人工智能实践

Recommender System

孙海超
智能科技学院
西南财经大学天府学院

为什么使用推荐系统?

- Amazon: 35%的购买来自于推荐
- Netflix: 75%的观看的电影来自于推荐
- Google新闻: 38%的点击率来自于推荐

推荐系统的更多应用

- 购物：淘宝，Amazon
- 视频：Netflix，爱奇艺
- 音乐：Spotify，虾米音乐
- 新闻：今日头条，Google News
- 社交：微信，Facebook
- 广告……

推荐系统的类型

- 协同过滤(Collaborative Filtering-CF)的推荐
  - 基于内存的(Memory-based)/基于邻域的(Neighborhood-based)
    - 基于用户的(User-based)
    - 基于项目的(Item-based)
  - 基于模型的(Model-based)
    - 机器学习，潜在因子，矩阵分解等
- 基于内容的(Contented-based)推荐
- 混合的(Hybrid)

Neighborhood-based Collaborative Filtering

Neighborhood-based CF

- User-based: 相似的用户对同一个项目具有相似的打分
- Item-based: 相似的项目收到相似的打分

本节的一些注解约定

- 用户u: u
- 用户数: m
- 用户v: v
- 项目j: j
- 一个m x n的打分矩阵: $R = [r_{uj}]$
- 用户u打过分的项目的索引集 :I_u
- 对项目j打过分的最接近用户u的k个用户的集合: $P_u(j)$

计算用户u的平均打分:

$$\mu_u = \frac{\sum_{k \in I_u} r_{uk}}{|I_u|} \quad \forall u \in \{1 \dots m\}$$

计算用户u和用户v相似度(similarity):

使用Pearson相关系数:

$$Sim(u,v) = \frac{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u) \cdot (r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u)^2} \cdot \sqrt{\sum_{k \in I_u \cap I_v} (r_{vk} - \mu_v)^2}}$$

或Cosine函数:

$$Sim(u,v) = \frac{\sum_{k \in I_u \cap I_v} r_{uk} \cdot r_{vk}}{\sqrt{\sum_{k \in I_u} {r_{uk}}^2} \cdot \sqrt{\sum_{k \in I_v} {r_{vk}}^2}}$$

由于不同用户打分的基准(scale)可能不同，原始打分数据需要被平均中心化(mean-centered),用户u对项目j平均中心化之后的打分s_{uj}:

$$s_{uj} = r_{uj} - \mu_u \quad \forall u \in \{1 \dots m\}$$

用户u对项目j的打分预测:

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in P_u(j)} Sim(u,v) \cdot S_{vj}}{\sum_{v \in P_u(j)} |Sim(u,v)|}$$

Item_id / User_id	1	2	3	4	5	6	Mean	Sim
1	7	6	7	4	5	4	5.5	0.894
2	6	7		4	3	4	4.8	0.939
3	?	3	3	1	1		2	1
4	1	2	2	3	3	4	2.5	-1
5	1		1	2	3	3	2	-0.817

Item_id / User_id	1	2	3	4	5	6
1	1.5	0.5	1.5	-1.5	-0.5	-1.5
2	1.2	2.2		-0.8	-1.8	-0.8
3	?	1	1	-1	-1	
4	-1.5	-0.5	-0.5	0.5	0.5	1.5
5	-1		-1	0	1	1

Mean centered

计算$\hat{r}_{31}$:

$$\mu_1 = (7+6+7+4+5+4)/6 = 5.5$$

$$\mu_3 = (3+3+1+1)/4 = 2$$

$$Sim(1,3) = \frac{(6-5.5)*(3-2)+(7-5.5)*(3-2)+(4-5.5)*(1-2)+(5-5.5)*(1-2)}{\sqrt{0.5^2+1.5^2+(-1.5)^2+(-0.5)^2}\cdot\sqrt{1+1+1+1}}$$

$$= 0.894$$

$$P_3(j) = \{1,2\}$$

$$\hat{r}_{31} = 2 + \frac{1.5*0.894+1.2*0.939}{0.894+0.939} = 3.35$$

长尾问题(Long tail problem)

- 只有少数的高频项目被经常访问
- 在购物推荐中长尾的项目价值大，在电影推荐中用户可以提高用户的惊喜度
- 解决方法：逆向用户频率(Inverse User Frequency)

Inverse User Frequency

- 用户总数: m
- 项目j获得的打分个数: m_j
- 项目j的权重: w_j

$$w_j = \log\left(\frac{m}{m_j}\right) \quad \forall j \in \{1 \dots n\}$$

此时的相似度:

$$Sim(u,v) = \frac{\sum_{k \in I_u \cap I_v} w_k \cdot (r_{uk} - \mu_u) \cdot (r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \cap I_v} w_k \cdot (r_{uk} - \mu_u)^2} \cdot \sqrt{\sum_{k \in I_u \cap I_v} w_k \cdot (r_{vk} - \mu_v)^2}}$$

