法律声明

- 课程详情请咨询
  - 微信公众号：北风教育
  - 官方网址：http://www.ibeifeng.com/

人工智能之推荐系统

推荐系统：推荐算法模型之协同过滤算法

主讲人：Gerry

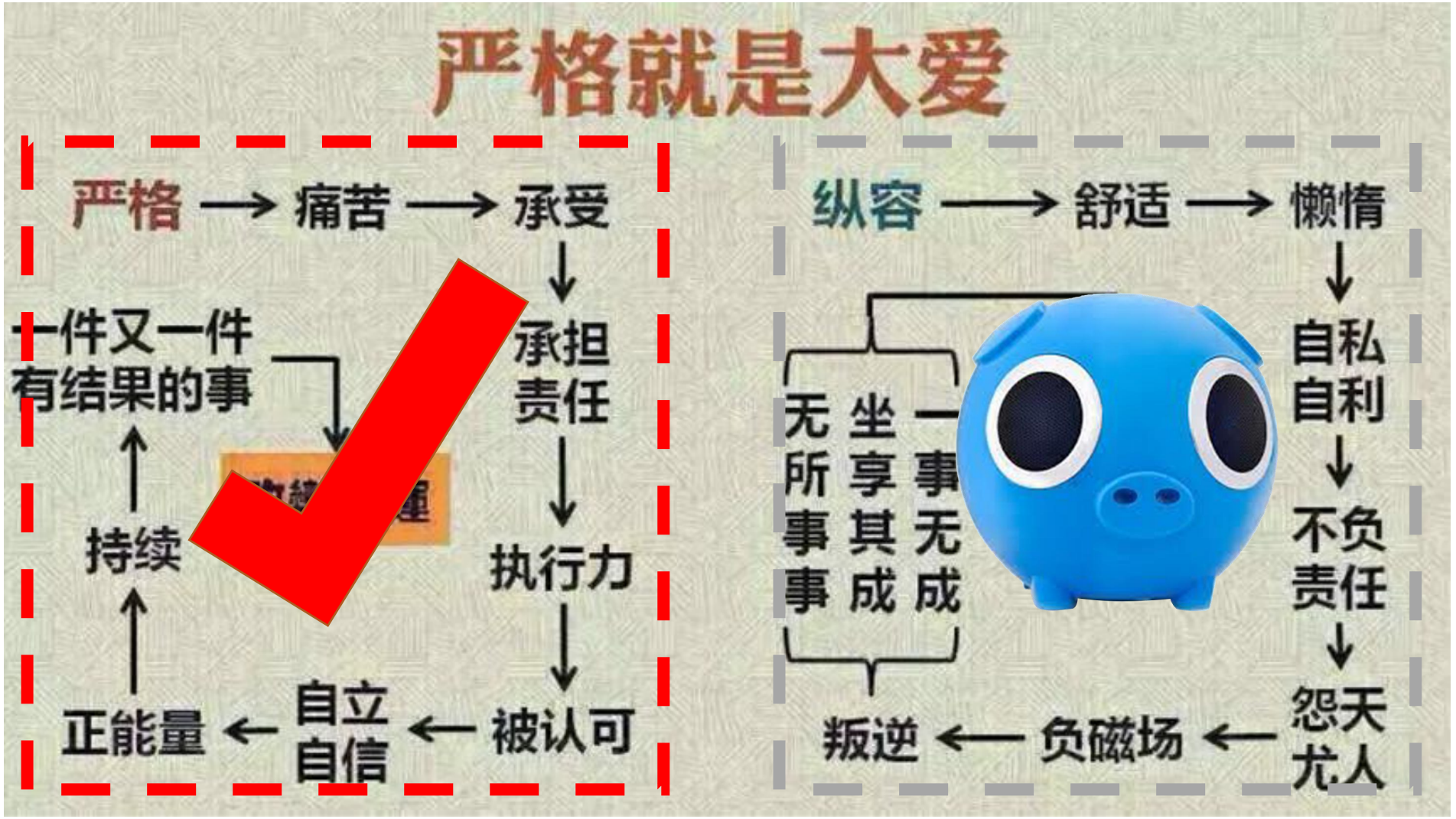
上海育创网络科技有限公司

北风网 IBEIFENG.COM

课程要求

- 课上课下“九字”真言
  - 认真听，善摘录，勤思考
  - **多温故，乐实践**，再发散
- 四不原则
  - 不懒散惰性，不迟到早退
  - 不请假旷课，不拖延作业
- 一点注意事项
  - 违反“四不原则”，不包就业和推荐就业

严格是大爱


严格就是大爱
严格
痛苦
承受
承担责任
执行力
被认可
自立自信
正能量
持续
一件又一件有结果的事
纵容
舒适
懒惰
自私自利
不负责任
怨天尤人
负磁场
叛逆
一事无成
坐享其成
无所事事

寄语

做别人不愿做的事，

做别人不敢做的事，

做别人做不到的事。

课程内容

- 基于统计的推荐算法模型
- 基于用户的协同过滤算法模型
- 基于物品的协同过滤算法模型

基于统计的推荐算法

$$f(r_{ui}) = \frac{1}{\sqrt{2\pi}\sigma_u} e^{-\frac{(r_{ui}-\mu_u)^2}{2\sigma_u^2}}$$

