内容主要围绕电商中用到的一些推荐算法，参考了 Xavier Amatriain 在 CMU 的 Machine Learning 暑期学校上的讲授的内容。

主要内容

- 推荐系统简介
- 传统的推荐算法
  - 非个性化推荐：热度排行 (Popularity)
  - 协同过滤 (Collaborative Filtering)
    - Memory-based (aka neighborhood-based)方法: Item-based/User-based CF
    - Model-based方法：频繁项挖掘/聚类/分类/回归/矩阵分解/RBM/图模型
  - 基于内容/知识的推荐(Content-based/Knowledge-based)
  - 混合方法(Hybird Approaches)
- 推荐算法的最新研究
  - 学习排序 (Learning to Rank)
  - 页面整体优化 (Page Optimization)
  - 情景推荐：张量分解/分解机(Factorization Machine)
  - 深度学习(Deep Learning)

信息过载问题

如何从海量信息中过滤出(对自己)有用的信息？

信息过滤：类目导航 VS 搜索 VS 推荐

- 类目导航：用户主动按类目逐层查找
  - 代表性公司：雅虎，新浪，搜狐，网易
- 搜索： 用户主动提供意图明确的query
  - 代表性公司：Google
- 推荐： 系统主动提供给用户一种选择
  - 代表性公司：Netflix, 今日头条
- 推荐的优势
  - 用户大多数情况下并没有明确的意图
  - 推荐可以给帮助用户发现，带给用户惊喜

推荐系统的发展 [3]

- 推荐系统在不同领域的探索
  - 1992年，Goldberg提出了第一个(个性化邮件)推荐系统Tapestry，第一次提出了协同过滤的思想，利用用户的标注和行为信息对邮件进行重排序。
  - 1994年，Resnick等人提出了针对新闻消息的协同过滤推荐系统 GroupLens。
  - 1996年，在Berkeley的协同过滤专题讨论会上，提出了推荐系统这一概念。
- 推荐系统的商业化
  - 1995年，MIT的Pattie Maes研究小组创立了Agents公司(后来更名为Firefly Networks)。
  - 关注技术问题：降低在线计算时间，冷启动问题，可信度、可解释性等用户体验问题。
- 推荐系统的学术研究
  - 2006年，Netflix的百万美元竞赛。
  - 2007年，J.A. Konstan 等人组织了第一届ACM推荐系统大会(RecSys)。
- 推荐系统的未来：基于上下文的推荐(情景推荐)
  - 人们开始更多关注推荐系统是否真正满足用户的需求。

推荐系统的应用和价值

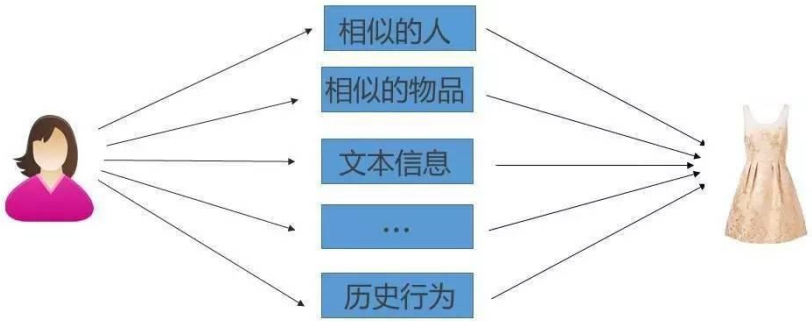
- 推荐的应用
  - 音乐、电影的推荐
  - 电子商务中商品推荐
  - 个性化阅读（新闻消息）
  - 社交网络好友推荐、朋友圈推荐
  - 基于位置的服务推荐
  - ...
- 推荐的价值
  - Netflix: 2/3 的电影是因为被推荐而观看
  - Google News: 推荐提升了38%的点击
  - Amazon: 销售中推荐占比高达35%

推荐系统的评价标准

- 用户满意度 (User Satisfaction)：调研或用户反馈；点击率、转化率等
- 准确性 (Accuracy)：precision/recall/F-score
- 覆盖率(Coverage)： 照顾到尾部物品和用户
- 多样性(Diversity)：两两之间不相似
- 新颖性(Novelty)：没听过、没见过的物品
- 惊喜性(Serendipity)：如何评价？
- 用户信任度(Trust) /可解释性（ explanation ）：推荐理由
- 鲁棒性/健壮性(Robustness)：哈利波特现象；抗攻击、反作弊
- 实时性(Real-time/online)：新加入的物品；新的用户行为(实时意图)
- 商业目标(business target)：一个用户带来多少盈利