Item-Based Neighborhood的一些约定

- 打分矩阵: $R = [r_{uj}]$
- 给项目i打过分的用户索引集: U_i
- 给项目j打过分的用户索引集: U_j
- 用户u打过分的和项目t最相似的k个项目的集合: $\mathrm{Q}_t(u)$

对于基于项目的协同过滤，建议使用修正余弦相似度(Adjusted Cosine Similarity)

$$Sim(i,j) = \frac{\sum_{u \in U_i \cap U_j} S_{ui} \cdot S_{uj}}{\sqrt{\sum_{u \in U_i \cap U_j} {S_{ui}}^2} \cdot \sqrt{\sum_{u \in U_i \cap U_j} {S_{uj}}^2}}$$

用户u对项目t的预测打分:

$$\hat{r}_{ut} = \frac{\sum_{j \in Q_t(u)} Sim(j,t) \cdot r_{uj}}{\sum_{j \in Q_t(u)} |Sim(j,t)|}$$

Neighborhood-Based的优缺点

- 优点：直观容易理解，便于开发和调试。
- 缺点：如果用户和项目的数量太大时线下计算不是很高效-user − based复杂度$O(m^2n)$
  - 使用k-means来进行线下计算-找出与目标项目最接近的目标，使用簇中的top-k用户/项目来计算预测打分。
  - 使用k-means只用计算簇内的相似度，从而提升效率（准确率也会下降）。

Model Based Recommender System

Model based推荐系统

- 训练，预测效率高
- 避免overfitting

降维(Dimensionality Reduction)

- 稀疏的打分矩阵不便于计算相似度
- 降为低维的矩阵-潜在因子(Latent Factor)
- 得到低维的完整表示后可以将低维矩阵反转从而得到完整的打分矩阵

矩阵分解(Matrix Factorization)