- Normal Predictor是一种假定用户对物品的评分数据是服从**正态分布**的，从而可以基于正态分布的期望μ和标准差σ随机的给出当前用户对于其它物品的评分。
- 基于大数定理或者使用最大似然估计(MLE)我们可以得到每个用户评分数据所属正态分布对应的期望μ和标准差σ分别为：

$$\mu_u = \frac{1}{|R_{train}|} \sum_{r_{ui} \in R_{train}} r_{ui} \qquad \sigma_u^2 = \frac{1}{|R_{train}|} \sum_{r_{ui} \in R_{train}} (r_{ui} - \mu_u)^2$$

- 案例：使用Normal Predictor实现对MovieLens的数据进行推荐模型的构建。

基于统计的推荐算法--Baseline Only

- 基线的含义其实是相对于平均值的偏差；假定现在需要使用Baseline Only算法给出Gerry对于《红海行动》这部电影的评分；比如我们现在计算出所有电影的评分均值为3.7分，此外《红海行动》这部电影比其它电影更好，所以它的评分常常比别的电影评分多0.5分，在另一方面，Gerry用户是一个要求比较严格的用户，通常会比别人少给0.3个评分。因此我们最终就可以计算出Gerry用户对于红海行动这部电影的评分为:3.7+0.5-0.3=3.9分。

基于统计的推荐算法--Baseline Only

- Baseline Only是一种基于统计的数据的**基线**评分预测算法，原理是认为每个用户对于每个商品的评分是有三部分构成的，即所有评分的均值μ、当前用户的评分基线b_u、当前物品的评分基线b_i；即评分公式如下：

$$\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i$$

- 在当前情况下，模型中的基线评分我们可以通过最小二乘的方式求解出来：

$$J(b_u, b_i) = \sum_{r_{ui} \in R_{train}} (r_{ui} - \mu - b_u - b_i)^2 + \lambda \left(\sum_u b_u^2 + \sum_i b_i^2 \right)$$

$$b_u, b_i = \min_{b_*} J(b_u, b_i)$$

基于统计的推荐算法--Baseline Only

- 对于对基线的求解，我们可以通过最小二乘的方式来求解，也可以通过梯度下降的方式来求解，同时我们也可以通过一种简单的估计的方式来求解，公式如下，根据经验推荐公式中的$\lambda_2=25$，$\lambda_3=10$；

$$b_i = \frac{\sum_{r_{ui} \in R_{train}} (r_{ui} - \mu)}{\lambda_2 + |\{u | r_{ui} \in R_{train}\}|} \qquad b_u = \frac{\sum_{r_{ui} \in R_{train}} (r_{ui} - \mu - b_i)}{\lambda_3 + |\{i | r_{ui} \in R_{train}\}|}$$

基于统计的推荐算法--Baseline Only

- 使用Baseline Only的方式对MovieLens的数据进行推荐模型的构建。

```
print('Using ALS')
bsl_options = {'method': 'als',
               'n_epochs': 5,
               'reg_u': 12,
               'reg_i': 5
               }
algo = BaselineOnly(bsl_options=bsl_options)


print('Using SGD')
bsl_options = {'method': 'sgd',
               'learning_rate': .00005,
               }
algo = BaselineOnly(bsl_options=bsl_options)
```

Collaborative Filtering

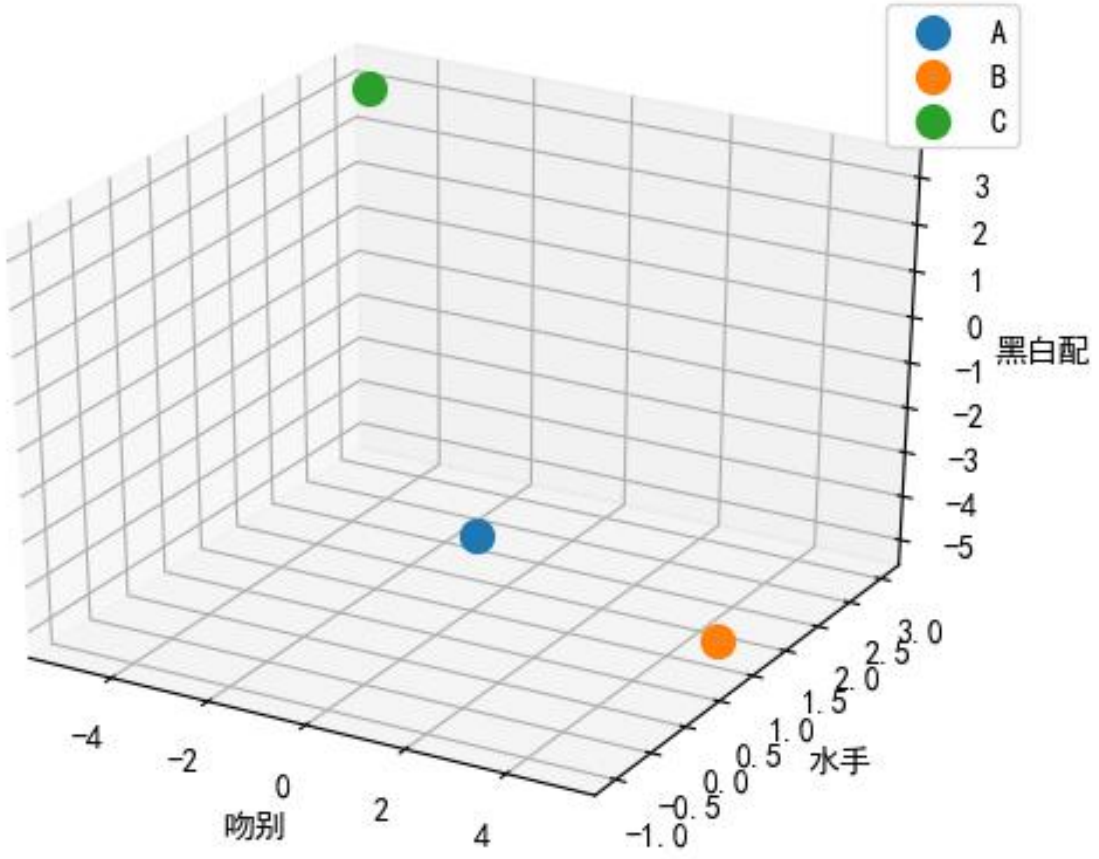
- **协同过滤**(CF, Collaborative Filtering)也叫做基于近邻的推荐算法，主要思想是：利用已有的用户群过去的行为或者意见预测数据，根据和当前用户/当前物品比较相似的近邻数据来产生推荐结果，和KNN算法的思想比较类似。主要应用场景是在线零售系统，目的是进行商品促销和提高销售额。
- 算法输入是一个**用户-物品评分矩阵**，输出的数据一般有两类：当前用户对物品喜欢和不喜欢程度的预测数值和n项的推荐物品的列表(不包含当前用户已经购买过的物品)
- 主要/最基础的实现方式有：
  - **基于用户的最近邻推荐**
  - **基于物品的最近邻推荐**