推荐系统核心问题

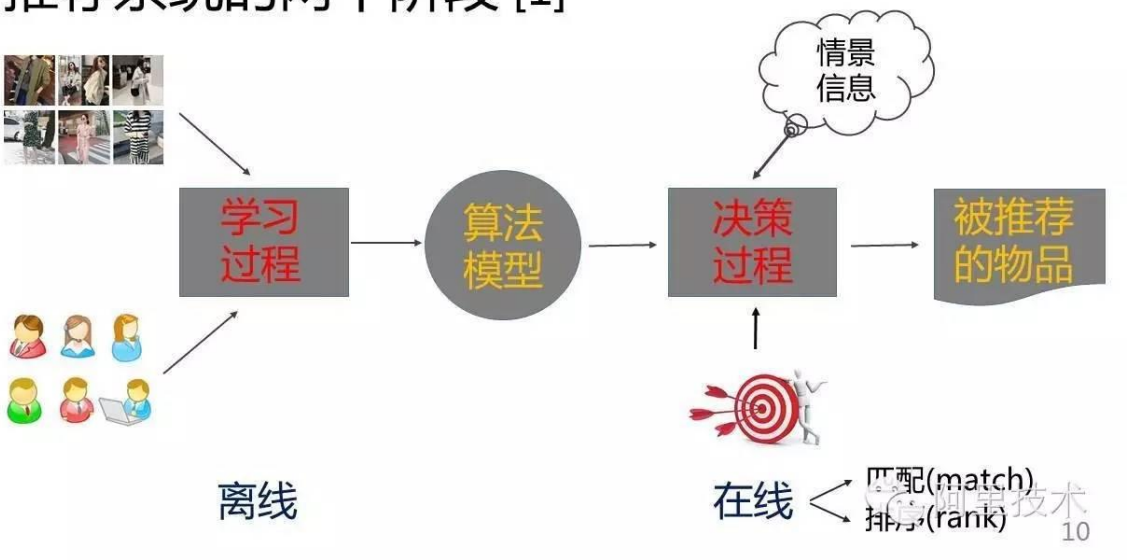
- 如何评估一个用户(user)对一个物品(item)的评分(喜欢程度)？



影响推荐效果的因素

- 用户交互界面 (User Interface)
  - 用户对推荐系统第一感知，很重要
  - 案例：Twitter的 ”favorite” 图案由star换成heart
- 数据 (Data)
  - 数据收集：有效性(是否真实、准确)、全面性(是否有偏)
  - 数据处理：清洗、挖掘 (算法工程师值得投入的地方)
- 领域知识 (Domain knowledge)
  - 产品的定位、具体推荐需求的理解
- 算法迭代 (Algorithm/Model)
  - 锦上添花（ 10%~20% ）、量变引起质变

推荐系统的两个阶段 [1]

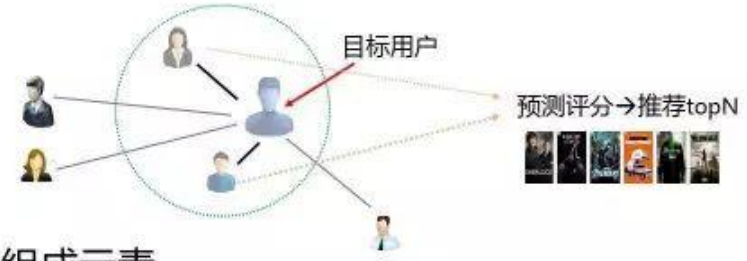


热度排行榜

- 不同的排行榜算法
  - 单一维度的投票
  - 考虑多个维度的综合打分
  - 考虑时间因素：如引入衰减权重 (半衰期、冷却定律)
  - 考虑反馈信息：如Reddit算法
  - 考虑置信度：如威尔逊区间
  - 防止马太效应：如MAB算法等
- 评价
  - 容易实现，可以解决新用户的冷启动问题
  - 更新很慢，很难推新
  - 排行榜更多的是一个业务问题，而不是一个算法问题

协同过滤 [1]

- 思想来源：现实生活中朋友之间相互推荐自己喜欢的东西



- 基本组成元素
  - N个用户，M个物品
  - 评分矩阵：用户对物品的评分(显式) 或 用户对物品的行为(隐式)
  - 相似性度量：物品与物品(人与人)之间的相似性

User-based CF

- 算法步骤
  - Step1 计算目标用户的(前k个)相似用户(neighborhood)　　协同（在线）
    - 相似性度量： Pearson相关系数，Jaccard距离，cosine相似性
  - Step2 找出相似用户喜欢的物品，并预测目标用户对这些物品的评分
    - 预测模型：kNN，regression
  - Step3 过滤掉目标用户已经消费过的物品
    - 消费：购买商品，浏览新闻等
  - Step4 将剩余物品按照预测评分排序，并返回前N个物品　　过滤（离线）

Item-based CF

- 2001年，Sarwar 等人提出了 item-based 的 CF [6]
- 2003年，Amazon介绍了其 Item-to-Item算法 [7]
- 基本思想：计算item之间两两的相似性，目标用户对某个item的评分可以根据其对相似的items的评分来预测，选取用户预测评分最高的前N个商品进行推荐
- Item-based在电商推荐中逐渐成为主流的原因
  - 降低实时计算的复杂性
  - 解决用户的冷启动问题

User-based VS Item-based

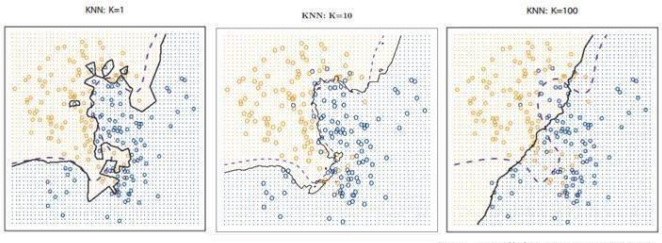
- Item-based：准确性好，表现稳定可控，便于离线计算；但是推荐结果的多样性会差一些，一般不会带给用户惊喜性。
- User-based：可以帮助用户发现新的商品，但是需要较复杂的在线计算，需要处理新用户的问题。
- Item之间的相似性比较单纯，是静态的；而user之间的相似性比较复杂，而且是动态的，应用时需要特别小心。

