《推荐系统实践》学习笔记系列(二)

推荐系统

第二章

基于用户行为分析的推荐算法是个性化推荐系统的重要算法，学术界一般将这种类型的算法称为**协同过滤算法。**

2.1 用户行为数据简介

- 用户行为数据在网站上最简单的形式就是**日志**（row log、session log、impression log、click log）。
- 用户行为日志分类
  按反馈明确性分：
  显性反馈行为（explicit feedback）包括用户明确表示对物品喜好的行为（喜欢/不喜欢）。
  隐式反馈（implicit feedback）指的是那些不能明确反应用户喜好。（比如页面浏览行为）的行为

	显性反馈数据	隐性反馈数据
用户兴趣	明确	不明确
数量	较少	庞大
存储	数据库	分布式文件系统
实时读取	实时	有延迟
正负反馈	都有	只有正反馈

显性反馈数据和隐性反馈数据的比较

	显性反馈	隐性反馈
视频网站	用户对视频的评分	用户观看视频的日志、浏览视频页面的日志
电子商务网站	用户对商品的评分	购买日志、浏览日志
门户网站	用户对新闻的评分	阅读新闻的日志
音乐网站	用户对音乐/歌手/专辑的评分	听歌的日志

各代表网站中显性反馈数据和隐性反馈数据的例子

按反馈的方向分：

分为**正反馈**和**负反馈。**

- 互联网中的用户行为表示

user id	产生行为的用户的唯一标识
item id	产生行为的对象的唯一标识
behavior type	行为的种类（比如是购买还是浏览）
context	产生行为的上下文，包括时间和地点等
behavior weight	行为的权重（如果是观看视频的行为，那么这个权重可以是观看时长；如果是打分行为，这个权重可以是分数）
behavior content	行为的内容（如果是评论行为，那么就是评论的文本；如果是打标签的行为，就是标签）

用户行为的统一表示

- 目前比较有代表性的数据集

无上下文信息的隐性反馈数据集：每一条行为记录仅仅包含用户ID和物品ID。（Booking Crossing）

无上下文信息的显性反馈数据集：无上下文信息的显性反馈数据集:每一条记录包含用户ID、物品ID和用户对物品的评分。

有上下文信息的隐性反馈数据集：每一条记录包含用户ID、物品ID和用户对物品产生行为的时间戳。（Lastfm）

有上下文信息的显性反馈数据集：每一条记录包含用户ID、物品ID、用户对物品的评分和评分行为发生的时间戳。（Netflix Prize）

2.2 用户行为分析

- 用户活跃度和物品流行度的分布

令$f_u(k)$为对k个物品产生过行为的用户数，令$f_i(k)$为被k个用户产生过行为的物品数，那么$f_u(k)$和$f_i(k)$都满足长尾分布：

$$f_u(k) = \alpha_u k^{\beta_u}$$

$$f_i(k) = \alpha_i k^{\beta_i}$$

- 用户活跃度和物品流行度的关系

用户越活跃越趋向于冷门的物品。

仅仅基于用户行为数据的推荐算法一般称为协同过滤算法。

协同过滤算法分类：

基于邻域的方法（neighborhood-based）

隐语义模型（latent factor model）

♦**基于用户的协同过滤算法**：用户推荐和他兴趣相似的其他用户喜欢的物品。

♦**基于物品的协同过滤算法**：给用户推荐和他之前喜欢的物品相似的物品。

基于图的随机游走方法（random walk on graph）

2.4 基于邻域的算法

- 基于用户的协同过滤算法(UserCF)

1.算法步骤

（1）找到和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，且目标用户没有听说过的物品推荐给目标用户。

2.用户相似度

i.基本相似度

给定用户u和v，令$N(u)$表示用户u曾经有过正反馈的物品集合，令$N(v)$为用户v曾经有过正反馈的物品集合。

Jaccard公式：

$$w_{uv} = \frac{|N(u) \bigcap N(v)|}{|N(u) \bigcup N(v)|}$$

余弦相似度：

$$w_{uv} = \frac{|N(u) \bigcap N(v)|}{\sqrt{N(u)||N(v)}}$$

3.用户相似度的改进

两个用户对冷门物品采取过同样的行为更能说明他们兴趣的相似度。

User-IIF算法：

$$w_{uv} = \frac{\sum_{i \epsilon N(u) \bigcap N(v)} \frac{1}{log(1+|N(i)|)}}{\sqrt{N(u)||N(v)}}$$

