Koren



- SVD
- SVD的转化

$$R = U\sum V^H = [u_1 \dots u_k | u_{k+1} \dots u_m] \begin{bmatrix} \sigma_1 & & & 0 \\ & \ddots & & \\ & & \sigma_k & \\ 0 & & & 0 \end{bmatrix} \begin{bmatrix} v_1^T \\ \vdots \\ v_k^T \\ v_{k+1}^T \\ \vdots \\ v_n^T \end{bmatrix}$$

$$= [u_1 \dots u_k] \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \begin{bmatrix} v_1^T \\ \vdots \\ v_k^T \end{bmatrix} + [u_{k+1} \dots u_m] \; [0] \begin{bmatrix} v_{k+1}^T \\ \vdots \\ v_n^T \end{bmatrix}$$

令$\mathrm{P} = [u_1 \dots u_k] \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} = [\sigma_1 u_1 \cdots \sigma_k u_k]$ $\qquad Q = \begin{bmatrix} v_1^T \\ \vdots \\ v_k^T \end{bmatrix}$

算法原理

- **SVD**
- SVD的模型构建

$$\hat{R}_{UI} = P_U Q_I = \sum_{k=1}^{K} P_{U,k} Q_{k,I}$$

- 定义损失函数

$$L = \sum_{(U,I)\in K} (R_{UI} - \hat{R}_{UI})^2 = \sum_{(U,I)\in K} (R_{UI} - \sum_{k=1}^{K} P_{U,k}\, Q_{k,I})^2 + \lambda||P_U||^2 + \lambda||Q_I||^2$$

算法原理

- **SVD**

$$C = \sum_{(U,I)\in K} (R_{UI} - \hat{R}_{UI})^2 = \sum_{(U,I)\in K} (R_{UI} - \sum_{k=1}^{K} P_{U,k}\, Q_{k,I})^2 + \lambda||P_U||^2 + \lambda||Q_I||^2$$

- **梯度下降**

$$\frac{\partial C}{\partial P_{U,k}} = -2\left(R_{UI} - \sum_{k=1}^{K} P_{U,k}Q_{k,I}\right)Q_{k,I} + 2\lambda P_{U,k}$$

$$\frac{\partial C}{\partial P_{U,k}} = -2\left(R_{UI} - \sum_{k=1}^{K} P_{U,k}Q_{k,I}\right)P_{U,k} + 2\lambda Q_{k,I}$$

- **迭代求解**

$$P_{U,k} = P_{U,k} + \alpha\left(\left(R_{UI} - \sum_{k=1}^{K} P_{U,k}Q_{k,I}\right) Q_{k,I} - \lambda P_{U,k}\right)$$

$$Q_{k,I} = Q_{k,I} + \alpha\left(\left(R_{UI} - \sum_{k=1}^{K} P_{U,k}Q_{k,I}\right) P_{U,k} - \lambda Q_{k,I}\right)$$

算法原理

- SVD

R

	0	1	2	3	4
0	4	3	0	5	0
1	5	0	4	4	0
2	4	0	5	0	3
3	2	3	0	1	0
4	0	4	2	0	5

↓

$\hat{R}$

	0	1	2	3	4
0	4.00602	2.99707	2.03283	4.96444	4.25486
1	4.95015	5.03918	4.0036	4.01224	4.1216
2	3.99701	3.96905	4.96942	2.85472	2.99905
3	2.03019	2.96364	2.13995	0.989033	1.92401
4	3.74247	3.99241	2.00265	4.32062	4.97782

优劣分析

- **基于邻域VS基于模型**
- 基于邻域算法一般在线推荐，可扩展性是主要挑战。

优点：简单直观、易于实现。

缺点：精度较基于模型算法较低。

- 基于模型算法一般离线训练，在线预测。

优点：精度、可扩展性高。

缺点：模型较复杂时，离线训练代价较高。若用户或项目更新较快，要频繁训练模型。

难点总结

难点总结

- **所面临的问题**
- 数据稀疏性（>98%）
- 冷启动:

用户冷启动：引导用户把自己的一些属性表达出来、根据用户注册属性、推荐排行榜单

物品冷启动：文本分析、主题模型、打标签

- 可扩展性：确保实时交互
- 隐私保护

参考文献

参考文献

[1]项亮. 推荐系统实践[M]. 人民邮电出版社, 2012.
[2] https://blog.csdn.net/guoxinian/article/details/62898742推荐系统中SVD算法详解
[3] https://blog.csdn.net/willduan1/article/details/51000601推荐算法分类：协同过滤、聚类、分类
[4]刘强. 协同过滤推荐系统中的关键算法研究[D]. 浙江大学, 2013.
[5]刘青文. 基于协同过滤的推荐算法研究[D]. 中国科学技术大学, 2013.