协同过滤的优缺点

- 优点
  - 需要很少的领域知识，模型通用性强
  - 工程上实现简单，效果很好
- 缺点
  - 冷启动问题 (new users/items)
  - 数据稀疏性问题
  - 假定“过去的行为决定现在”，没有考虑具体情景的差异
  - 热门倾向性(Popularity Bias)：很难推荐出小众偏好

最近邻算法

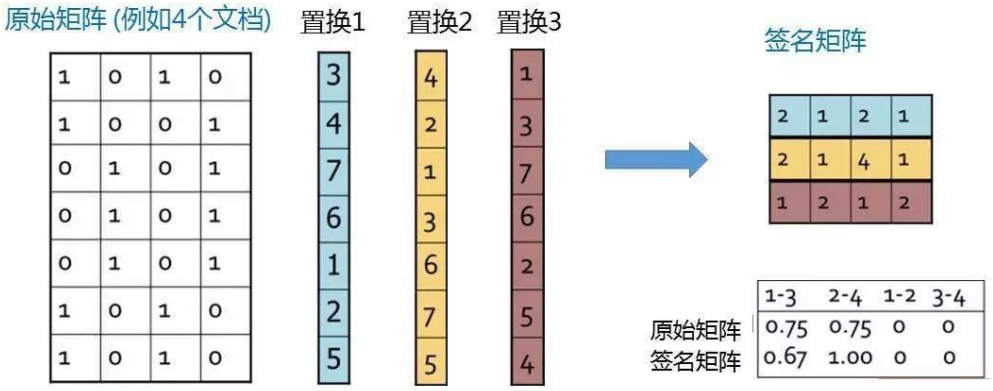
- 邻域大小的选择
  - 太小不好，不具有代表性
  - 太大也不好，噪音数据



(注： 图中紫色是Bayes分类边界)

- 最近邻的查找：Locality-Sensitive Hashing
  - Locality-sensitive： 相似的item哈希后仍保持较大概率相似，这样在计算相似性时可以只关注少量的候选集，从而大大加快相似项的查找速度。
  - 利用适当的hash将原来(高维)空间的数据哈希到一个新的(低维)空间，得到签名矩阵(Signature Matrix)，然后利用签名矩阵近似计算原空间的相似性。
  - hash函数的选择跟相似性度量选取有关，例如相似性度量是Jaccard距离，hash函数通常选用MinHash/SimHash (可以证明，MinHash的hash值相等的概率等于其原空间的Jaccard相似度)

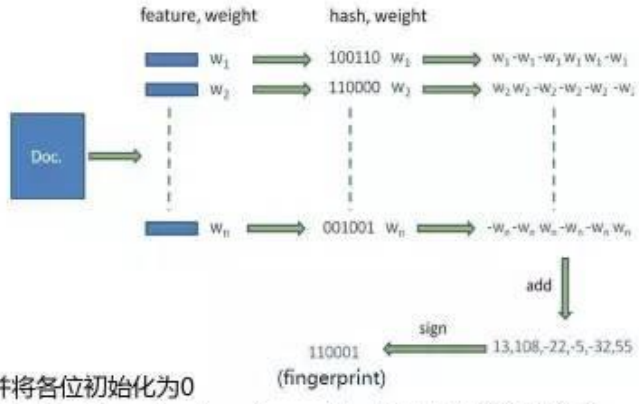
MinHash 例子



MinHash: 生成一个行的随机置换，矩阵的每一列的哈希值就定义为按照这个顺序进行置换后该列第一个值为1的行的行号

18

SimHash 例子



整个过程可以分为以下几步:

1、选择simhash的位数(比如说32位)，并将各位初始化为0
2、提取原始文本中的特征及其权重，比如对于“the cat sat on the mat”，采用两两分词的方式得到如下结果：{ “th”， “he”， “e_”， “_c”， “ca”，…，“_m”， “ma” }，权重都为1。
3、计算各个word的hashcode，比如： “th”.hash = -502157718 ， “he”.hash = -369049682，.....
4、对各word的hashcode的每一位，如果该位为1，则在simhash的相应位加它的权重；否则在simhash的相应位减去它的权重
5、对最后得到的32位的simhash，如果该位大于1，则设为1；否则设为0

19

Mode-based CF

- 关联规则，如 Apriori，FP-Growth
- 聚类
- 分类 / 回归模型
- 隐语义模型，如 LSA / pLSA / LDA
- SVD，SVD++
- RBM
- 图模型, 如SimRank, MDP

20

关联规则(Associate Rule)

