推荐系统实践

推荐系统基本概念

- 推荐系统与搜索引擎比较
  - 推荐系统和搜索引擎都是用来解决信息过载问题
  - 搜索引擎满足用户有明确目的时主动查找的寻求
  - 推荐系统在用户没有明确的目的是帮助他们发现感兴趣的新内容
- 推荐系统实验方法
  - 离线实验
  - 用户调查
  - 在线实验
- 评测指标
  - 用户满意度
  - 预测准确度
  - 覆盖率
  - 多样性
  - 新颖度
  - 惊喜度

    如果推荐结果和用户的历史兴趣不相似，但却让用户觉得满意，那么就可以说推荐结果的惊喜度很高，而推荐的新颖性仅仅取决于用户是否听说过这个推荐结果
  - 信任度
  - 实时性
  - 健壮性

基于用户行为数据的推荐算法

用户行为数据

- 无上下文信息的隐性反馈数据集
- 无上下文信息的显性反馈数据集
- 有上下文信息的隐性反馈数据集
- 有上下文信息的显性反馈数据集

评测指标(TopN推荐)

- 召回率：$Recall = \frac{\sum_u |R(u) \cap T(u)|}{\sum_u |T(u)|}$
- 准确率：$Precision = \frac{\sum_u |R(u) \cap T(u)|}{\sum_u |R(u)|}$
- 覆盖率：$Coverage = \frac{|\cup_u R(u)|}{|I|}$
  - $R(u)$表示对用户推荐的N个物品
  - $T(u)$表示测试集用户喜欢的物品集合
  - I表示所有的物品

基于邻域的算法

- 基于用户的协同过滤
  - 找到和目标用户兴趣相似的用户集合
    - Jaccard用户兴趣相似度

      $$w_{uv} = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$$
    - 余弦相似度

      $$w_{uv} = \frac{|N(u) \cap N(v)|}{\sqrt{|N(u)||N(v)|}}$$
    - 用户兴趣相似度改进

      $$w_{uv} = \frac{\sum_{i \in N(u) \cap N(v)} \frac{1}{log(1 + |N(i)|)}}{\sqrt{|N(u)||N(v)|}}$$
  - 找到这个集合中的用户喜欢的，但是目标用户没有听说过的物品推荐给目标用户

        - 用户u对物品i的感兴趣程度

          $$p_{ui} = \sum_{v \in S(u,K) \cap N(i)} w_{uv} r_{vi}$$

      $N(i)$表示喜欢物品i的用户集合

      r_{vi}表示用户v对物品i的兴趣
  - 基于物品的协同过滤
    - 计算物品之间的相似度
      - 物品的相似度

        $$w_{ij} = \frac{|N(i) \cap N(j)|}{|N(i)|}$$
      - 余弦相似度

        $$w_{ij} = \frac{|N(i) \cap N(j)|}{\sqrt{|N(i)||N(j)|}}$$
      - 用户活跃度对物品相似度的影响

        $$w_{ij} = \frac{\sum_{u \in N(i) \cap N(j)} \frac{1}{log(1 + |N(u)|)}}{\sqrt{|N(i)||N(j)|}}$$
    - 根据物品的相似度和用户的历史行为给用户生成推荐列表
      - 用户u对一个物品j的兴趣