相似度度量(similarity)

- 假设有三首歌，分别是：《吻别》、《水手》、《黑白配》
  - A：收藏了《吻别》，而遇到《水手》、《黑白配》的时候总是跳过；
  - B：经常单曲循环《吻别》，对于《水手》经常听那么一两次，而《黑白配》基本上拉黑；
  - C：直接拉黑《吻别》，而收藏了《水手》、《黑白配》。
- 可以非常明显的看出来：A、B两个人的品味比较接近；而C和他们不太一样；换句话来讲，我们可以认为A、B比较相似，而C和他们不太相似。那么如何量化这个相似度呢？？现在假定对喜欢程度做一个度量(eg: 单曲循环=5, 分享=4, 收藏=3, 主动播放=2，听完=1, 跳过=-1，拉黑=-5)，那么此时对每个人来讲，每个人的喜好都是一个向量，即：A=(3,-1,-1)，B=(5,1,-5)，C=(-5,3,3)，我们可以使用这些点之间的距离作为我们最终的度量公式。

相似度度量(similarity)

- A=(3,-1,-1)，B=(5,1,-5)，C=(-5,3,3)



相似度度量(similarity)

- 欧几里得距离(MSD：平均平方距离相似度)：

$$dist(X,Y)=\sqrt{\sum_{i=1}^{n}|x_i-y_i|^2}$$

$$msd(u,v)=\frac{1}{|I_{uv}|}\sum_{i\in I_{uv}}(r_{ui}-r_{vi})^2$$

$$msd(i,j)=\frac{1}{|U_{ij}|}\sum_{u\in U_{ij}}(r_{ui}-r_{uj})^2$$

$$msd\_sim(u,v)=\frac{1}{msd(u,v)+1}$$

$$msd\_sim(i,j)=\frac{1}{msd(i,j)+1}$$

相似度度量(similarity)

- 余弦相似度(cosine similarity):

$$a = (x_{11}, x_{12}, \ldots, x_{1n}), b = (x_{21}, x_{22}, \ldots, x_{2n})$$

$$\cos(\theta) = \frac{\sum_{k=1}^{n} x_{1k} x_{2k}}{\sqrt{\sum_{k=1}^{n} x_{1k}^2} * \sqrt{\sum_{k=1}^{n} x_{2k}^2}}$$

$$\text{cosine\_sim}(u,v) = \frac{\sum_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum_{i \in I_{uv}} r_{vi}^2}}$$

$$\text{cosine\_sim}(i,j) = \frac{\sum_{u \in U_{ij}} r_{ui} \cdot r_{uj}}{\sqrt{\sum_{u \in U_{ij}} r_{ui}^2} \cdot \sqrt{\sum_{u \in U_{ij}} r_{uj}^2}}$$

相似度度量(similarity)

- 改进余弦相似度(Adjuested Cosine Similarity):

$$sim(u,v)=\frac{\sum_{i\in I_{uv}}\left(r_{u,i}-\bar{r}_i\right)\left(r_{v,i}-\bar{r}_i\right)}{\sqrt{\sum_{i\in I_{uv}}\left(r_{u,i}-\bar{r}_i\right)^2}\sqrt{\sum_{i\in I_{uv}}\left(r_{v,i}-\bar{r}_i\right)^2}}$$

$$sim(i,j)=\frac{\sum_{u\in U_{ij}}\left(r_{u,i}-\bar{r}_u\right)\left(r_{u,j}-\bar{r}_u\right)}{\sqrt{\sum_{u\in U_{ij}}\left(r_{u,i}-\bar{r}_u\right)^2}\sqrt{\sum_{u\in U_{ij}}\left(r_{u,j}-\bar{r}_j\right)^2}}$$

相似度度量(similarity)