- 基本思想：基于物品之间的共现性挖掘频繁项
- 一般应用场景：看了又看，买了又买，商品搭配
- 算法：A-priori、FP-Growth

支持度 (support) $s(X,Y)=\frac{包含(X,Y)的记录数}{总记录数}$

置信度(confidence) $c(X,Y)=\frac{包含(X,Y)的记录数}{包含X的记录数}$

- 评价
  - 实现简单，通用性较强，适合“推荐跟已购买商品搭配的商品”等场景
  - 相似商品的推荐效果往往不如协同过滤好
  - 解释变量之间的关联时要特别小心，很可能受一些隐含因素的影响，我们得出完全相反的结论(辛普森悖论)

21

辛普森悖论(Simpson's Paradox)

举例：考虑买高清电视(HDTV)和健身器材之间的联系，如下图所示

买HDTV	买健身器		
	是	否	
是	99	81	180
否	54	66	120
	153	147	300

规则
{买HDTV=是} ->{买健身器材=是}的confidence是99/180=55%,
{买HDTV=否} ->{买健身器材=是}的confidence是54/120=45%;

结论：购买了HDTV的人比没有购买HDTV的人更可能买健身器材。

完全相反的结论！

然而，我们发现顾客可以分组，分为大学生和在职人员，如下图所示

顾客分组	买HDTV	买健身器		
		是	否	
大学生	是	1	9	10
	否	4	30	34
在职人员	是	98	72	170
	否	50	36	86

对于大学生，
{买HDTV=是} ->{买健身器材=是}的confidence是1/10=10%
{买HDTV=否} ->{买健身器材=是}的confidence是4/34=11.8%
对于在职人员，
{买HDTV=是} ->{买健身器材=是}的confidence是98/170=57.7%
{买HDTV=否} ->{买健身器材=是}的confidence是50/86=58.1%

结论：不买HDTV的人比购买了HDTV的人更可能买健身器材。

22

辛普森悖论

- 英国统计学家辛普森在1951年提出来的，即两组数据分别讨论时都会满足某种性质，可是一旦合并考虑，却可能导致相反的结论。
- 产生辛普森悖论的主要原因是数据的不同分组之间基数差异很大(通常一个组相对于其他组来说数量占绝对优势)
- 数学解释
  - a/b < c/d 并且 p/q < r/s, 其中 a/b 和 p/q 是 A->B 在两个不同组的置信度，c/d 和 r/s 是 A' ->B 在两个不同组的置信度（这里A'为A的补），当两组数据混在一起时，两个规则置信度就是分别变为了(a+p)/(b+q)和(c+r)/(d+s)。当(a+p)/(b+q)>(c+r)/(d+s) 时，辛普森悖论就出现了。
- 结论：分析数据时合理地将数据分组(分层)很重要

基于统计的相似性

- OddsRatio

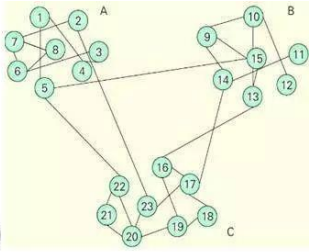
	Item 2	! Item 2	
Item 1	N11	N12	R1
! Item1	N21	N22	R2
	C1	C2	N

OddsRatio(item1,item2) = Odds(item2|item1)/Odds(item2|!item1)
= (N11*N22)/(N12*N21)

解释： OddsRatio 值表示了“看了item1的用户中看了item2的用户数与没看item2的用户数的比例 N11/N12”与“没看item1的用户中看了item2的用户数与没看item2的用户数的比例N21/N22”的比值。该比值大于1表明看了item1的用户更倾向于看item2，并且 OddsRatio 值越大，说明item1到item2的关系越紧密。

平滑： LogOddsRatio(item1,item2)=logN11+logN22-logN12-logN21

聚类 (Clustering)



- 用户分群：同好人群
  - 有时可以带给用户惊喜感，但个性化稍差(人群 vs 个体)
- 商品聚类：相似商品
  - 精准性较高，但推荐的商品对用户大多无新鲜感
- 常用算法
  - k-means, 层次聚类 (HC)
  - Louvain(2008), 基于密度的聚类方法(2014)
- 评价
  - 聚类可以一定程度解决数据稀疏性问题，可以捕捉到一些隐(扩展)相似性
  - 聚类的精准度往往不如协同过滤好

分类/回归

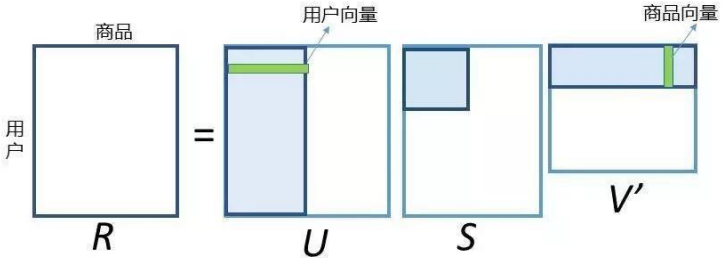
- 基本思想：把评分预测看做一个多分类（回归）问题
- 常用分类器：LR, Naive Bayes
- 输入：通常是item的特征向量
- 评价
  - 比较通用，可以跟其他方法组合，提高预测的准确性
  - 需要大量训练数据，防止过拟合现象

矩阵分解-SVD分解

- 数学上的SVD

$$R_{m\times n}=U_{m\times m}S_{m\times n}V'_{n\times n}$$

$$R_{m\times n}\approx U_{m\times k}S_{k\times k}V'_{k\times n}$$



- 传统SVD应用到推荐中的挑战
  - SVD要求矩阵必须是稠密矩阵，而评分矩阵非常稀疏
  - SVD计算复杂度很高，在稠密大规模矩阵上运行速度很慢

FunkSVD [15]