        $$p_{uj} = \sum_{i \in N(u) \cap S(j,K)} w_{ji} r_{ui}$$

      $N(u)$表示用户u有过正反馈的物品集合

      r_{ui}表示用户u对物品i的兴趣

隐语义模型

核心思想是通过隐含特征(*latent factor*)联系用户兴趣和物品

LFM通过如下公式计算用户u对物品i的兴趣：

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

其中$p_{u,k}$度量了用户u的兴趣和第k个隐类的关系，而$p_{i,k}$度量了第k个隐类和物品i之间的关系

优化如下的损失函数来找到最合适的参数p和q：

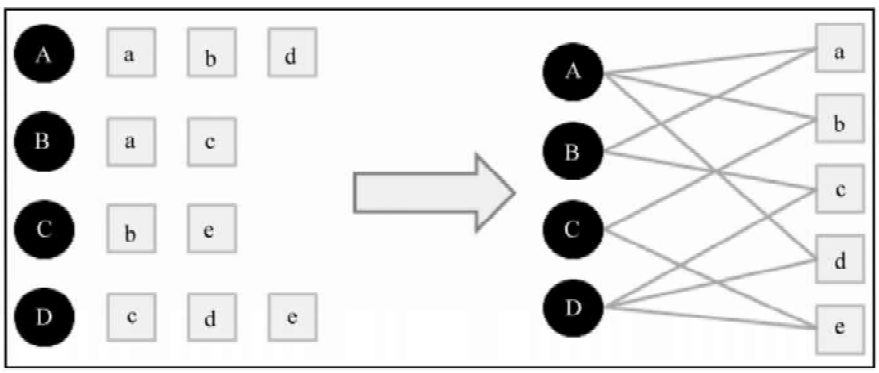
- $C(p,q) = \sum_{(u,i) \in train} (r_{ui} - \hat{r}_{ui})^2 = \sum_{(u,i) \in train} \left(r_{ui} - \sum_{f=1}^{F} p_{uf} q_{if} \right)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2)$

随机梯度下降法：

- $\frac{\partial C}{\partial p_{uf}} = -2q_{if} + 2\lambda p_{uf}$
- $\frac{\partial C}{\partial q_{if}} = -2p_{uf} + 2\lambda q_{if}$
- $p_{uf} = p_{uf} + \alpha(q_{ik} - \lambda p_{uk})$
- $q_{if} = q_{if} + \alpha(p_{uk} - \lambda q_{ik})$

基于图的模型

令$G(V,E)$表示用户物品二分图，其中$V = V_U \cup V_I$由用户顶点集合V_U和物品顶点集合V_I组成。对于数据集中每一个二元组(u,i)，图中都有一套对应的边$e(v_u, v_i)$，其中 $v_u \in V_u$是用户u对应的顶点， $v_i \in V_I$是物品i对应的顶点



图中顶点的相关性主要取决于：

- 两个顶点之间的路径数
- 两个顶点之间路径的长度
- 两个顶点之间的路径经过的顶点

相关性高的一对顶点一般具有如下特征：

- 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顶点

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：

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

评分预测问题

$$RMSE=\frac{\sqrt{\sum_{(u,i)\in T}(r_{ui}-\hat{r}_{ui})^2}}{|Test|}$$

平均值

- 全局平均值
  - $\mu=\frac{\sum_{(u,i)\in train} r_{ui}}{\sum_{(u,i)\in train} 1}$
  - $\hat{r}_{ui}=\mu$
- 用户评分平均值
  - $\bar{r_u}=\frac{\sum_{i\in N(u)} r_{ui}}{\sum_{i\in N(u)} 1}$
  - $\hat{r}_{ui}=\bar{r_u}$
- 物品评分平均值
  - $\bar{r_i}=\frac{\sum_{u\in N(i)} r_{ui}}{\sum_{u\in N(i)} 1}$
  - $\hat{r}_{ui}=\bar{r_i}$
- 用户分类对物品分类的平均值
  - $\hat{r}_{ui}=\frac{\sum_{(v,j)\in train,\phi(u)=\phi(v),\varphi(i)=\varphi(j)} r_{vj}}{\sum_{(v,j)\in train,\phi(u)=\phi(v),\varphi(i)=\varphi(j)} 1}$

