第3章

基于显式反馈的物品排序

推荐系统有两个重要任务：评分预测和物品排序。前者旨在拟合用户对物品的评分值，后者侧重根据用户对物品的预测偏好对候选物品进行排序。本章主要关注物品排序任务，因为它最后的输出是物品排序列表，与真实应用中的推荐问题更为匹配，可以直接用于向用户推荐他或她可能感兴趣的物品。

对于用户反馈数据类型，通常可以分为显式反馈数据和隐式反馈数据。显式反馈数据通过评分值来反映用户对物品的喜爱程度，隐式反馈数据则通常是点击、浏览等形式的单值数据。本章关注基于显式反馈的物品排序算法，因为它包含了比隐式反馈更精确的偏好信息。

3.1 协同排序(CR)问题

本章研究的问题是协同排序(collaborative ranking，CR)问题，其定义如下：给定 n 个用户、m 个物品以及用户对物品的历史评分记录 $\mathcal{R}=\{(u,i,r_{ui})\}$，其中 r_{ui} 表示用户 u 对物品 i 的评分。目标是为每个用户生成个性化的物品排序列表，如图 3-1 所示。需要注意的是，当向用户进行推荐时，候选物品通常是指该用户所有未交互的物品，即 $\mathcal{I}\backslash\mathcal{I}_u$，其中 $\mathcal{I}$ 表示物品集合，$\mathcal{I}_u$ 表示用户 u 交互过的物品集合。本章所用

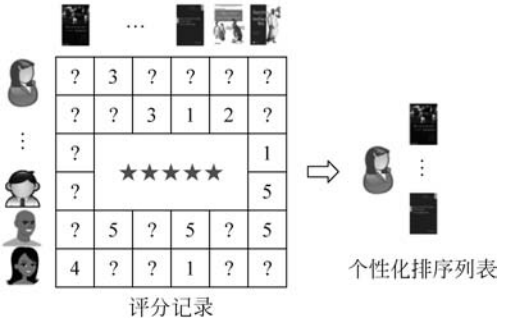

图 3-1 CR 问题示意图

到的符号和其说明如表 3-1 所示。

表 3-1　CR 问题中的符号及其说明

符　　号	说　　明
n	用户数
m	物品数
d	潜在特征向量的维度
$\mathcal{U}=\{u\}$	用户集合
$\mathcal{I}=\{i\}$	物品集合
$\mathcal{R}=\{(u,i,r_{ui})\}$	训练集的评分记录集合
$\mathbb{M}$	多类偏好集合
$r_{ui}\in\mathbb{M}$	用户 u 对物品 i 的真实评分
$\bar{r}_{ui}$	r_{ui} 归一化后的评分值
$\mathcal{I}_u^{\mathrm{P}}$	训练集中用户 u 购买过的物品集合
$\mathcal{I}_u^{\mathrm{E}}$	训练集中用户 u 浏览过的物品集合
$\mathcal{I}_u^r, r\in\mathbb{M}$	用户 u 评分为 r 的物品集合
$\hat{r}_{ui}$	用户 u 对物品 i 的预测评分
$\boldsymbol{U}_{u\cdot}\in\mathbb{R}^{1\times d}$	用户 u 的潜在特征向量
$\boldsymbol{V}_{i\cdot}\in\mathbb{R}^{1\times d}$	物品 i 的潜在特征向量
$\boldsymbol{M}_{i'\cdot}^r\in\mathbb{R}^{1\times d}$	物品 i'的潜在特征向量(评分为 r)
$b_u\in\mathbb{R}$	用户 u 的偏置
$b_i\in\mathbb{R}$	物品 i 的偏置
$\lambda_{\mathrm{mod.}}$	HoToR(mod.)中的权衡参数
$\lambda_{\mathrm{nei.}}$	HoToR(nei.)中的权衡参数
s_{ki}	物品 k 和物品 i 的相似度
$\mathcal{N}_i$	物品 i 的邻居集合

3.2　粗精迁移排序

迁移排序(transfer to rank,ToR)[1]是一种简单有效的算法,包含浏览和评分两个阶段。它首先使用用户的浏览行为数据进行全局偏好学习,然后利用评分行为数据来进一步优化候选物品列表。ToR 模型虽然在一定程度上模拟了用户的购物过程,但忽略了用户在购买选择上的差异,也就是说,某些用户虽然给物品打了高分,但他们不一定会购买该物品。为了解决此问题,粗精迁移排序(coarse-to-fine transfer to rank,CoFiToR)[2]将用户购物过程进一步细分为三个阶段,即浏览阶段(E 阶段)、评分阶段(S 阶段)和购买阶段(P 阶段),对应三个具体问题:①用户是否会浏览一个物品?②用户浏览一个物品后,会给它评多少分?③用户最终是否会购买该物品?