- 2006年, Simon Funk在其博客上公布了一个算法(FunkSVD)
- 思想: 评分矩阵R分解为两个低秩矩阵的乘积 $R_{m\times n}=P_{m\times k}^{T}Q_{k\times n}$，则用户u对物品i的评分为 $\hat{r}_{ui}=\mathrm{q}_i^T\mathrm{p}_u$，利用RMSE为目标去学习P、Q，并且学习过程中不关心缺失值

目标函数： $\min C=\sum_{(u,i)}(r_{ui}-\mathrm{q}_i^T\mathrm{p}_u)^2+\lambda\left(\|\mathrm{q}_i\|^2+\|\mathrm{p}_u\|^2\right)$

梯度下降法求解：

$$\mathrm{p}_u=\mathrm{p}_u+\alpha(e_{ui}\,\mathrm{q}_i-\lambda\,\mathrm{p}_u)$$

$$\mathrm{q}_i=\mathrm{q}_i+\alpha(e_{ui}\mathrm{p}_u-\lambda\mathrm{q}_i)$$

$$e_{ui}=r_{ui}-\hat{r}_{ui}$$

Funk的博客地址: http://sifter.org/~simon/journal/20061211.html

BiasSVD

- 思想(基本假设)
  - 评分系统有一些跟用户和物品无关的固有属性
  - 用户有一些跟物品无关的固有属性
  - 商品有一些跟用户无关的固有属性
- 引入偏置项

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

μ 所有评分的全局平均分 (该参数不需要学习)

b_u 用户偏置项，用户评分习惯中跟物品无关的因素

b_i 物品偏置项，物品接受评分中跟用户无关的因素

目标函数： $\min \sum_{(u,i)} (r_{ui} - \mu - b_i - b_u - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2 + \| b_i \|^2 + \| b_u \|^2)$

b_u, b_i 初始化为零向量, 然后迭代 $b_u = b_u + \alpha(e_{ui} - \lambda b_u)$, $b_i = b_i + \alpha(e_{ui} - \lambda \ldots)$

SVD++ [16]

- 思想：考虑用户的隐式反馈 (提供额外的用户偏好信息)

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u + \sum_{j \in N(u)} c_{ij}$$

$N(u)$ 是用户u提供了隐式反馈信息的item集合

c_{ij} 是利用u对j的一些隐式反馈对i的评分进行补偿(修正)

为了减少参数(防止过拟合)，矩阵 C 可分解为 $c_{ij} = x_i^T y_j$

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u + |N(u)|^{-\frac{1}{2}} \cdot x_i^T \sum_{j \in N(u)} y_j$$

(引入 $|N(u)|^{-\frac{1}{2}}$ 消除 不同 $N(u)$ 的大小差异)

为了进一步减少参数，令 $x_i = q_i$, 得到SVD++模型 ：

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T (p_u + |N(u)|^{-\frac{1}{2}} \cdot \sum_{j \in N(u)} y_j)$$

可以看作利用隐式反馈信息对用户向量 p_u 做了修正

目标函数：

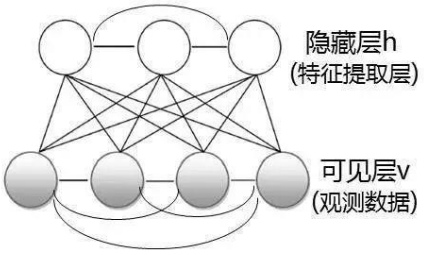
$$\min \sum_{(u,i)} (r_{ui} - \mu - b_i - b_u - q_i^T (p_u + N(u)^{-\frac{1}{2}} \sum_{j \in N(u)} y_j))^2 + \lambda(\| q_i \|^2 + \| p_u \|^2 + \| b_i \|^2 + \ldots \| \sum_{j \in N(u)} y_j \|^2)$$

Yehuda Keren. Factorization meets the neighborhood: a multifaceted collaborative filtering model, 2008

Item的向量化

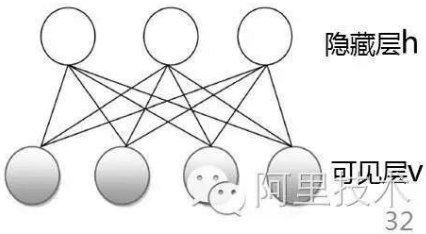
- 为什么进行向量化？
  - 知识、概念的表示
  - 为智能推荐系统的设计提供了强有力的工具
- 向量化的常用方法
  - pLSA (Thomas Hofmann,1999[11])
  - LDA (David M. Blei, 2003[12])
  - Word2Vec (Mikolov, 2013 [13])
- 实际应用的问题
  - 向量化时大多基于行为数据，行为少的item向量化效果不好

限制玻尔兹曼机(RBM)



