→一般稀疏

1. 效用矩阵 utility matrix：user 和 item，每个对应元素值代表当前用户对当前项喜好程度。

		item H_1	H_2	H_3
user	A	1	0	1
	B	⋮		
	C			
	D			

基于内容过程：

① 建立项模型，用一系列特征向量代表项。→使用 TF-IDF 打分，挑选分高的。

② 项模型表示：构建(特征-值)成对的项模型，用效用矩阵中每一行构建反映用户偏好的用户模型。

③ 构建用户模型。

1. 推荐系统的三种方法：

1. content-based 基于内容
2. collaborative 协作式
3. latent factor based 基于潜在因素 → 无需其他用户数据

① content-based：向用户x推荐 与 用户x评价较高的商品 类似的商品。 → 关注项的属性，项之间的相似度通过计算它们属性间的相似度确定。

eg：推荐同演员的电影。

item profile：项模型，用于代表该项的重要特征的一条或多条记录。 模型 → 项模型

eg：movie：演员、标题、导演
text：一系列关键词

① 文档特征：将具有最高 TF-IDF 得分的 n 个词作为一篇文档的特征。或词语

② 度量两文档相似度：△文档词之间 Jaccard 距离。△向量间余弦距离。

③ TF-IDF：

feature item

f_{ij}：term i 在 doc j 中的频率

$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$

Doc profile：一系列 TF-IDF 分数最高的词。

$IDF_i = \log \frac{N}{n_i}$

n_i：含 i 的 doc 数目。

N：doc 总数

TF-IDF score：$w_{ij} = TF_{ij} \times IDF_i$

④ user profile：关联 item profile 的加权平均

user profile → x　　item profile → i

$u(x,i) = \cos(x,i) = \frac{x \cdot i}{\|x\| \cdot \|i\|}$

余弦距离 $\frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum x_i^2} \cdot \sqrt{\sum y_i^2}}$

② collaborative Filtering. 找到与用户A相似度高的k个人，将k个人喜欢的推荐.

计算相似度：

Jacard距离：（适用于数据只有0、1）
- J相似度：交集大小 / 并集大小
- J距离：1 - 交/并

关注集合而不关注具体数值

余弦距离：$\frac{r_x \cdot r_y}{\|r_x\| \cdot \|r_y\|} = \cos(r_x, r_y)$ cos值越大代表角度越小，越近.

比较适合复杂如评分

归一化：每一个值要减平均值后计算.

皮尔逊系数：$r = \frac{\sum (x-\bar{x})\cdot(y-\bar{y})}{\sqrt{\sum (x-\bar{x})^2} \cdot \sqrt{\sum (y-\bar{y})^2}}$

预测（对项目）：r_x：user x's ratings 的 vector.

N：与x最相似的k个用户的集合，这些用户对项目i进行评分.

User-user

$s_{xy} = sim(x,y)$

$r_{xi} = \frac{1}{k}\sum_{y \in N} r_{yi}$ → 平均

$r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$ → 引入相似度.

两种

item-item 估计i的rating by 与i相似的item.

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

$N(i;x)$：x用户中相似于i的集合.

就是 nearest neighbors.

常规化（对k系数）→

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

↑ baseline

$b_{xi} = \mu + b_x + b_i$

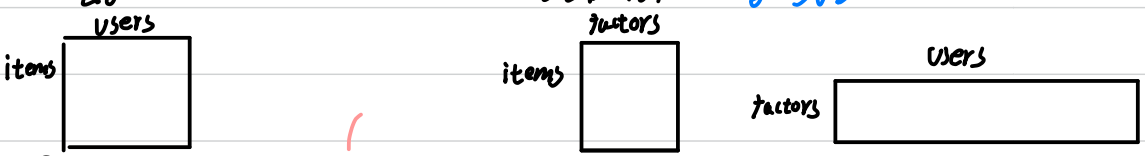
- μ：overall mean
- b_x：用户x的评分偏差（用户x评分平均数 - μ）
- b_i：novie i的评分偏差

评估：RMSE

$\sqrt{\Sigma_{xi}(r_{xi}-r_{xi}^*)^2}$ ：r_{xi}为预测，r_{xi}^*为真实。

$r_{2,4} = Q_2 \cdot P_4$

Latent Factor Model（潜在因素）：eg. SVD.