4.基于用户的协同过滤算法的缺陷

随着网站的用户数目越来越大，计算用户兴趣相似度矩阵将越来越困难，其运算时间复杂度和空间复杂度的增长和用户数的增长近似于平方关系。
基于用户的协同过滤很难对推荐结果作出解释。

- 基于物品的协同过滤算法(ItemCF)

  1.算法步骤

  （1）计算物品之间的相似度。

  （2）根据物品的相似度和用户历史行为给用户生成推荐列表。

  2.物品相似度

$$w_{ij} = \frac{|N(i) \bigcap N(j)|}{|N(i)|}$$

上述公式存在的问题是如果物品j很热门，那么w_{ij}会很大，接近1，因此该公式会造成任何物品和热门物品有很大的相似度。

改进——惩罚了物品j的权重：

$$w_{ij} = \frac{|N(i) \bigcap N(j)|}{\sqrt{|N(i)||N(j)|}}$$

- UserCF和ItemCF的综合比较

	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
	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	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

$UserCF$和$ItemCF$优缺点的对比

UserCF的推荐更社会化，反映了用户所在的小型兴趣群体中物品的热门程度，而ItemCF的推荐更加个性化，反映了用户自己的兴趣传承。

- 哈利波特问题

回顾之前ItemCF物品相似度的经典公式：

$$w_{ij} = \frac{|N(i) \bigcap N(j)|}{\sqrt{|N(i)||N(j)|}}$$

如果j非常热门，那么上面公式的分子$|N(i) \bigcap N(j)|$就会越来越接近 $N(i)$ 。尽管上面的公式分母已经考虑到了j的流行度，但在实际应用中，热门的j仍然会获得比较大的相似度。

解决方案：

可以在分母上加大对热门物品的惩罚，比如采用如下公式：

$$w_{ij} = \frac{|N(i) \bigcap N(j)|}{\sqrt{|N(i)|^{1-\alpha}|N(j)|^{\alpha}}}$$

其中$\alpha \epsilon [0.5, 1]$。通过提高α就可以惩罚热门的j。

2.5 隐语义模型

1. 基础算法

隐语义模型是最近几年推荐系统领域最为热门的研究话题，它的核心思想是通过隐含特征(latent factor)联系用户兴趣和物品。

需要解决的问题

2. 如何给物品进行分类？

♦找编辑给物品分类（存在很多问题）

♦隐含语义分析技术因为采取基于用户行为统计的自动聚类，较好地解决了编辑分类存在的问题。

ii.如何确定用户对哪些类的物品感兴趣，以及感兴趣的程度？

iii.对于一个给定的类，选择哪些属于这个类的物品推荐给用户，以及如何确定这些物品在一个类中的权重？

3.隐含特征模型（LFM）

$$Preference(u,i) = r_{ui} = p_u^T q_i = \sum_{f=1}^{F} p_{u,k} q_{i,k}$$

这个公式中$p_{u,k}$和$q_{i,k}$是模型的参数，其中$p_{u,k}$度量了用户u的兴趣和第k个隐类的关系，而$q_{i,k}$度量了第k个隐类和物品i之间的关系。

要计算这两个参数，需要一个训练集，对于每个用户u，训练集里都包含了用户u喜欢的物品和不感兴趣的物品，通过学习这个数据集，就可以获得上面的模型参数。

3. 损失函数

$$C = \sum_{(u,i)\epsilon K}(r_{ui} - \hat{r_{ui}})^2 = \sum_{u,j\epsilon K}(r_{ui} - \sum_{k=1}^{K} p_{u,k} q_{i,k})^2 + \lambda||p_u||^2 + \lambda||q_i||^2$$

这里$\lambda||p_u||^2 + \lambda||q_i||^2$是用来防止过拟合正则化项，$\lambda$可以通过实验获得。要最小化上面的损失函数，可以利用一种称为随机梯度下降法的算法。