- 使用**皮尔森/皮尔逊相关系数**(**Pearson Correlation Coefficient**)来表示两个用户之间的相关性；取值范围为[-1,+1]，-1表示强负相关，+1表示强正相关，0表示不相关。

$$per\_sim_{u,v} = \frac{\sum_{i \in I_{u,v}} \left(r_{u,i} - \bar{r}_u\right)\left(r_{v,i} - \bar{r}_v\right)}{\sqrt{\sum_{i \in I_{u,v}} \left(r_{u,i} - \bar{r}_u\right)^2} \sqrt{\sum_{i \in I_{u,v}} \left(r_{v,i} - \bar{r}_v\right)^2}}$$

- Pearson相关系数的应用要求两个变量的标准差都不为零的时候，该相关系数才具有定义，使用场景如下：
  - 两个变量之间是线性关系，都是连续数据
  - 两个变量的总体是正态分布或者解决正态的单峰分布
  - 两个变量的观察值是成对的，每对观测值之间是相互对立的

Pearson相关系数

协方差可以描述X和Y的相关程度，但是协方差的值和X/Y的值采用的是不同的量纲，导致协方差在数值上表现出比较大的差异，因此可以引入相关系数来表示X和Y的相关性。

$$\rho(X,Y)=\frac{Cov(X,Y)}{\sqrt{D(X)}\cdot\sqrt{D(Y)}}$$

当p(X,Y)=0的时候，称X和Y不线性相关。

Pearson相关系数取值范围为[-1,1]

绝对值范围	含义
0.8-1.0	极强相关
0.6-0.8	强相关
0.4-0.6	中等程度相关
0.2-0.4	弱相关
0-0.2	极弱相关或无相关

相似度度量(similarity)

- 杰卡德相似系数(Jaccard Similarity):

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

$$jac\_sim(u,v)=\frac{|I_{uv}|}{|I_u|+|I_v|-|I_{uv}|}$$

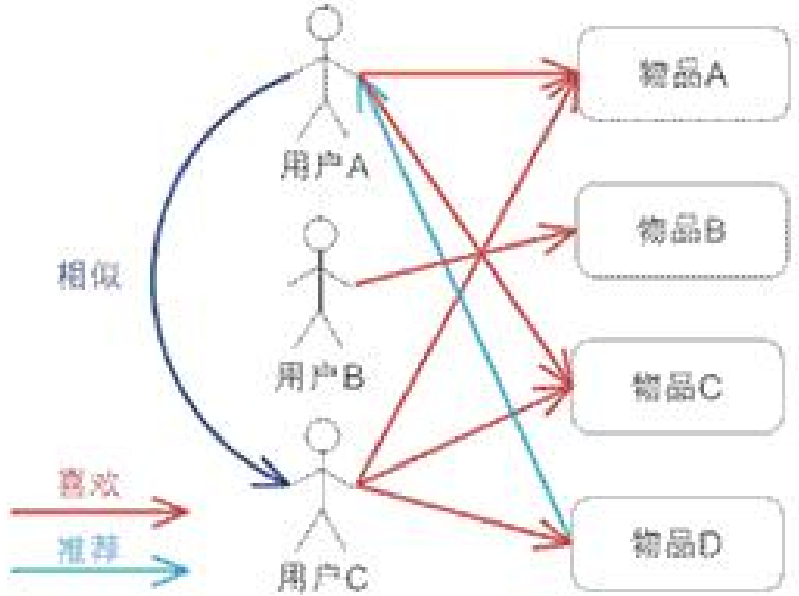
$$jac\_sim(i,j)=\frac{|U_{ij}|}{|U_i|+|U_j|-|U_{ij}|}$$

UserCF

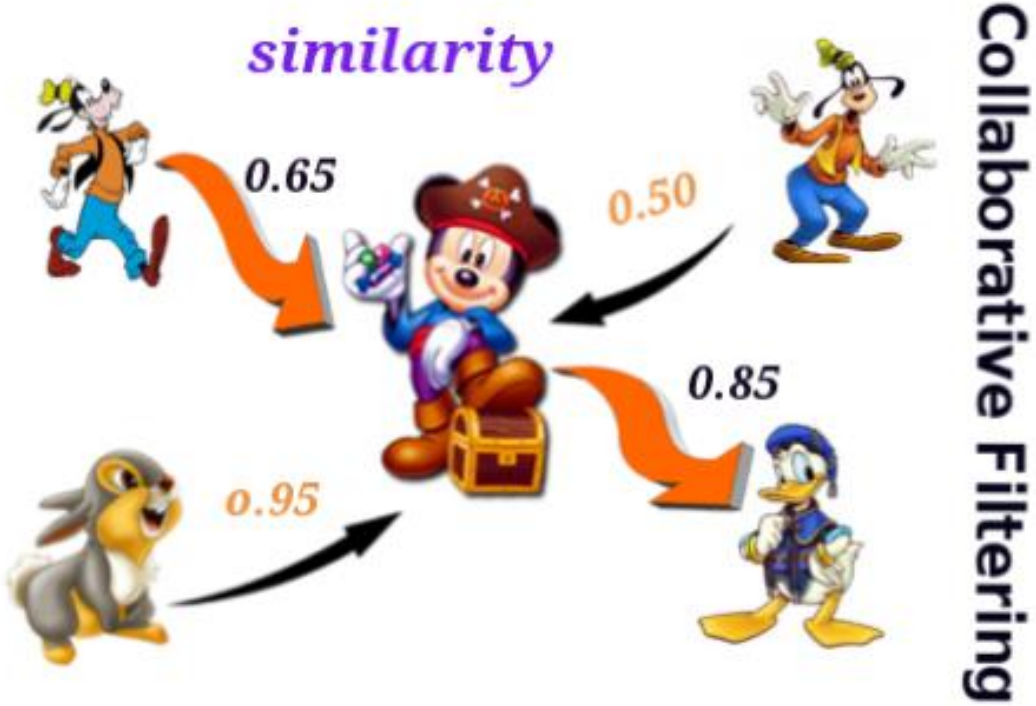
- **基于用户的最近邻推荐(user-based nearest neighbor recommendation, UserCF)**主要思想是：首先，对输入的评分数据集和当前用户ID作为输入，找出与当前用户过去有相似偏好的其它用户，这些用户叫做**对等用户**或者**最近邻**；然后，对当前用户没有见过的每个产品p，利用用户的近邻对产品p的评分进行预测；最后，选择所有产品评分最高的TopN个产品推荐给当前用户
- 前提/假设：
  - 如果用户过去有相似的偏好，那么该用户在未来也会有相似的偏好
  - 用户的偏好不会随着时间而变化