3.2.1 模型介绍

CoFiToR 算法的结构示意图如图 3-2 所示。它包含浏览、评分和购买三个阶段，从粗粒度到细粒度以递进的方式来建模用户对物品的偏好，每个阶段都代表用户购物流程中的一个步骤。具体而言，浏览阶段旨在提取出用户最有可能会浏览的物品；接着，在评分阶段，它通过预测用户对可能会浏览的物品的评分，来进一步优化浏览阶段得到的候选物品列表；最后，在购买阶段，它通过学习用户对物品的购买偏好，从评分阶段得到的候选物品列表筛选出用户最有可能会购买的物品，从而组成最终的候选物品列表。可以看出，CoFiToR 的三个阶段依次对用户的购物过程进行建模分析，具有递进关系。在不同阶段之间，它通过候选物品列表的形式共享和传递知识，并通过不断优化候选物品列表来提升推荐性能。

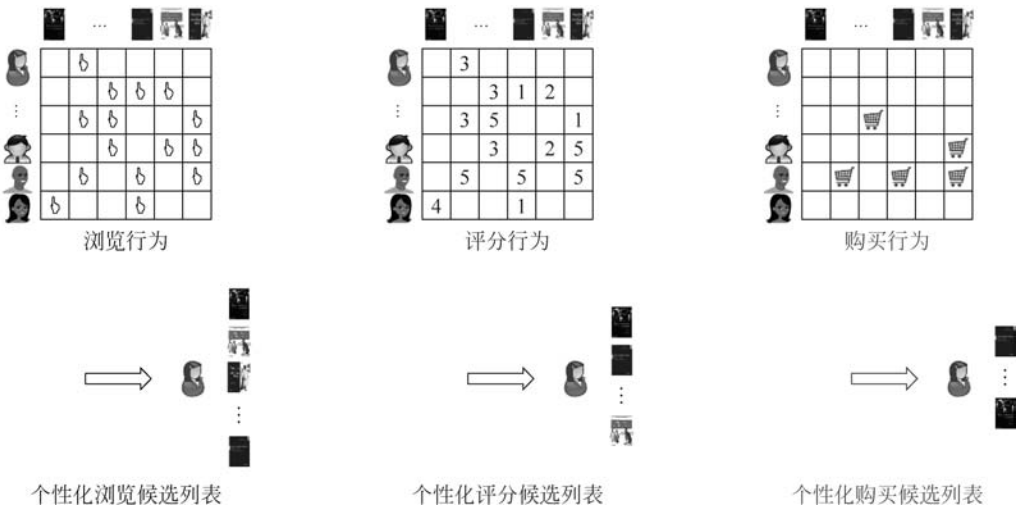

图 3-2 CoFiToR 算法的模型结构示意图(见彩插)

CoFiToR 的优化问题可以表示如下：

$$\mathrm{Prob}(\mathcal{E},\mathcal{S},\mathcal{P}|\mathcal{R}) \tag{3-1}$$

其中，$\mathcal{E}$、$\mathcal{S}$和$\mathcal{P}$分别表示基于原始评分记录生成的用于模拟浏览、评分和购买行为的数据。为了解决上面的优化问题，CoFiToR 将其转换成三个子问题，并得到包含三个阶段的解决方案：

$$\mathrm{Prob}(\mathcal{E}|\Theta_{\mathrm{E}}) \rightarrow \mathrm{Prob}(\mathcal{S}|\Theta_{\mathrm{S}};\ \mathcal{L}_{\mathrm{E}}) \rightarrow \mathrm{Prob}(\mathcal{P}|\Theta_{\mathrm{P}};\ \mathcal{L}_{\mathrm{S}}) \tag{3-2}$$

其中，Θ_{E}、Θ_{S} 和 Θ_{P} 是要学习的模型参数；$\mathcal{L}_{\mathrm{E}}$是浏览阶段输出的候选物品列表；$\mathcal{L}_{\mathrm{S}}$是评分阶段输出的候选物品列表。下面，我们将详细描述 CoFiToR 模型的三个阶段。

1. 浏览阶段

浏览阶段的目标是找出用户最有可能浏览的物品。为此，CoFiToR 将用户所有的评分记录$\mathcal{R}$当作用户的浏览行为$\mathcal{E}$(如图 3-2 所示)，即$\mathcal{E}=\{(u,i)|(u,i,r_{ui})\in\mathcal{R}\}$，目标函数的表示如下：

$$\mathrm{Prob}(\mathcal{E}|\ \Theta_{\mathrm{E}}) \tag{3-3}$$

其中，Θ_{E} 为浏览阶段要学习的模型参数。

为了提取用户可能浏览的物品，CoFiToR 在浏览阶段采用 BPR 算法[3]。BPR 算法基于成对的偏好假设，将所有观测到的记录都当作正样本，将未观测到的记录当作负样本，通过建模用户对正样本和负样本的偏好之间的差异来对物品进行排序，其目标函数如下：

$$\min_{\Theta}\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{J}_u^{\mathrm{P}}}\sum_{j\in\mathcal{J}\setminus\mathcal{J}_u^{\mathrm{P}}} f_{uij} \tag{3-4}$$

其中，$f_{uij}=-\ln\sigma(\hat{r}_{ui}-\hat{r}_{uj})+\frac{\alpha_{\mathrm{u}}}{2}\|\boldsymbol{U}_{u\cdot}\|^2+\frac{\alpha_{\mathrm{v}}}{2}\|\boldsymbol{V}_{i\cdot}\|^2+\frac{\alpha_{\mathrm{v}}}{2}\|\boldsymbol{V}_{j\cdot}\|^2+\frac{\beta_{\mathrm{v}}}{2}b_i^2+\frac{\beta_{\mathrm{v}}}{2}b_j^2$ 是定义在(u,i,j)三元组上的目标函数；$\sigma(\cdot)$是 sigmoid 函数。用户 u 对物品 i 评分的预测公式如下：

$$\hat{r}_{ui}=\boldsymbol{U}_{u\cdot}\boldsymbol{V}_{i\cdot}^{\mathrm{T}}+b_i \tag{3-5}$$