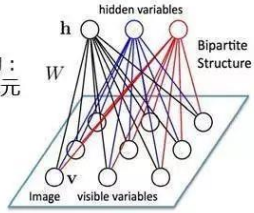
- Boltzmann机
  - 两层神经网络 (无输出层)
  - 对称、全连接
  - 随机：每个神经元只有两种状态(激活/未激活), 通常用0/1表示，但是某一时刻处于哪种状态是随机的，由一定的概率确定 $P(s_i=1)=1/(1+e^{-E_{zi}/T})$

- 受限Boltzmann机
  - 受限：层内无连接 (大大简化了计算)

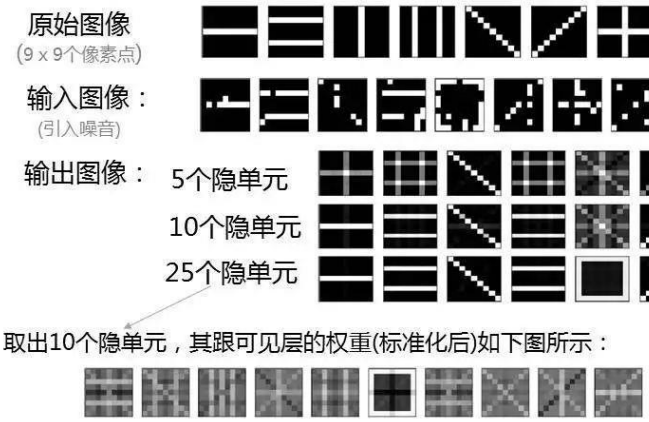


32

RBM中隐藏层的作用



- 图像去噪和重构的例子



每个隐单元的权重“记忆”了某一(几)种特征的模式的部分或全部

33

限制玻尔兹曼机(RBM) [18]

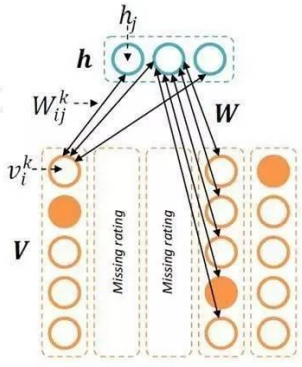
问题：N个用户，M个电影，评分等级 1~K

如何建模 ？(RBM的结构)

如何处理缺失值 ？

一个用户一个RBM

共享隐层，共享权重W，共享偏置b



$v_i^k=1$：用户i对电影v的评分为k

Salakhutdinov et al. , Restricted Boltzmann Machines for Collaborative Filtering ,2006.

RBM进行评分预测

算法步骤（预测用户u对电影i的评分）：

1. 用用户已有的评分来给相应的可见单元赋值
2. 对每一个隐单元 j，计算 $\hat{p}_j = P(h_j = 1 \mid V) = 1/(1+e^{-b_j - \sum_{i=1}^{m}\sum_{k=1}^{K} v_i^k W_{ij}^k})$
3. 计算 $P(v_i^k = 1 \mid \hat{p}) = e^{b_i^k + \sum_{j=1}^{F} \hat{p}_j W_{ij}^k} \Big/ \sum_{s=1}^{K} e^{b_i^s + \sum_{j=1}^{F} \hat{p}_j W_{ij}^s}$
4. 用户 u 对电影 i 的评分为 $R(u,i) = \sum_{k=1}^{K} p(v_i^k = 1)k$

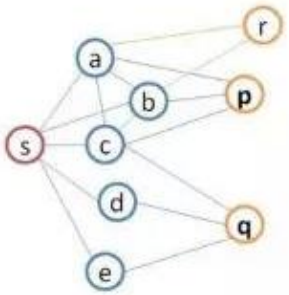
选择预测评分最高的电影进行推荐

在Netflix Prize中，RBM是单个算法中表现最好的

基于图的相似性

- 多用于社交网络人与人的关系挖掘
- 共同邻居数: $s(a,b) = |N(a) \cap N(b)|$
- Jaccrad： $s(a,b) = \left|\frac{N(a) \cap N(b)}{N(a) \cup N(b)}\right|$
- Adamic/Adar:

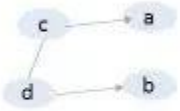
$$s(a,b) = \sum_{z \in N(a) \cap N(b)} \frac{1}{\log |N(z)|}$$



- TriangleRank

$$s(a,b) = \sum_{i \in N(a) \cap N(b)} \sum_{j \in N(a) \cap N(b)} \frac{Trg(i,j,b)}{1 + \sum_{c \in N(N(a))} Trg(i,j,c)}$$

图模型



- SimRank (Jeh & Widom, 2002 [9])

基本思想：被相似的对象引用的两个对象也具有某种相似性.

$$s(a,b) = \frac{c}{|I(a)||I(b)|} \sum_{c \in I(a)} \sum_{d \in I(b)} s(c,d), \quad c \in (0.6, 0.8) \xrightarrow{\text{随机游走}} s(a,b) = E(c^{X_{a,b}})$$

迭代： $s_{k+1}(a,b) = \frac{c}{|I(a)||I(b)|} \sum_{i \in I(a)} \sum_{j \in I(b)} s_k(i,j)$ $\quad s_0(a,b) = \begin{cases} 1, a = b \\ 0, a \neq b \end{cases}$

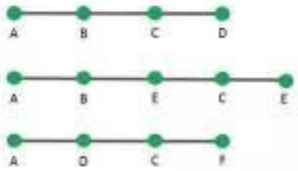
- SimRank++ (Ioannis Antonellis, 2007 [10])