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	√		√	√



UserCF



Collaborative Filtering

UserCF

- UserCF的算法执行流程：
  - 获取和当前用户最相似的K个近邻用户(要求K个近邻用户必须都对物品i有评分)；
  - 根据这K个近邻用户计算当前用户对物品i的评分；
  - 重复上述两个操作，计算出当前用户对所有物品的评分；
  - 重复上述三个操作，计算所有用户对所有物品的评分；
  - 对于每个用户，将该用户对所有物品的评分按照从大到小进行排序，获取评分值最大的前N个物品作为当前用户的推荐结果。
  - 备注：一般情况下，需要进行去重，当用户已经查看过该商品后，那么该商品就不应该出现在推荐列表中。

UserCF

- 最原始的UserCF：首先获取出K个最相似的近邻用户，然后将这些用户对物品i的评分进行加权求和。

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} sim(u,v) * r_{vi}}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

UserCF

- 进行均值转换后的UserCF：该算法的原理是认为用户对物品的评分应该是位于该用户对所有物品评分的均值附近的，所以在计算过程中，不是直接使用相似用户对物品i的评分，而是使用物品评分和期望之间的差值进行计算。

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in N_i^k(u)} sim(u,v) * (r_{vi} - \bar{r}_v)}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

UserCF

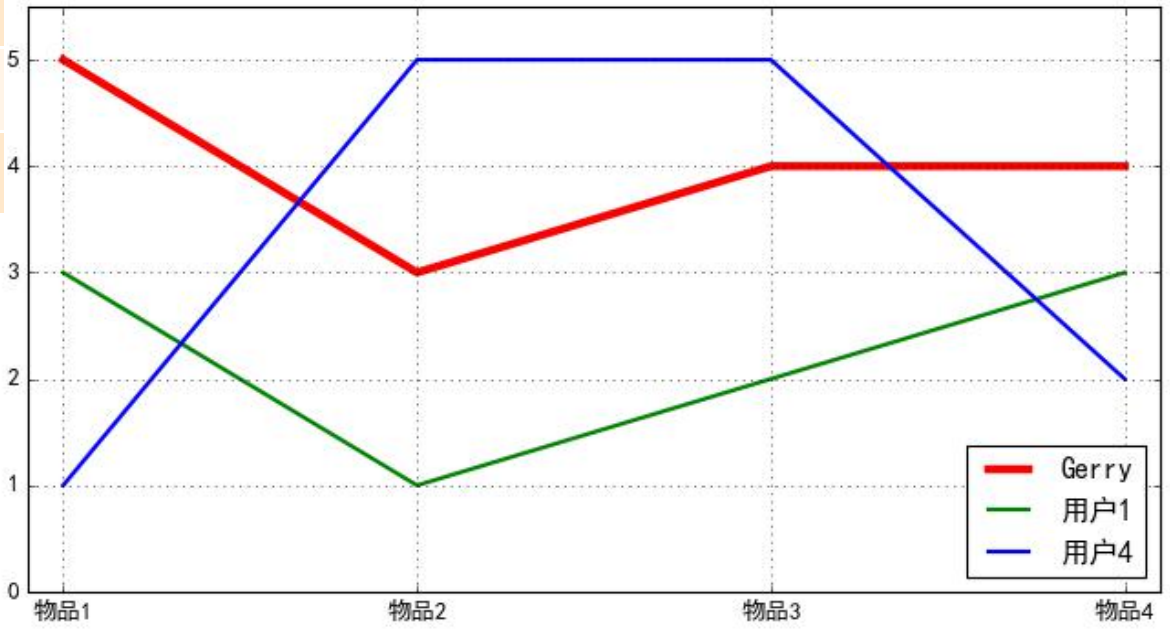
- 在进行基线转换后的UserCF算法中，使用baseline的值来代替均值即可。因为均值体现的是当前用户在所有物品中的评分均值，而baseline可以认为是当前用户在当前物品上可能的评分，通过计算相似用户实际评分和baseline可能评分之间的差值从而可以得到当前用户的预测评分(预测评分=baseline+可能的差值)

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in N_i^k(u)} sim(u,v) * (r_{vi} - b_{vi})}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

UserCF

- 基于评分数据矩阵预测Gerry用户对于物品5的评分

	物品1	物品2	物品3	物品4	物品5
Gerry	5	3	4	4	?
用户1	3	1	2	3	3
用户2	4	3	4	3	5
用户3	3	3	3	5	4
用户4	1	5	2	4	1