其中，$\boldsymbol{U}_{u\cdot}$ 和 $\boldsymbol{V}_{i\cdot}$ 分别表示用户 u 和物品 i 的潜在特征；b_i 表示物品 i 的偏置信息。

对于协同排序问题，直接使用 BPR 算法可能会造成信息损失，因为它平等地对待所有评分记录，没有考虑用户对物品的评分信息。为了赋予高分的物品更多的权重，CoFiToR 将物品的评分信息融入 BPR 的目标函数中，获得的目标函数如下：

$$\min_{\Theta}\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{J}_u^{\mathrm{E}}}\bar{r}_{ui}\sum_{j\in\mathcal{J}\setminus\mathcal{J}_u^{\mathrm{E}}} f_{uij} \tag{3-6}$$

其中，$\mathcal{J}_u^{\mathrm{E}}$是训练集中用户 u 浏览过的物品集合；$\bar{r}_{ui}=(2^{r_{ui}}-1)/2^5$ 是用户 u 对物品 i 的评分 r_{ui} 的归一化值[4]，5 代表最高的评分值。通过引入归一化评分 $\bar{r}_{ui}$，不仅增加了正负样本之间的差异，而且有助于区分不同分数的正样本。

通过利用修正的 BPR 算法，CoFiToR 在浏览阶段获得了候选物品列表$\mathcal{L}_{\mathrm{E}}$，该列表将作为评分阶段的输入，以继续对其进行优化。

2. 评分阶段

在浏览阶段得到用户可能浏览的物品后，下一步需要估计用户对物品的评分值，这也是 CoFiToR 在评分阶段的目标。具体来说，CoFiToR 将用户对物品的评分记录$\mathcal{R}$当作用户的评分行为$\mathcal{S}$(如图 3-2 所示)，即$\mathcal{S}=\mathcal{R}$，目标函数的表示如下：

$$\mathrm{Prob}(\mathcal{S}|\Theta_{\mathrm{S}};\ \mathcal{L}_{\mathrm{E}}) \tag{3-7}$$

其中，Θ_{S} 是评分阶段要学习的模型参数；$\mathcal{L}_{\mathrm{E}}$是浏览阶段得到的候选物品列表。

为了预测用户对物品的评分，在评分阶段，CoFiToR 采用 PMF 算法[5]来学习用户和物品的潜在特征。具体来说，它将用户对物品的偏好建模成用户潜在特征向量和物品潜在特征向量的内积，目标函数如下：

$$\min_{\Theta}\sum_{u=1}^{n}\sum_{i=1}^{m} y_{ui}\left[\frac{1}{2}(r_{ui}-\hat{r}_{ui})^2+\frac{\alpha_{\mathrm{u}}}{2}\|\boldsymbol{U}_{u\cdot}\|^2+\frac{\alpha_{\mathrm{v}}}{2}\|\boldsymbol{V}_{i\cdot}\|^2\right] \tag{3-8}$$

预测公式为：

$$\hat{r}_{ui}=\boldsymbol{U}_{u\cdot}\boldsymbol{V}_{i\cdot}^{\mathrm{T}}+b_u+b_i+\mu \tag{3-9}$$

其中，$\boldsymbol{U}_u$. 和 $\boldsymbol{V}_i$. 分别表示用户 u 和物品 i 的潜在特征向量；b_u 和 b_i 分别表示用户 u 和物品 i 的偏置信息；μ 表示全局平均偏好。

在评分阶段，通过利用 PMF 算法，CoFiToR 可以得到用户对物品的预测评分，然后它对浏览阶段得到的候选物品列表$\mathcal{L}_{\mathrm{E}}$进行重排(re-ranking)，同时移除排序靠后的物品，从而得到大小为 N_{S}($N_{\mathrm{S}}<N_{\mathrm{E}}$)的新候选物品列表$\mathcal{L}_{\mathrm{S}}$，该列表会作为购买阶段的输入以进一步对其进行优化。值得注意的是，在 ToR[1] 中，$\mathcal{L}_{\mathrm{S}}$就是最终的候选物品列表，它会被直接推荐给用户，而 CoFiToR 会继续对其进行优化，这也是 CoFiToR 与 ToR 最主要的区别。

需要说明的是，如果利用 PMF 的预测结果对所有未评分的物品(而非浏览阶段得到的候选物品列表$\mathcal{L}_{\mathrm{E}}$)进行排序，结果往往会非常差。主要有两个原因，首先，PMF 在建模过程中没有利用未观测到的(用户，物品)对，即没有考虑大量的负反馈信息，这样不利于算法找出用户感兴趣的物品；其次，最小化平方误差$(r_{ui}-\hat{r}_{ui})^2$ 不能有效地区分比真值 r_{ui} 大一点和小一点的预测值 $\hat{r}_{ui}$ 的好坏，但预测值的相对大小在排序中是非常重要的，即存在建模目标与排序目标的不一致性。ToR 和 CoFiToR 中的重排策略巧妙地规避了以上两个问题，且第二阶段的逐点偏好学习算法又能与第一阶段的成对偏好学习算法形成一定的互补。