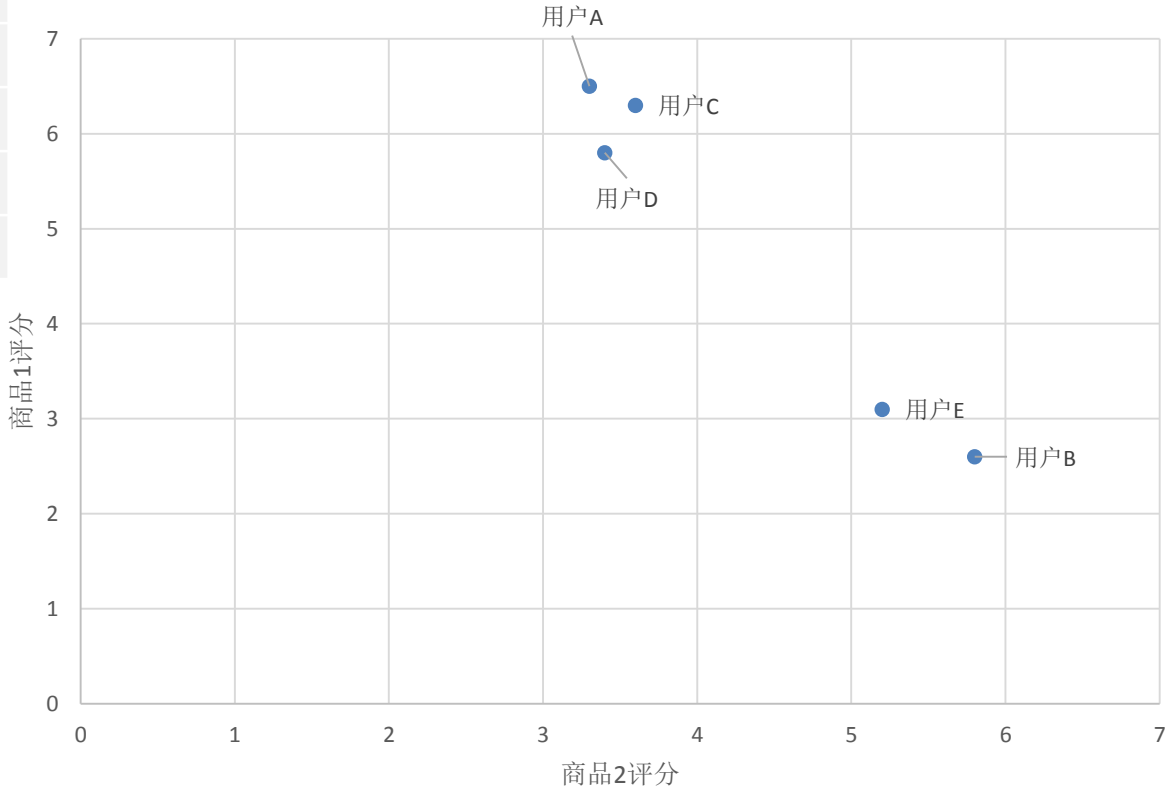
道德经

知人者智，自知者明。
胜人者有力，自胜者
强。知足者富。强行
者有志。不失其所者
久。死而不亡者，寿。

谢谢！

- **相似度计算**

	商品1	商品2
用户A	3.3	6.5
用户B	5.8	2.6
用户C	3.6	6.3
用户D	3.4	5.8
用户E	5.2	3.1



算法原理

- **相似度计算**
- 欧几里德距离（Euclidean Distance）

$$\mathrm{d}(x,y)=\sqrt{(\sum(x_i-y_i)^2)}$$

- 皮尔逊相关系数（Pearson Correlation Coefficient）

$$p(x,y)=\frac{\sum x_i y_i-n\bar{x}\bar{y}}{(n-1)S_x S_y}=\frac{n\sum x_i y_i-\sum x_i\sum y_i}{\sqrt{n\sum x_i^2-(\sum x_i)^2}\sqrt{n\sum y_i^2-(\sum y_i)^2}}$$

- Cosine 相似度（Cosine Similarity）

$$T(x,y)=\frac{x\cdot y}{||x||^2\times||y||^2}=\frac{\sum x_i y_i}{\sqrt{\sum x_i^2}\sqrt{\sum y_i^2}}$$

算法原理

- **相邻点的选择**
- （1）固定数量的相邻点
- （2）基于相似度门槛的相邻点