UserCF

- 均值调整后的评分矩阵

	物品1	物品2	物品3	物品4	物品5	均值
Gerry	1	-1	0	0	?	4
用户1	0.6	-1.4	-0.4	0.6	0.6	2.4
用户2	0.2	-0.8	0.2	-0.8	1.2	3.8
用户3	-0.6	-0.6	-0.6	1.4	0.4	3.6
用户4	-1.6	2.4	-0.6	1.4	-1.6	2.6

UserCF

- 用户Gerry和其它用户之间的相似度列表

	用户1	用户2	用户3	用户4
Gerry	0.839	0.707	0.0	-0.894

$$\bar{r}_{gerry} = \bar{r}_a = 4$$

$$\bar{r}_{User1} = \bar{r}_b = 2.4$$

$$p_{gerry,User1} = \frac{(5-\bar{r}_a)*(3-\bar{r}_b)+(3-\bar{r}_a)*(1-\bar{r}_b)+(4-\bar{r}_a)*(2-\bar{r}_b)+(4-\bar{r}_a)*(3-\bar{r}_b)}{\sqrt{(5-\bar{r}_a)^2+(3-\bar{r}_a)^2+(4-\bar{r}_a)^2+(4-\bar{r}_a)^2}*\sqrt{(3-\bar{r}_b)^2+(1-\bar{r}_b)^2+(2-\bar{r}_b)^2+(3-\bar{r}_b)^2}} = 0.839$$

- 选择N为2，那么可以得到Gerry对物品5的评分预测为:4.872

$$pred(gerry, product5) = 4 + \frac{0.853*(3-2.4)+0.707*(5-3.8)}{0.853+0.707} = 4.872$$

UserCF案例

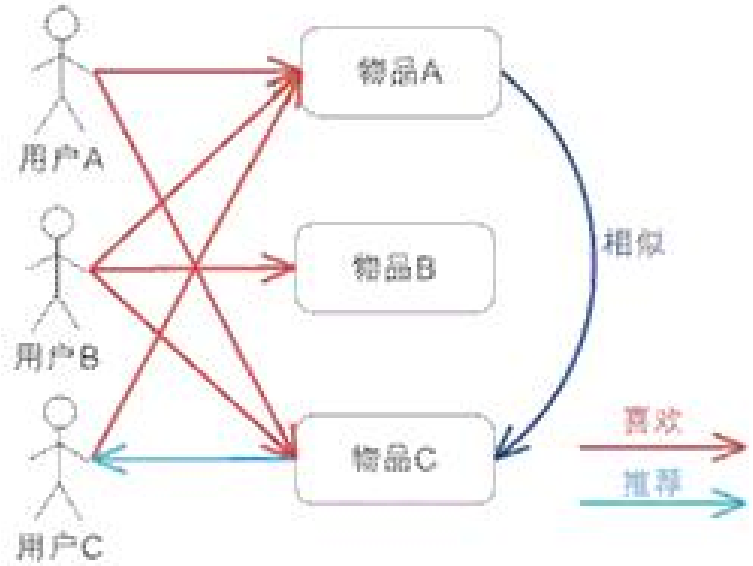
- 分别使用KNNBasic、KNNWithMeans、KNNBaseline三个API对电影数据进行基于用户的协同过滤推荐，并比较这三种API在UserCF推荐中的效果。

ItemCF

- **基于物品的最近邻推荐(item-based nearest neighbor recommendation，ItemCF)**的思想是基于物品之间的相似度，而不是基于用户之间的相似度来进行预测评分值。
- 在基于物品的最近邻推荐算法中使用**余弦相似度**来计算两个物品中间的相似度的值比较多，相似度取值范围:[0,1]，值越接近1表示越相似。另外由于余弦相似度不能体现用户评分之间的的差异性，所以一般使用**改进余弦相似度**比较多。
- ItemCF和UserCF相比，在原理上来讲，只有一个区别：UserCF计算的是用户之间的相似度，从而是将相似用户喜好的物品推荐给当前用户；ItemCF中计算的是物品与物品之间的相似度，从而是根据当前用户喜好的物品来推荐其它物品列表。

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



ItemCF

- ItemCF的算法执行流程：
  - 获取和当前物品最相似的K个近邻物品(要求K个近邻物品必须都被用户u评分过)；
  - 根据这K个近邻物品计算用户u对当前物品的评分；
  - 重复上述两个操作，计算出所有用户对当前物品的评分；
  - 重复上述三个操作，计算所有用户对所有物品的评分；
  - 对于每个用户，将该用户对所有物品的评分按照从大到小进行排序，获取评分值最大的前N个物品作为当前用户的推荐结果。
  - 备注：一般情况下，需要进行去重，当用户已经查看过该商品后，那么该商品就不应该出现在推荐列表中。

ItemCF

- 最原始的ItemCF：首先获取出K个最相似的近邻物品，然后将当前用户在这些物品上的评分进行加权求和。

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u^k(i)} sim(i,j) * r_{uj}}{\sum_{j \in N_u^k(i)} sim(i,j)}$$

ItemCF

- 进行均值转换后的ItemCF：该算法的原理是认为用户u对物品i的评分应该是位于该物品i被其它所有用户评分的均值附近的。所以在计算过程中，不是直接使用用户u对相似物品的评分，而是使用物品评分和期望之间的差值进行计算。

$$\hat{r}_{ui} = \bar{r}_i + \frac{\sum_{j \in N_u^k(i)} sim(i,j) * (r_{uj} - \bar{r}_j)}{\sum_{j \in N_u^k(i)} sim(i,j)}$$