3. 购买阶段

在评分阶段得到用户对物品的预测评分后，需要知道用户是否会购买该物品，这也是 CoFiToR 在购买阶段要解决的问题。为了提取用户最有可能会购买的物品，CoFiToR 将评分记录$\mathcal{R}$中用户对物品的评分为 5 的记录(5 为分数的最大值)当作用户的购买行为(如图 3-2 所示)，即$\mathcal{P}=\{(u,i)\mid(u,i,5)\in\mathcal{R}\}$，目标函数的表示如下：

$$\mathrm{Prob}(\mathcal{P}\mid\Theta_{\mathrm{P}};\ \mathcal{L}_{\mathrm{S}}) \tag{3-10}$$

其中，Θ_{P} 是在购买阶段要学习的模型参数；$\mathcal{L}_{\mathrm{S}}$是评分阶段得到的候选物品列表。

为了提取用户最有可能购买的物品，CoFiToR 在购买阶段再次采用 BPR 算法，原因在于基于成对偏好学习的 BPR 与基于逐点偏好学习的 PMF 有较强的互补性，这也在 ToR 中得到了证实[1]。注意，购买阶段使用的是原始的 BPR 算法，即式(3-4)，没有融入用户的评分信息，这与浏览阶段的不同。

在购买阶段，为了识别用户可能会购买的物品，CoFiToR 根据 BPR 的预测评分来对$\mathcal{L}_{\mathrm{S}}$中的候选物品列表再次进行重排，并移除排序靠后的物品，最后得到大小为 N_{P}($N_{\mathrm{P}}<N_{\mathrm{S}}<N_{\mathrm{E}}$)的候选物品列表$\mathcal{L}_{\mathrm{P}}$，即最终的物品推荐列表。

3.2.2 算法流程

CoFiToR 的算法流程如算法 3-1 所示。它包含三个阶段，不同阶段之间通过候选物品列表来实现知识的共享和迁移。需要说明的是，CoFiToR 的三个阶段是相对独立的，因此，每个阶段的模型可以根据实际需要进行更改，阶段的数量也可以灵活增减。

算法 3-1　CoFiToR 算法

1. **E 阶段**
2. 输入：浏览行为数据 $\mathcal{E}=\{(u,i)|(u,i,r_{ui})\in\mathcal{R}\}$
3. 输出：候选物品列表$\mathcal{L}_{\mathrm{E}}$
4. 优化式(3-6)的目标函数
5. 根据式(3-5)计算(u,i)对的浏览概率
6. 为每个用户生成大小为 N_{E} 的候选列表$\mathcal{L}_{\mathrm{E}}$
7. **S 阶段**
8. 输入：评分行为数据$\mathcal{S}=\mathcal{R}$，E 阶段的候选列表$\mathcal{L}_{\mathrm{E}}$
9. 输出：候选物品列表$\mathcal{L}_{\mathrm{S}}$
10. 优化式(3-8)的目标函数
11. 根据式(3-9)计算(u,i)对的预测评分
12. 为每个用户生成大小为 N_{S} 的候选列表$\mathcal{L}_{\mathrm{S}}$
13. **P 阶段**
14. 输入：购买行为数据$\mathcal{P}=\{(u,i)|(u,i,5)\in\mathcal{R}\}$，S 阶段的候选列表$\mathcal{L}_{\mathrm{S}}$
15. 输出：候选物品列表$\mathcal{L}_{\mathrm{P}}$
16. 优化式(3-4)的目标函数
17. 根据式(3-5)计算(u,i)对的购买概率
18. 为每个用户生成大小为 N_{P} 的候选列表$\mathcal{L}_{\mathrm{P}}$

3.2.3　代码实现

CoFiToR 的算法代码采用 Java 语言编写，工具是 JDK 1.8 和代码编辑器 Eclipse。它具有三个阶段。第一阶段采用加强版本的 BPR 算法，其中采样、对评分进行归一化和计算损失函数的代码如下：

```
public static void train() throws IOException
    {
        for (int iter = 0; iter < num_iterations; iter++){
            for (int iter2 = 0; iter2 < num_train; iter2++){
                //随机采样一个(u,i,r_ui)三元组
                int idx = (int) Math.floor(Math.random() * num_train);
                int u = indexUserTrain[idx];
                int i = indexItemTrain[idx];
                float rating = indexRatingTrain[idx];

                //对评分进行归一化
                float r_ui = 0f;
                if(rtype == 5){
                    int loc = (int)rating;
                    r_ui = rating_weight[loc];
                }
```

```
                else if(rtype == 10){
                    int loc = (int)(rating * 2);
                    r_ui = rating_weight[loc];
                }

                //获取训练集中用户 u 的评分记录
                Map < Integer, Float > Item_Rating = TrainData.get(u);
                int j = i;
                while (true){
                    //随机采样一个物品 j
                    j = (int) Math.floor(Math.random() * m) + 1;
                    //判断物品 j 是否是负样本
                    if (ItemSetTrain.contains(j) && !Item_Rating.containsKey(j))
                    {
                        break;
                    } else {
                        continue;
                    }
                }

                //计算损失函数
                float r_uij = biasV[i] - biasV[j];
                for (int f = 0; f < d; f++){
                    r_uij += U[u][f] * (V[i][f] - V[j][f]);
                }
                float EXP_r_uij = (float) Math.pow(Math.E, r_uij);
                float loss_uij = -1f / (1f + EXP_r_uij);
            }
        }
    }
```