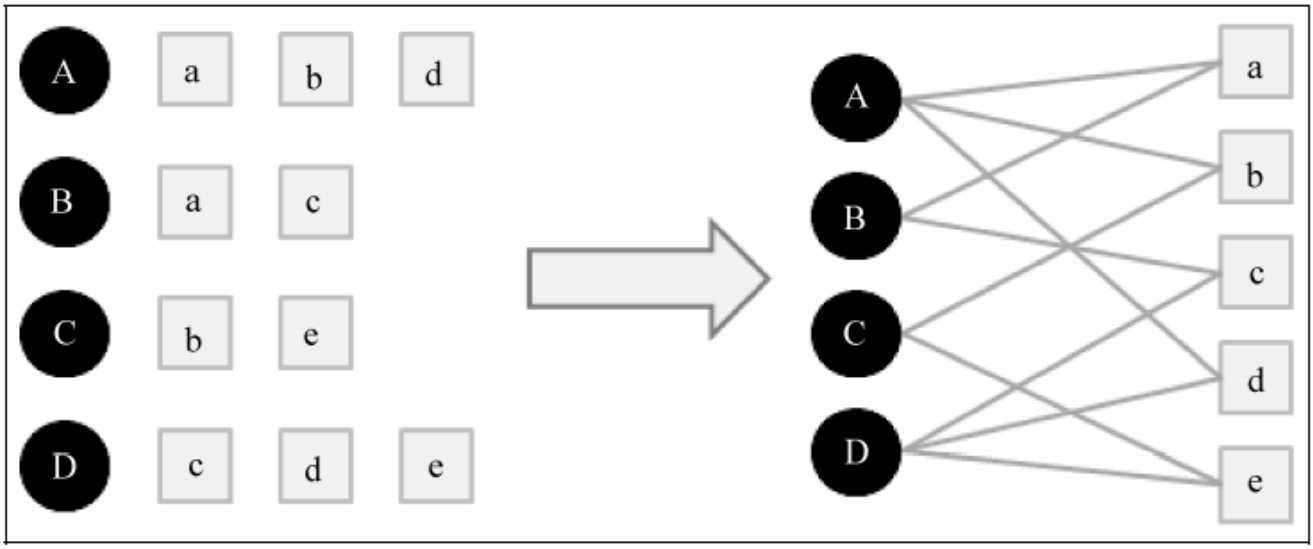
4. LFM和基于邻域的方法的比较

方面	**LFM**	**基于邻域的方法**
理论基础	LFM具有比较好的理论基础，它是一种学习方法，通过优化一个设定的指标建立最优的模型。	基于邻域的方法更多的是一种基于统计的方法，并没有学习过程。
离线计算的空间复杂度	LFM只需占用很少的内存。	基于邻域的方法需要维护一张离线的相关表。在离线计算相关表的过程中，如果用户/物品数很多，将会占据很大的内存。

方面	LFM	基于邻域的方法
离线计算的时间复杂度	两者无太大差别。	两者无太大差别。
在线实时推荐	LFM在给用户生成推荐列表时，需要计算用户对所有物品的兴趣权重，然后排名，返回权重最大的N个物品。那么，在物品数很多时，这一过程的时间复杂度非常高，LFM不太适合用于物品数非常庞大的系统，也不能实时计算。	UserCF和ItemCF在线服务算法需要将相关表缓存在内存中，然后可以在线进行实时的预测。
推荐解释	LFM无法提供这样的解释。	ItemCF算法支持很好的推荐解释，它可以利用用户的历史行为解释推荐结果。

2.6 基于图的模型

基于图的模型（graph-based model）是推荐系统中的重要内容。其实，很多研究人员把基于邻域的模型也称为基于图的模型，因为可以把基于邻域的模型看做基于图的模型的简单形式。



用户物品二分图模型

* 基于图的推荐算法

1. 度量图中两个顶点之间相关性的方法很多，但一般来说图中顶点的相关性主要取决于下面3个因素：

两个顶点之间的路径数

两个顶点之间的路径长度

两个顶点之间的路径经过的顶点

2. 而相关性高的一对顶点一般具有如下3个特征：

两个顶点之间有很多路径相连

连接两个顶点之间的路径长度都比较短

连接两个顶点之间的路径不会经过出度比较大的点

3. PersonalRank算法

算法描述：假设要给用户u进行个性化推荐，可以从用户u对应的节点v_u开始在用户物品二分图上进行随机游走。游走到任何一个节点时，首先按照概率α决定是继续游走，还是停止这次游走并从v_u节点开始重新游走。如果决定继续游走，那么就从当前节点指向的节点中按照均匀分布随机选择一个节点作为游走下次经过的节点。这样，经过很多次随机游走后，每个物品节点被访问到的概率会收敛到一个数。最终的推荐列表中物品的权重就是物品节点的访问概率。

如果将上面的描述表示成公式，可以得到如下公式：

$$PR(v) = \begin{cases} \alpha \sum\limits_{v' \epsilon in(v)} \dfrac{PR(v')}{|out(v')|} & (v \neq v_u) \\ (1-\alpha) + \alpha \sum\limits_{v' \epsilon in(v)} \dfrac{PR(v')}{|out(v')|} & (v = v_u) \end{cases}$$

作者[钱昊达]
2018年8月19日