ItemCF

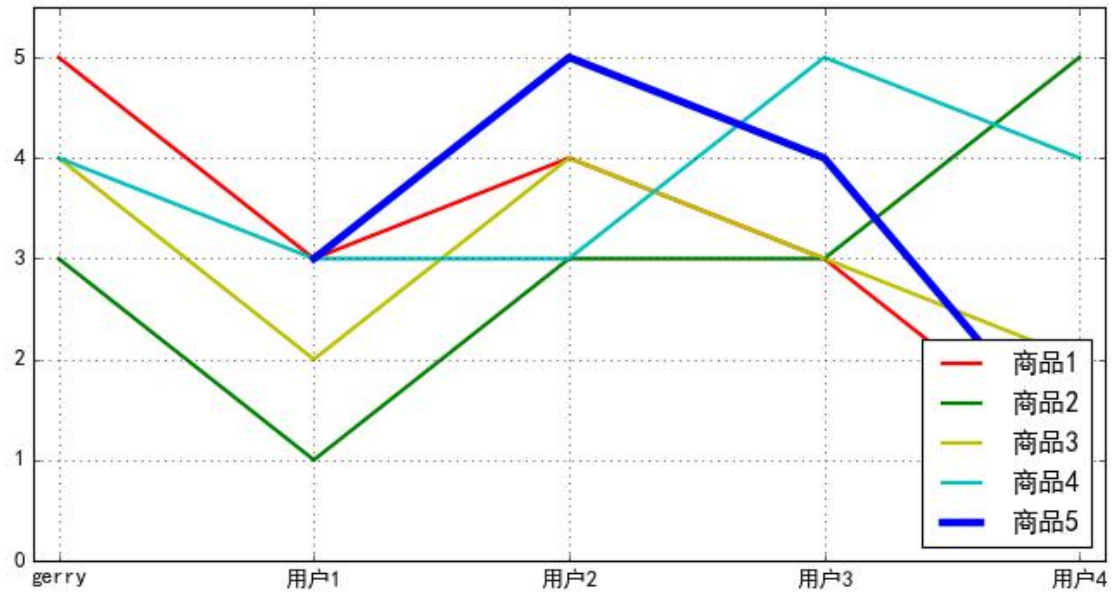
- 在进行基线转换后的ItemCF算法中，使用baseline的值来代替均值即可。因为均值体现的是所有用户在当前物品i中的评分均值，而baseline可以认为是当前用户u在当前物品i上可能的评分，通过计算用户u在相似物品j上的实际评分和baseline可能评分之间的差值从而可以得到用户u的对于物品i预测评分(预测评分=baseline+可能的差值)

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in N_u^k(i)} sim(i,j) * (r_{uj} - b_{uj})}{\sum_{j \in N_u^k(i)} sim(i,j)}$$

ItemCF

- 基于评分数据矩阵预测Gerry用户对于物品5的评分

	物品1	物品2	物品3	物品4	物品5
Gerry	5	3	4	4	?
用户1	3	1	2	3	3
用户2	4	3	4	3	5
用户3	3	3	3	5	4
用户4	1	5	2	4	1



ItemCF

- 均值调整后的评分矩阵

	物品1	物品2	物品3	物品4	物品5	均值
Gerry	1	-1	0	0	?	4
用户1	0.6	-1.4	-0.4	0.6	0.6	2.4
用户2	0.2	-0.8	0.2	-0.8	1.2	3.8
用户3	-0.6	-0.6	-0.6	1.4	0.4	3.6
用户4	-1.6	2.4	-0.6	1.4	-1.6	2.6

ItemCF

- 计算得到物品5和其它物品之间的相似度为：

	物品1	物品2	物品3	物品4
物品5	0.754	-0.937	0.353	-0.483

$$p_{item5,item1}=\frac{0.6*0.6+0.2*1.2+(-0.6)*0.4+(-1.6)*(-1.6)}{\sqrt{0.6^2+0.2^2+(-0.6)^2+(-1.6)^2}*\sqrt{0.6^2+1.2^2+0.4^2+(-1.6)^2}}=0.754$$

- 选择N为2，相似度阈值为0.3，那么可以得到Gerry对物品5的评分预测为:4.68

$$pred(gerry,item5)=\frac{0.754*5+0.353*4}{0.754+0.353}=4.68$$

ItemCF

- 基于物品的最近邻推荐可以离线进行数据预计算，先构建出一个**物品相似度矩阵**，用于描述两个物品之间的相似度。在线上运行的时候，通过确定与物品p最相似的物品并计算u对这些邻近物品评分的加权总和来得到u对p的预测评分；近邻数量受限于用户有评分的物品个数，由于这样的物品数量一般都比较少，因此计算预测值的过程可以在线上交互应用允许的短时间内完成。
- 原则上来讲，这种离线的计算方式也适合于基于用户的最近邻推荐，但是实际情况下，两个用户评分重叠的情况非常少，这样的话就会导致其它的评分值影响到最终用户间的相似度的计算。换言之，物品之间的相似度比用户之间的相似度更加稳定，这种计算的方式不会改变物品之间的相似度。

ItemCF案例

- 分别使用KNNBasic、KNNWithMeans、KNNBaseline三个API对电影数据进行基于用户的协同过滤推荐，并比较这三种API在ItemCF推荐中的效果。