第二阶段采用 PMF 算法，其中采样和计算损失函数的代码如下：

```
public static void train()
{
    for (int iter = 0; iter < num_iterations; iter++){
        for (int iter_rand = 0; iter_rand < num_train_target; iter_rand++){
            //随机采样一个(u,i,r_ui)三元组
            int rand_case = (int) Math.floor(Math.random() * num_train_target);
            int userID = indexUserTrain[rand_case];
            int itemID = indexItemTrain[rand_case];
            float rating = ratingTrain[rand_case];

            //计算预测公式和误差
            float pred = 0;
            float err = 0;
            for (int f = 0; f < d; f++){
                pred += U[userID][f] * V[itemID][f];
            }
            pred += g_avg + biasU[userID] + biasV[itemID];
```

```
                err = rating - pred;
            }
        }
    }
```

第三阶段采用原始的 BPR 算法，可以参考第一阶段的代码实现。

3.2.4　实验设置

对于实验参数的设置，CoFiToR 设置潜在特征向量的维度 $d=100$，学习率 $\gamma=0.01$，迭代次数 T 从{100,500,1000}中选取，正则化项上的权衡参数 α 从{0.1,0.01,0.001}中选取。根据验证集上的 NDCG@5 指标选择最优的参数值。

3.2.5　讨论

CoFiToR 算法由三个阶段组成，分别从浏览、评分和购买三种不同但相关的视角看待用户的评分记录。每个阶段都代表用户购物流程中的一个步骤，从粗粒度到细粒度逐步建模用户的偏好，不同阶段之间通过候选物品列表的形式来实现知识的共享与迁移。与 ToR 模型相比，CoFiToR 模型对用户的购买行为建模，因此能够更准确地捕捉用户的偏好信息。需要说明的是，分阶段建模（例如检索、匹配、重排、规则过滤等）在实际应用中也是一种比较常用的做法，因此 CoFiToR 模型可以看作工业界推荐系统的一个缩影。

CoFiToR 是一个基于迁移学习的框架，其基础模型可以替换为其他模型，因此，在未来工作中，可以将其他互补性较强的算法整合到 CoFiToR 框架中，以进一步提升其推荐性能。此外，CoFiToR 使用的模型要么基于单点偏好，要么基于成对偏好，未来可以考虑使用信息检索中常用的基于成列偏好的模型[6-7]。

3.3　上下文感知协同排序

结合多类偏好上下文的矩阵分解（matrix factorization with multiclass preference context，MF-MPC）算法[8]将偏好上下文定义为用户已评分的物品，并将其结合到传统的矩阵分解模型中以解决评分预测问题。受到 MF-MPC 算法的启发，上下文感知协同排序（context-aware collaborative ranking，CCR）算法[9]通过在用户偏好的建模过程中融入上下文信息来提高推荐效果。具体而言，它首先从用户的原始评分记录中提取隐含的上下文作为辅助信息，然后将其融入用户偏好的建模过程中，以预测用户对物品的喜好。

3.3.1　模型介绍

CCR 算法的示意图如图 3-3 所示，可以看到它从一个新的视角对评分数据进行建模。具体来说，它将原始的评分矩阵拆分成两个矩阵，一个表示用户的正负反馈，另一个表示用户的偏好上下文；然后，它通过一个对数几率损失函数来连接这两个矩阵，并从中学习用户和物品的特征；最后，它根据预测的用户对物品的偏好来对物品进行排序，从而得到最终的物品推荐列表。

CCR 算法的建模原理如下。它首先计算每个用户的平均评分值作为其个性化的阈

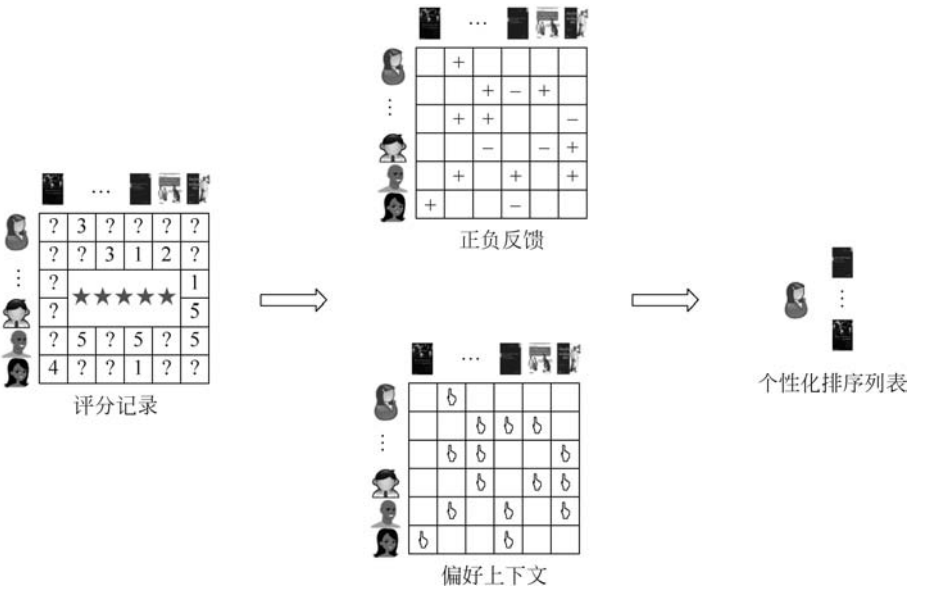

图 3-3 CCR 算法的模型结构示意图(见彩插)