改进1： 引入evidence系数 (两个节点的共同节点越多越相似)

改进2： 考虑了边的权重

图模型(续)

- Markov模型（n-gram）



$$P(A \to B) = \frac{2}{3}$$

$$P(A \to D) = \frac{1}{3}$$

Skipping
Weighting
Smoothing

- Markov Decision Process
- 图模型的评价
  - 借助图的结构的传导性，可以发现CF发现不了的弱相似性，给推荐带来一定的惊喜性；但是图模型仍然面临数据稀疏性问题和冷启动问题。

基于内容/知识的推荐

- 什么是内容？
  - 文本描述：通常用NLP技术挖掘关键词
  - Item的属性，如电影的主题，衣服的材质等
  - Item的特征，例如语音信号表示、图像向量表示等
- 基本组成
  - item特征向量，如文本的TF-IDF向量
  - 用户profile向量：通常通过用户偏好的items来提取
  - 匹配分：直接计算cosine， 分类/回归模型

类似搜索

基于内容推荐的评价

- 优点
  - 能够推荐出用户独有的小众偏好
  - 可以一定程度解决数据稀疏问题和item的冷启动问题
  - 通常具有较好的可解释性
- 不足
  - 对于很多推荐问题，提取出有意义的特征并不容易
  - 很难将不同item的特征组合在一起 (邻域思想)
  - 很难带给用户惊喜性
  - 如果用户的profile挖掘不准，推荐效果往往很差

混合方法 (Hybird Approaches)

- 模型融合方法
  - 加权 (weighted)组合
  - 切换 (switching)：确定一个合理的切换跳进
  - 混合(Mixed)
  - 特征组合(feature combination)：不同模型特征组合到一起
  - 特征扩展 (feature augmentation)：一个模型的输出作为另一个的特征
  - 级联(Cascade) ：粗排->精排
- 评价
  - 通常比单个算法表现好
  - 需要在不同算法之间、理论效果和实际可行性之间仔细权衡

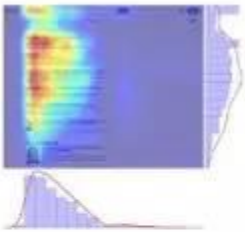
Learning to Rank

- 大多数推荐结果是一个列表，因此推荐结果的排序也很重要。
- L2R： 将排序看做一个ML问题
- 排序的评价准则：
  - Normalized Discounted Cumulative Gain (NDCG)
  - Mean Reciprocal Rank (MRR)
- 问题：很难直接利用这些指标来指导学习过程
- 常用算法
  - Pointwise： LR， GBDT，SVM
  - Pairwise：RankSVM，RankBoost, RankNet, LambdaRank...
  - Listwise： AdaRank，LambdaRank/LambdaMart, ListNet, ListMLE

页面整体优化

- 用户注意力建模

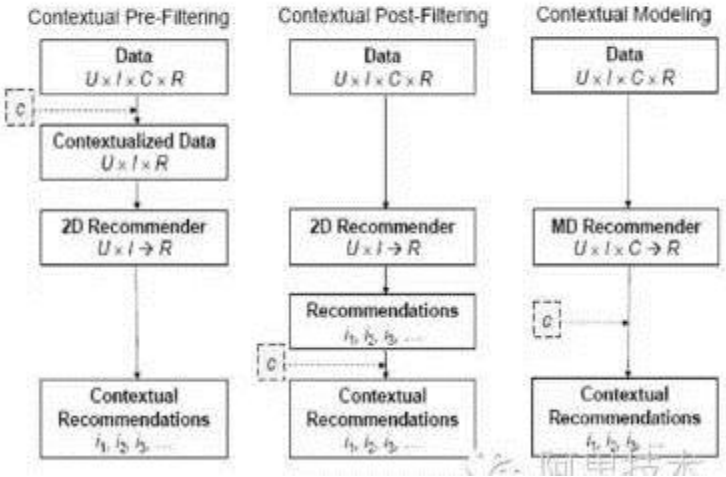
Web Search (Google)



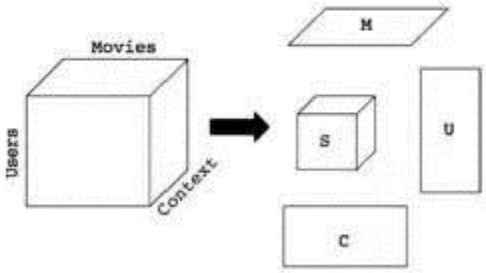
参见：
Modeling User Attention and Interaction on the Web, Dmitry Lagun, 2014. (PhD Thesis) (Emory U.)
Beyond Ranking: Optimizing Whole-Page Presentation, Yue Wang, 2016 WSDM

情景(Context-aware)推荐

- 什么是情景？
  - 看电影：工作日？周末？
  - 订餐：餐厅距离
  - 网购：用户心情
  - 母婴：孩子年龄
- 三种实现方式 [21]



情景推荐的算法



- 张量分解(Tensor Factorization)
  - 参见 A Karatzoglou,2010 [19]
- 分解机(Factorization Machine)
  - 矩阵(张量)分解跟线性(逻辑)回归相融合的一种推广