CF-优点

- 简单性：实现简单，而且在调整参数的过程中只有一个近邻数需要调整
- 合理性：对于预测推荐提供了简洁并直观的理由
- 高效性：基于近邻方法的推荐的效果特别高，因为可以先进行预处理，构建出相似度矩阵，在实际应用过程中，可以提供近似实时的推荐结果。
- 稳定性：当相似度矩阵构建完成后，如果有一个新的用户或者新的物品有评分的时候，只需要计算这个个体和其它个体之间的相似性即可完成更新操作。

CF-评分

- 由于在CF算法中，需要输入的是用户-物品评分矩阵，所以构建用户-物品评分矩阵是一个在进行协同过滤的重点。评分一般采用两分制、五分制、七分制和十分制这四种。可以通过以下两种方式收集用户对物品的评分，分别是：
  - **显式评分**：通过问卷调查的方式收集用户对于商品的评分，优点是数据比较准确，缺点是当用户看不到好处的时候可能不会提供评分。
  - **隐式评分**：当用户购买一个商品或者浏览一个商品的时候，我们可以认为这是一个正向评分/正向意图，根据既定的规则，可以将其转换为评分值；这种方式的优点是可以收集到最够多的数据，缺点是很难保证得到的评分数据是一个准确的用户评分数据。

CF-评分矩阵数据稀疏

- 在实际的应用中，由于用户一般只会评价(或者购买、浏览)少部分商品，这样就会导致评分矩阵一般比较稀疏。这种情况下的挑战就是使用相对较少的有效评分来得到一个比较准确的预测。最直接的方式就是利用用户/物品的额外特征属性数据，进行数据的分类、聚类操作，来求解相似用户/物品列表。除此之外，可以使用缺省投票的形式补全物品的评价分数，思想是给那些只有一两个用户评价的物品给定一个缺省值。

CF-冷启动问题

- 在CF算法中，存在着冷启动的问题，主要包括：
  - 如何向还没有任何物品评分的新用户进行推荐
  - 如何处理从未被评过分或者购买过的商品
- 解决方案：
  - 利用混合方法进行推荐，即利用额外的外部特征属性；对样本进行分类、聚类建模即可完成推荐
  - 结合基于用户的近邻推荐和基于物品的近邻推荐算法
  - 推荐Top10热门商品或者专门给定一个最近新加商品推荐列表

CF-近邻的选择

- 在基于近邻进行推荐的算法中，近邻数量的选择和选择近邻的规则对于推荐系统的质量会产生重要的影响，一般现在近邻可以通过以下方式来选择
  - 先过滤出预选近邻：Top-N过滤、阈值过滤、负数过滤
  - 在从预选近邻列表中获取k个近邻；如果k的数目过小，会导致预测精度非常低，随着k的增大，预测精度会有一定的提升，但是达到一定值的时候(>50)，由于存在一些重要的关联被不重要的关联所削弱，会导致预测精度下降，推荐选值一般为：25~50之间，当然在实际工作中，最优k的值一般需要通过进行交叉验证获取。

CF-基于近邻推荐算法的缺陷

- 基于近邻的算法是基于评分之间的关联性进行推荐的，所以存在两个重要的缺陷：
  - **覆盖有限**：由于计算两个用户之间的相似性是基于他们对相同物品的评分，而且只有对相同物品进行评分的用户才能作为近邻，但是在实际应用中，有些用户有很少或者没有共同评分，但是他们可能具有相似的爱好，所以推荐算法的覆盖将会受到影响。
  - **对稀疏数据的敏感**：由于用户只会对一部分物品进行评分，所以评分矩阵的稀疏性是大多数推荐系统的共同问题。当数据是稀疏的时候，两个用户或者物品之间的相似性计算仅适用很少量有限的近邻。另外，相似性权重的计算也可能依赖小部分评分，从而有可能导致推荐偏差。这也是一个比较重要的问题：**冷启动问题**。
- 解决方案：默认值填充、额外特性属性关联(非评分，内容信息填充)

CF总结

- 基于用户行为，不需要先验知识；只需要用户和物品关联的评分矩阵，结构简单；在用户行为比较丰富的时候，推荐效果不错。
- 需要根据用户行为(用户物品评分矩阵)，计算用户/物品相似度矩阵，一般采用离线计算相似度矩阵和用户物品评分矩阵的策略，在线模块只负责根据用户物品评分信息对用户产生推荐结果。
- CF基于的是用户对完全相同物品的行为，对于相似物品的行为是没法计算相似度的。

CF总结-UserCF和ItemCF比较

	UserCF	ItemCF
性能	适用于用户较少的场合，如果用户过多，计算用户的相似度矩阵代价比较高。	适用于物品数明显小于用户数的场合，如果物品数过多，计算物品之间的相似度矩阵代价过大。
领域	时效性较强，用户个性化兴趣不明显的领域	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，不一定会对推荐结果产生影响	用户有新行为的时候，一定会导致推荐结果实时变化
冷启动	在新用户对很少物品产生行为的时候，不能立即对用户进行个性化推荐 新物品上线一段时间后，一旦有用户对该物品产生行为，那么就可以将新物品推荐给其他兴趣相似的用户	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似的其它物品 必须等到更新物品相似度矩阵的时候才可以将新物品更新进去。
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做推荐解释，可以令用户比较信服

作业：

- 在Surprise中实现Jaccard相似度度量类，使用该相似度度量方式，分别使用UserCF和ItemCF对电影数据进行推荐模型构建，并比较两者的效果区别。

THANK YOU

上海育创网络科技有限公司