值,即 $\bar{r}_u$,然后将用户对物品的原始评分值按照以下方式转换成二值评分:

$$y_{ui}=\begin{cases}+1, & r_{ui}\geqslant\bar{r}_u\\ -1, & r_{ui}<\bar{r}_u\end{cases} \tag{3-11}$$

具体而言,它将 $y_{ui}=+1$ 的记录当作正样本,并将其加入正样本集合 $\mathcal{S}^{\mathrm{P}}$ 中,其他样本则当作负样本加入负样本集合 $\mathcal{S}^{\mathrm{N}}$ 中。此外,它还随机采样了一部分未观测到的记录加入负样本集合中作为补充,其中未观测样本的采样数是观测样本的 ρ 倍(ρ 为预设的整数)。因此,负样本集合 $\mathcal{S}^{\mathrm{N}}$ 由两部分组成,一部分是评分 $r_{ui}<\bar{r}_u$ 的观测记录,另一部分是采样得到的未观测到的样本(u,i),其中 $y_{ui}=-1$。需要注意的是,在训练 CCR 时,每次迭代都需要从集合 $\mathcal{S}^{\mathrm{P}}\cup\mathcal{S}^{\mathrm{N}}$ 中随机采样一个(u,i,y_{ui})三元组。

为了进一步利用用户的评分记录,CCR 采用与 MF-MPC[8] 类似的方法,从评分记录中挖掘出有价值的信息,即用户的偏好上下文,以准确地建模用户对物品的偏好。在实际应用中,当用户决定是否购买某个物品时,通常会根据自己之前的购物经历来作决定,即用户对历史物品的偏好会影响其对未交互物品的态度。因此,参照 MF-MPC[8],CCR 在传统的矩阵分解模型中融入用户的偏好上下文:

$$\hat{r}_{ui}=(\boldsymbol{U}_{u\cdot}+\bar{\boldsymbol{U}}_{u\cdot}^{\mathrm{MPC}})\boldsymbol{V}_{i\cdot}^{\mathrm{T}}+b_u+b_i+\mu \tag{3-12}$$

其中,$\bar{\boldsymbol{U}}_{u\cdot}^{\mathrm{MPC}}$ 表示用户 u 的内在偏好上下文,其定义如下:

$$\bar{\boldsymbol{U}}_{u\cdot}^{\mathrm{MPC}}=\sum_{r\in\mathbb{M}}\frac{1}{\sqrt{|\mathcal{J}_u^r\backslash\{i\}|}}\sum_{i'\in\mathcal{J}_u^r\backslash\{i\}}\boldsymbol{M}_{i'\cdot}^r \tag{3-13}$$

其中,$\boldsymbol{M}_{i'\cdot}^r$ 是评分值为 r 的物品 i' 的潜在特征向量。由式(3-13)可得,所有的上下文物品根据用户对它们的评分被分为$|\mathbb{M}|$组,这有助于学习不同评分值物品的特征,以及用户对

不同评分值物品的偏好。

为了生成每个用户的物品推荐列表，CCR 使用对数几率损失函数来最小化真实值和预测值之间的误差，形式化如下：

$$l(\hat{r}_{ui}, r_{ui}) = -\ln \sigma(y_{ui}\hat{r}_{ui}) \tag{3-14}$$

如果(u, i, r_{ui})是正样本，$y_{ui}=1$；否则，$y_{ui}=-1$。

最后，CCR 的目标函数形式化如下：

$$\arg\min_{\Theta} \frac{1}{n}\sum_{u=1}^{n}\sum_{i=1}^{m} l(\hat{r}_{ui}, r_{ui}) + \mathrm{reg}(u, i) \tag{3-15}$$

其中，$\mathrm{reg}(u,i)=\frac{\lambda}{2}(\|\boldsymbol{U}_{u\cdot}\|^2+\|\boldsymbol{V}_{i\cdot}\|^2+b_u^2+b_i^2+\sum_{r\in\mathbb{M}}\sum_{i'\in\mathcal{J}_u^r\setminus\{i\}}\|\boldsymbol{M}_{i'\cdot}^r\|^2)$是正则化项，用于防止过拟合；$\Theta=\{\boldsymbol{U}_{u\cdot}, \boldsymbol{V}_{i\cdot}, b_u, b_i, \mu, \boldsymbol{M}_{i'\cdot}^r \mid u=1,2,\cdots,n; i=1,2,\cdots,m; r\in\mathbb{M}\}$为要学习的模型参数。