对于一个mxn的矩阵R，当$k \ll \{m, n\}$时，总能用一个mxk的矩阵U和一个nxk的矩阵V表示为：

$$R = UV^T$$

即便当矩阵R的阶数比k大，也可以近似的表示为：

$$R \approx UV^T$$

此时剩余的矩阵可以表示为：

$$R - UV^T$$

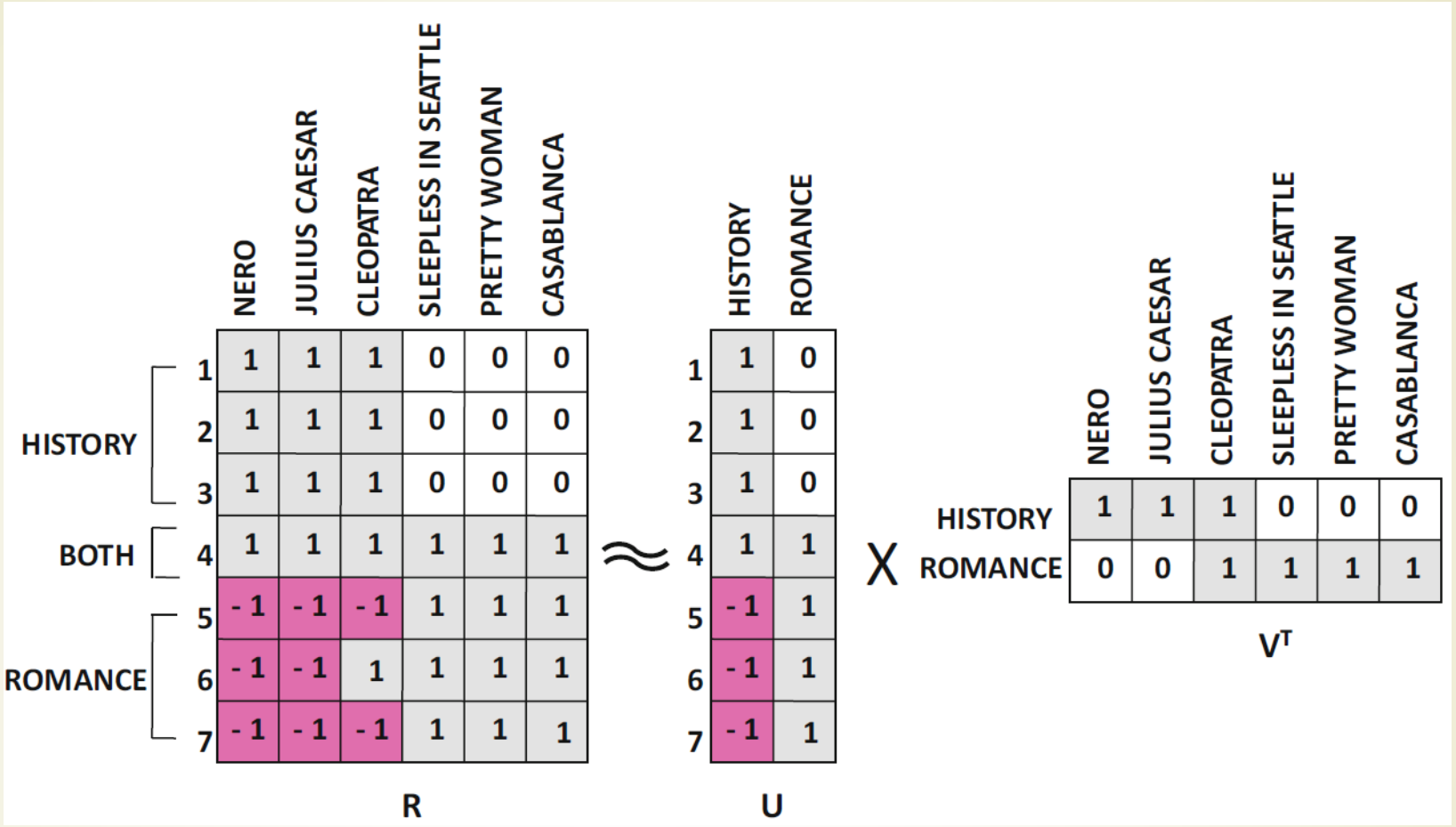
而造成的错误可以使用Frobenius范数表示为：

$$||\mathrm{R} - UV^T||^2$$

矩阵R中的每一个打分表示为：r_{ij}

矩阵U第i行-用户因子(user factor)表示为：$\bar{\mathrm{u}}_i$

矩阵V^T第j列- 项目因子(item factor)表示为：$\bar{\mathrm{v}}_j$



如图中例子所示：

$$r_{ij} \approx \sum_{s=1}^{k} u_{is} \cdot v_{js}$$ 其中s为电影的类型

$$r_{ij} = \sum_{s=1}^{k} (\text{用户i对历史类型的电影的喜好程度}) * (\text{项目j是否属于历史类型}) + (\text{用户i对浪漫类型的电影的喜好程度}) * (\text{项目j是否属于浪漫类型})$$

奇异值分解（Singular Value Decomposition-SVD）

目标: 将mxn的打分矩阵R转化为mxd的矩阵R′，where d ≪ n

1. 将矩阵R缺失的值用对应列的均值填充，得到mxn的矩阵R_f
2. 将矩阵R_f与它的转置矩阵相乘得到nxn的项目相似矩阵:

$$S = R_f^T R_f$$

3. 将矩阵S对角化:

$$S = P\Delta P^T$$

此时nxn的P矩阵的列是S的正交特征向量，对角矩阵Δ对角上的值为S的奇异值

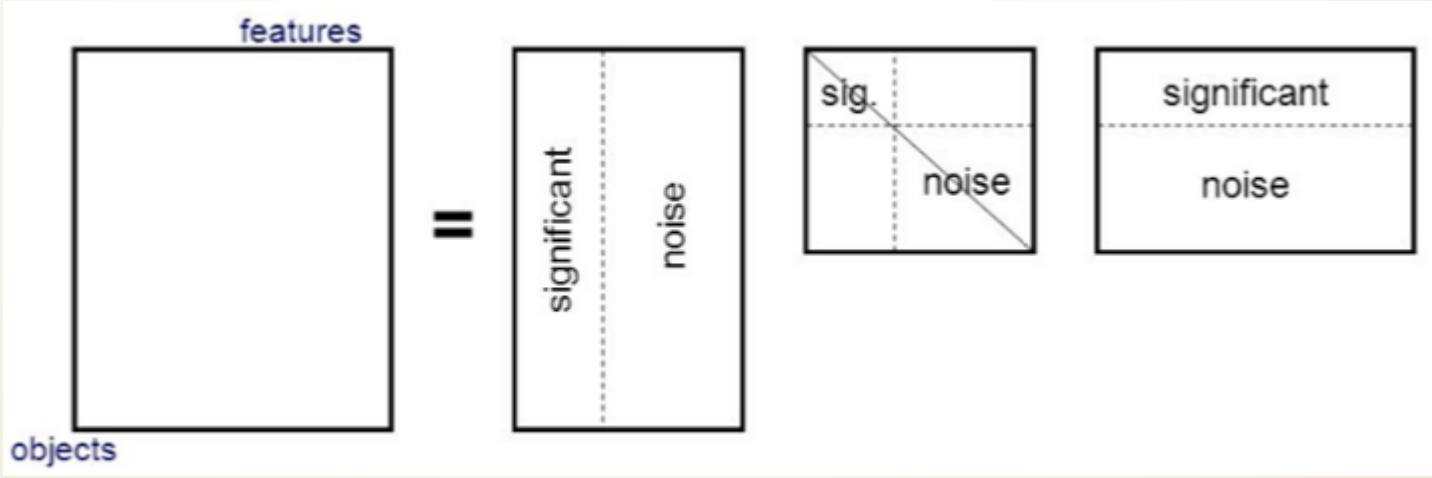
4. 选取最大的d个奇异值对应的nxd的矩阵P_d

此时可以得到低维的mxd的矩阵R′:

$$R' = R_f P_d$$

使用特征分解(Eigen decomposition)进行奇异值分解

- 之前的奇异值分解中的矩阵对角化限制矩阵U和矩阵V是正交的，潜在因子难以直观理解，SVD同样可以使用特征分解的方法来解决
- 对于一个mxn的打分矩阵R，当$k \ll \{m, n\}$时：
- $R \approx Q_k \sum_k P_k^T$，此时$Q_k$, $\sum_k$, P_k^T三个矩阵的维度分别为mxk, kxk, nxk



根据之前矩阵分解的例子，将用户因子和项目因子重新表示为：

$$\mathrm{U} = Q_k \sum_k$$

$$V = P_k$$

$$R = UV^T$$

使用无限制的矩阵分解对SVD进行优化

优化目标为：

$$Min\ J = \frac{1}{2}\|\mathrm{R} - UV^T\|^2$$

然而此时矩阵R依然是稀疏的，因此将目标转为使用机器学习的方法直接''学习''出UV^T

将R中可观察的打分r_{ij}的集合定义为：

$$S = \{(i,j)\}$$

预测的打分为：

$$\hat{r}_{ij} = \sum_{s=1}^{k} u_{is} \cdot v_{js}$$

预测打分和真实打分的误差为：

$$e_{ij} = r_{ij} - \sum_{s=1}^{k} u_{is} \cdot v_{js}$$

此时需要优化：

$$Min\ J = \frac{1}{2} \sum_{(i,j) \in S} {e_{ij}}^2 = \frac{1}{2} \sum_{(i,j) \in S} (r_{ij} - \sum_{s=1}^{k} u_{is} \cdot v_{js})^2$$

求导

$$\frac{\partial}{\partial \mathrm{u}_{iq}} = \eta \sum_{j:(i,j)\epsilon S} \left(r_{ij} - \sum_{s=1}^{k} u_{is} \cdot v_{js} \right) (-v_{jq}) = \eta \sum_{j:(i,j)\epsilon S} (e_{ij})(-v_{jq})$$

$$\forall i \epsilon \{1 \dots m\}, q \epsilon \{1 \dots k\}$$

$$\frac{\partial}{\partial v_{jq}} = \eta \sum_{i:(i,j)\epsilon S} \left(r_{ij} - \sum_{s=1}^{k} u_{is} \cdot v_{js} \right) (-u_{iq}) = \eta \sum_{i:(i,j)\epsilon S} (e_{ij})(-u_{iq})$$

$$\forall i \epsilon \{1 \dots m\}, q \epsilon \{1 \dots k\}$$

Content-Based Recommender System

Content-Based Recommender System

- CF使用打分之间的关联来推荐
- Content-based推荐系统使用有关item的描述信息来推荐
  - 电影的类型（如喜剧，悬疑等）
  - 用户的资料信息中的喜好

基于内容的推荐步骤

预处理和特征提取

- 获取描述性的特征来表示项目
- 如：
  - 电影：电影名，演员，台词，类型，简介
  - 网站：新闻标题，网页名，网页内容，cookies
  - 音乐：即兴重复段，和谐旋律，击鼓节拍

学习用户资料信息与过滤

- 学习用户资料信息与分类和回归模型相似
  - 离散打分（赞👍/踩👎）：分类
  - 数值打分（1-5）：线性回归
- D_L: 训练文档集，由一个特定用户标记
- D_U: 测试文档集，无标记，和该特定用户相关，该用户还没有看过或打分过

- 方法:
  - K Nearest Neighbors
  - Bayes Classifier

Content-Based vs Collaborative Filtering

- 优点：
  - CF对于新用户和新项目有冷启动(cold-start)问题
  - Contented Based可以处理新项目
  - 线下文本分类，节省资源
- 缺点：
  - 容易过于专业，推荐缺少新颖性，惊喜度