$R \approx Q \quad P^T$

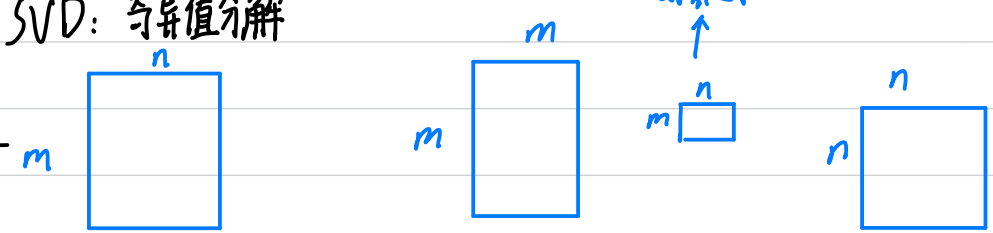
user x for item i.

估计缺失值：$\hat{r}_{xi} = q_i \cdot p_x = \sum_f q_{if} \cdot p_{xf}$

q_i：Q的行i

p_x：P^T的列x

SVD：奇异值分解



$A \approx U \cdot \Sigma \cdot V^T$

在使用中 $A=R$ $Q=U$ $P^T=\Sigma V^T$ $(A \approx Q P^T)$

$(r_{xi} \approx q_i \cdot p_x)$

找 P, Q：$\min_{P,Q} \Sigma_{(i,x)\in R}(r_{xi}-q_i \cdot p_x)^2$ → 让其最小

解决过拟合问题

$$\min_{P,Q} \sum_{train} (r_{xi} - q_i p_x)^2 + [\lambda_1 \sum_x \|p_x\|^2 + \lambda_2 \sum_i \|q_i\|^2]$$

error　　length

用户设定正则化参数.

考虑用户bias和电影bias，有 $r_{xi} = \mu + \underbrace{b_x}_{\text{用户bias}} + \underbrace{b_i}_{\text{电影bias}} + q_i \cdot p_x$

补充：　效用矩阵拆分的原理：考虑到用户一般只能对很小一部分特征感兴趣取，故拆成低维度的矩阵以分析特征.　它们在非零元素上会很相近.

$M = UV$　　└→且乘积的值为0可以预测M中空值

RMSE：衡量矩阵拆分后，UV和M的相近程度.

计算过程：　① M与UV，计算每一个元素差的平方，求合.

② 开根号

③ 除以

UV分解过程：

思路：选择任意U，V. 反复调整U与V使RMSE越来越小.

过程：

① 矩阵M的预处理

② U和V的初始化

③ U和V元素优化的排序

预处理：
- 每个 m_{ij} 减去用户 i 评分的平均值，再减去项 j 的平均值.
- 每个 m_{ij} 减去用户 i 和项平均评分的均值.

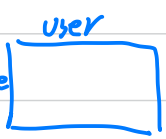
梯度的计算：X为输入，theta为模型参数，Y为输出.

梯度是一个向量，每个分量都表示损失函数对相应参数的偏导.

```
gra = np.dot(X.T, np.dot(X, theta) - Y) / len(Y)
```

SGD vs GD：
- GD 用总样本平均值来更新参数
- SGD 随机选一个样本更新参数.

PPT里 r_{xi} 为 none



矩阵中用户 x 对 movie i 的评分.

→ 防止迭代太多.

η 学习率用控控制步长. λ 为正则化参数 ∇ 为梯度.

$\nabla Q = [\Delta_{q\_it}]$ → 偏导数是一个向量，存放每个元素偏导.

$[q_{it}]$ 代表每一个 i 的偏导集合.