3.3.2　算法流程

CCR 的算法流程如算法 3-2 所示。它由参数初始化、评分转换和模型训练三部分组成，并将每个用户和每个物品的潜在特征和偏置项初始化为随机值。

算法 3-2　CCR 算法

输入：评分记录 $\mathcal{R}$
输出：Top-K 物品推荐列表
1. 初始化所有参数
2. **for** $u=1,2,\cdots,n$ **do**
3.　　计算用户 u 的平均评分 $\bar{r}_u$
4. **end for**
5. **for** each$(u,i,r_{ui})\in\mathcal{R}$ **do**
6.　　**if** $r_{ui}\geqslant\bar{r}_u$ **then**
7.　　将$(u,i,+1)$加入正样本集合$\mathcal{S}^{\mathrm{P}}$
8.　　**else**
9.　　将$(u,i,-1)$加入负样本集合$\mathcal{S}^{\mathrm{N}}$
10.　　**end if**
11. **end for**
12. **for** $t=1,2,\cdots,T$ **do**
13.　　有放回地随机采样 $\rho|\mathcal{R}|$个未观测的样本，并加入负样本集合$\mathcal{S}^{\mathrm{N}}$中
14.　　**for** $t_2=1,2\cdots,|\mathcal{S}^{\mathrm{P}}\cup\mathcal{S}^{\mathrm{N}}|$ **do**
15.　　随机采样一个(u,i,y_{ui})三元组
16.　　根据式(3-13)计算$\bar{\boldsymbol{U}}_{u\cdot}^{\mathrm{MPC}}$
17.　　根据式(3-12)计算 $\hat{r}_{ui}$
18.　　更新模型参数 Θ
19.　　**end for**
20. **end for**

3.3.3 代码实现

CCR 的算法代码采用 Java 语言编写，工具是 JDK 1.8 和代码编辑器 Eclipse。其中计算用户的内在偏好上下文的代码如下：

```
public static void CCR(float yui, int u, int i)
    {
        HashMap<Float, HashSet<Integer>>  rating_itemSet = Data.TrainData.get(u);

        //计算 Uu_bar_MPC
        float[]Uu_bar_MPC = new float[Data.d];

        for (float key : rating_itemSet.keySet()){
            HashSet<Integer>  itemSet = rating_itemSet.get(key);
            int rating_size = itemSet.size();

            //计算用户的评分数量,不包含物品 i
            if(itemSet.contains(i))
                rating_size = rating_size - 1;
            if(rating_size == 0)
                continue;

            float[]temp = new float[Data.d]; //用于存储 M 中的潜在特征向量的和
                if(Data.rtype == 10){
                    for(int itemID : itemSet){
                        if(itemID == i) //排除物品 i
                            continue;
                        for(int f = 0; f < Data.d; f++){
                            int pos = (int)(key * 2);
                            temp[f] = temp[f] + Data.M[pos][itemID][f];
                        }
                    }
                }
                else{
                    for(int itemID : itemSet){
                        if(itemID == i) //排除物品 i
                            continue;
                        for(int f = 0; f < Data.d; f++){
                            int pos = (int)key;
                            temp[f] = temp[f] + Data.M[pos][itemID][f];
                        }
                    }
                }

            for(int f = 0; f < Data.d; f++){
                 Uu_bar_MPC[f] = Uu_bar_MPC[f] + (float)(temp[f]/Math.sqrt((float)
rating_size)) ;
```

```
            }
        }

        //计算 rui
        float rui_pre = Data.biasU[u] + Data.biasV[i] + Data.mu;
        for (int f = 0; f < Data.d; f++)
        {
            rui_pre += (Data.U[u][f] + Uu_bar_MPC[f]) * Data.V[i][f];
        }

        //计算损失函数
        float e_ui = -yui * Sigmoid.sigmoid(-yui * rui_pre);
    }
```

3.3.4　实验设置

对于实验参数的设置，CCR 设置潜在特征向量的维度 $d=20$，学习率 $\gamma=0.01$ 以及采样大小 $\rho=3$，迭代次数 T 从{100，500，1000}中选取，正则化项上的权衡参数 α 从{0.1，0.01，0.001}中选取。根据验证集上的 NDCG@5 指标选择最优参数值。

3.3.5　讨论

CCR 算法从原始评分记录中挖掘出偏好上下文，即用户的历史交互物品，并将其融入用户偏好的建模过程中，有效地利用了他们的评分偏好信息。此外，它采用对数几率损失函数，并设计了一种改进的采样策略来帮助区分正负反馈，进而为用户提供精准的推荐。

CCR 通过线性相加的方式将内在偏好上下文结合到目标函数中，在未来工作中，可以探索更加有效的方式来整合偏好上下文。此外，可以考虑将偏好上下文应用到深度学习的框架中，以更好地学习用户的潜在特征。

3.4　整全迁移排序

在 3.2 节，我们提到 CoFiToR 模型[2]采用三阶段的方式来对用户行为进行建模，但这可能会给模型部署带来不便，因为它需要进行多次训练。为了解决此问题，整全迁移排序框架(holistic transfer to rank，HoToR)[10]被提出，其主要思想是将用户的原始评分记录转换为浏览行为数据、评分行为数据和购买行为数据，并通过知识共享和迁移把它们结合到一个整全的框架中，从而得到一个仅包含一个阶段的解决方案。

3.4.1　模型介绍

HoToR 模型的示意图如图 3-4 所示。它首先将用户的原始评分记录转换为三种不同但相关的视角，即浏览、评分和购买。然后，与两阶段的 ToR 和三阶段的 CoFiToR 不同，HoToR 将这三种视图结合起来，使得知识可以在不同行为之间共享，最终得到一个仅包含

一个阶段的解决方案。更具体地说，HoToR 将用户的评分行为数据（即第二种行为数据）作为浏览行为数据（即第一种行为数据）的加权策略，然后通过一种整全的方式来结合加权的浏览行为数据和购买行为数据（即第三种行为数据）。基于此框架，HoToR 分别使用基于矩阵分解的方法和基于邻域的方法作为基本模型，并实例化了两个算法，即基于模型的整全迁移排序算法和基于邻域的整全迁移排序算法，下面我们将详细介绍这两个算法。