基于邻域的方法

- 基于用户的邻域算法
  - 通过皮尔逊系数计算用户之间的相似度（相关系数）
    - $w_{uv}=\frac{\sum_{i\in I}(r_{ui}-\bar{r_u})(r_{vi}-\bar{r_v})}{\sqrt{\sum_{i\in I}(r_{ui}-\bar{r_u})^2\sum_{i\in I}(r_{vi}-\bar{r_v})^2}}$
  - 用户对该物品的评分
    - $\hat{r}_{ui}=\bar{r_u}+\frac{\sum_{v\in S(u,K)\cap N(i)} w_{uv}(r_{vi}-\bar{r_v})}{\sum_{v\in S(u,K)\cap N(i)}|w_{uv}|}$
- 基于物品的邻域算法
  - 余弦相似度
    - $w_{ij}=\frac{\sum_{u\in U} r_{ui}r_{uj}}{\sqrt{\sum_{u\in U} r_{ui}^2\sum_{u\in U} r_{uj}^2}}$
  - 皮尔逊相似度
    - $w_{ij}=\frac{\sum_{u\in U}(r_{ui}-\bar{r_i})(r_{uj}-\bar{r_j})}{\sqrt{\sum_{u\in U}(r_{ui}-\bar{r_i})^2\sum_{u\in U}(r_{uj}-\bar{r_j})^2}}$

- 修正的余弦相似度
  - $$w_{ij}=\frac{\sum_{u\in U}(r_{ui}-\bar{r_u})(r_{uj}-\bar{r_u})}{\sqrt{\sum_{u\in U}(r_{ui}-\bar{r_u})^2\sum_{u\in U}(r_{uj}-\bar{r_u})^2}}$$
- 用户对物品的评分
  - $\hat{r}_{ui}=\bar{r_i}+\frac{\sum_{j\in S(u,K)\cap N(u)}w_{ij}(r_{uj}-\bar{r_j})}{\sum_{j\in S(i,K)\cap N(u)}|w_{ij}|}$

隐语义模型与矩阵分解模型

用户的评分行为可以表示成一个评分矩阵R，其中$R[u][i]$就是用户u对物品i的评分。但是，用户不会对所有的物品评分，所以这个矩阵里有很多元素都是空的，这些空的元素称为缺失值（*missing value*）

- 传统的SVD分解
  - 首先对评分矩阵的缺失值进行简单的补全，比如全局平均值，或者用户/物品平均值补全，得到补全之后的矩阵R'，利用SVD进行分解：
    $R'=U^T\,SV$
    其中$U\in R^{k*m}$，$V\in R^{k*n}$，$S\in R^{k*k}$对角线上每个元素都是矩阵的奇异值
  - 为了对R'进行降维，可以取最大的f个奇异值组成对角矩阵S_f，并且找到这f个奇异值中每个值在U、V矩阵中对应的行和列，得到U_f、V_f，从而可以得到一个降维后的评分矩阵：
    $R_f{}'=U_f^T\,S_f\,V_f$
    $R'_f(u,i)$ 就是用户u对物品i评分的预测值
- 隐语义模型
  从矩阵分解角度来说，LFM就是将评分矩阵R分解成为两个低维矩阵相乘：
  - $\hat{R}=P^TQ$

  则用户u对物品i的评分的预测值为：
  - $\widehat{r_{ui}}=\sum_f p_{uf}q_{if}$

  LFM通过训练集中的观察值来最小化RMSE学习P、Q矩阵
  - 损失函数
    - $$C(p,q)=\sum_{(u,i)\in train}(r_{ui}-\hat{r}_{ui})^2=\sum_{(u,i)\in train}\left(r_{ui}-\sum_{f=1}^{F}p_{uf}q_{if}\right)^2$$
  - 损失函数正则化
    - $$C(p,q)=\sum_{(u,i)\in train}\left(r_{ui}-\sum_{f=1}^{F}p_{uf}q_{if}\right)^2+\lambda(\|p_u\|^2+\|q_i\|^2)$$
  - 随机梯度下降法
    - $\frac{\partial C}{\partial p_{uf}}=-2q_{if}+2\lambda p_{uf}$
    - $\frac{\partial C}{\partial q_{if}}=-2p_{uf}+2\lambda q_{if}$
    - $p_{uf}=p_{uf}+\alpha(q_{ik}-\lambda p_{uk})$
    - $q_{if}=q_{if}+\alpha(p_{uk}-\lambda q_{ik})$