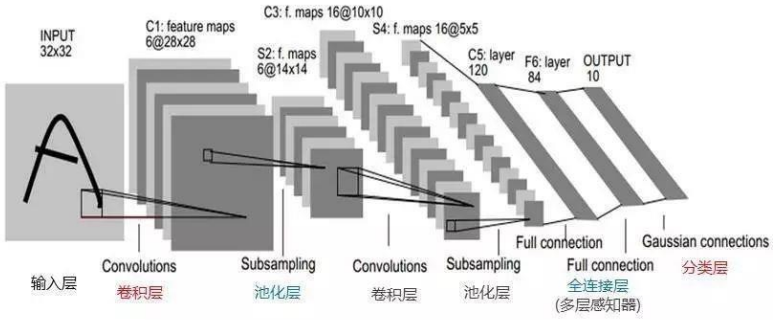
$$f(\mathbf{x}) = b + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} x_i x_j \mathbf{v}_i^T \mathbf{v}_j$$

$$= b + \sum_{i=1}^{d} w_i x_i + \frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{d} x_i v_{i,f} \right)^2 - \sum_{i=1}^{d} x_i^2 v_{i,f}^2 \right)$$

  - 参见Rendle,2010 [20]
- Factorization Machine效果更好

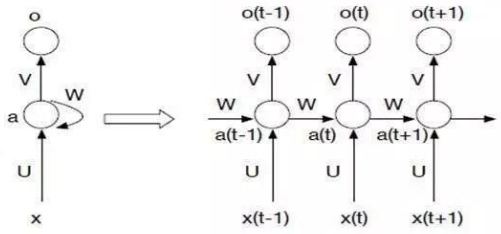
深度学习算法

- 卷积神经网络(CNN)
- 循环神经网络(RNN)



$$o(i) = softmax(Va(i))$$

$$a(i) = f(Ux(i) + Wa(i-1))$$



深度学习在推荐上的应用

- Spotify将CNN用于音乐推荐 [23]
  - 利用CNN从歌曲音频提取隐特征向量 (解决冷启动问题)
- 考拉将RNN用于电商推荐 [24]
  - 传统i2i：基于最后一个行为
  - RNN：基于用户整个session所有行为(序列数据)
- 评价
  - 利用DL有希望开发出更加智能化的推荐系统
  - RNN可用于捕捉用户实时意图，但是仅仅利用行为数据的RNN可能效果有限（可能的改进点：1.将图像、文本、行为数据综合起来; 2. 压缩状态空间； 3. 巧妙构造训练数据）

总结

- 深入理解推荐需求比算法更重要！
- 深入理解业务目标比算法更重要！
- 对数据的理解和处理比算法更重要！
- 对用户的理解比算法更重要！
- 多个模型的融合可以提高预测准确性，但在实际应用中需要权衡成本。
- 一个推荐系统应该从多个指标去综合评价，可以追求单一指标是危险的。
- 随着移动互联网时代的到来，(智能)推荐会变得越来越重要。

参考文献

[1] Xavier Amatriain , Recommender Systems, MLSS-2014 @CMU.
[2] Recommender Systems Handbook. Francesco Ricci et al. Springer, 2010.
[3] Recommender systems: an introduction. Jannach, Dietmar, et al., Cambridge University Press, 2010.
[4] Using collaborative filtering to weave an information Tapestry, David Goldberg et al., 1992.
[5] GroupLens: An Open Architecture for Collaborative Filtering Contents, Bradley N. Miller, 1994.
[6] Item-based Collaborative Filtering Recommendation Algorithms, B. Sarwar et al. 2001.
[7] Amazon Recommendations_Item-to-item Collaborative Filtering, Greg Linden et al., 2003.
[8] Locality-Sensitive Hashing for Finding Nearest Neighbors, M Slaney, M Casey, 2008.
[9] SimRank: A Measure of Structural-Context Similarity, Glen Jeh, Jennifer Widom, 2002.
[10] SimRank++: Query Rewriting Through Link Analysis of the Click Graph, Antonellis et al. , 2007.
[11] Probabilistic Latent Semantic indexing, Thomas Hofmann, 1999.
[12] Latent Dirichlet Allocation, David M.Blei et al., 2003.
[13] Linguistic Regularities in Continuous Space Word Representations, Mikolov et al., 2013.

参考文献

[14] Content-based Recommendation Systems. M. Pazzani and D. Billsus, 2007.
[15] Funk's blog: http://sifter.org/~simon/journal/20061211.html
[16] Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering model, Koren, 2008.
[17] Matrix Factorization Techniques for Recommender Systems, Koren et al., 2009.
[18] Restricted Boltzmann machines for collaborative filtering. R. Salakhutdinov et al., 2007.
[19] Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering, A Karatzoglou,2010.
[20] Factorization Machines, Steffen Rendle , 2010.
[21] Context-Aware Recommender Systems, Gediminas Adomavicius, Alexander Tuzhilin, 2008.
[22] Learning to rank: From pairwise approach to listwise approach. Z. Cao and T. Liu, ICML,2007.
[23] Deep content-based music recommendation, Aaron van den Oord et al. 2013.
[24] Personal Recommendation Using Deep Recurrent Neural Networks in NetEase, Sai Wu, 2016.