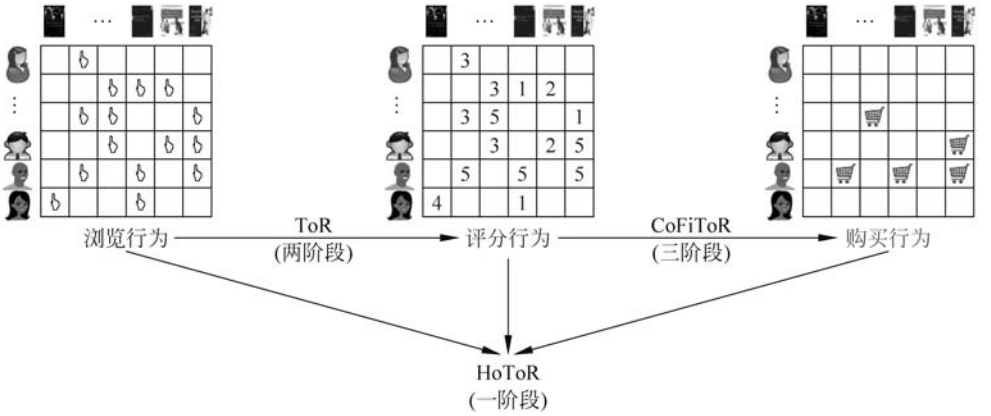

图 3-4　HoToR 算法的模型结构示意图（见彩插）

3.4.2　基于模型的整全迁移排序

为了对图 3-4 中的浏览行为数据和购买行为数据建模，基于模型的整全迁移排序（HoToR(mod.)）算法采用了两种偏好假设，即用户喜欢其浏览过的物品多于其未浏览的物品，以及用户喜欢其购买过的物品多于其未购买的物品。具体来说，HoToR(mod.)算法采用 BPR 算法[3]中的成对学习方法来分别学习用户在这两种行为下的偏好，并通过引入权衡参数 $\lambda_{\text{mod.}}$ $(0\leqslant\lambda_{\text{mod.}}\leqslant 1)$来控制这两种行为数据的比例：

$$\lambda_{\text{mod.}}(u,i,j)^{\text{E}} \text{ vs. } (1-\lambda_{\text{mod.}})(u,i,j)^{\text{P}} \tag{3-16}$$

其中，$(u,i,j)^{\text{E}}\in\{(u,i,j)\mid(u,i)\in\mathcal{E}\setminus\mathcal{P};\ j\in\mathcal{J}\setminus\mathcal{J}_u^{\text{E}}\}$，$(u,i,j)^{\text{P}}\in\{(u,i,j)\mid(u,i)\in\mathcal{P};\ j\in\mathcal{J}\setminus\mathcal{J}_u^{\text{P}}\}$。注意，用户浏览的物品只包含该用户浏览但未购买的物品（即$\mathcal{E}\setminus\mathcal{P}$），而用户购买的物品则包含了该用户浏览且购买的物品（即$\mathcal{P}$）。还有一点需要注意的是，HoToR(mod.)将用户所有评过分的物品视为他们的浏览物品，并将评分为 5 分的物品视为他们的购买物品，即$\mathcal{E}=\{(u,i)\mid(u,i,r_{ui})\in\mathcal{R}\}$，$\mathcal{P}=\{(u,i)\mid(u,i,5)\in\mathcal{R}\}$。

为了进一步利用图 3-4 中的评分行为数据，HoToR(mod.)算法采用了 CoFiToR 模型中加权版本的 BPR 算法[2,4]，即式(3-6)，来强调具有较高分数的正样本的重要性。最后，结合式(3-16)中两种类型的三元组，得到整全的目标函数如下：

$$\min_{\Theta}\lambda_{\text{mod.}}\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{J}_u^{\text{E}}\setminus\mathcal{J}_u^{\text{P}}}\bar{r}_{ui}\sum_{j\in\mathcal{J}\setminus\mathcal{J}_u^{\text{E}}}f_{uij}+(1-\lambda_{\text{mod.}})\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{J}_u^{\text{P}}}\sum_{j\in\mathcal{J}\setminus\mathcal{J}_u^{\text{P}}}f_{uij} \tag{3-17}$$

其中，$\bar{r}_{ui}=(2^{r_{ui}}-1)/2^5$ 是(u,i)的权重[4]；f_{uij} 是 BPR 中的目标函数，用户对物品的预测偏好 $\hat{r}_{ui}$ 与式(3-5)相同。式(3-17)的左半部分捕捉了基于浏览和评分行为数据的偏好

假设，右半部分则捕捉了基于购买行为数据的偏好假设。

在式(3-17)中，当 $\lambda_{\text{mod.}}=0$ 时，HoToR(mod.)算法等价于基于购买行为数据的 BPR 算法，即 BPR(P)；当 $\lambda_{\text{mod.}}=1$ 并使用 $\mathcal{J}_u^{\text{E}}$ 代替 $\mathcal{J}_u^{\text{E}}\backslash\mathcal{J}_u^{\text{P}}$ 时，HoToR(mod.)算法等价于基于加权的浏览行为数据的 BPR 算法，即 W-BPR(E)；当 $0<\lambda_{\text{mod.}}<1$ 时，HoToR(mod.)算法将浏览、评分和购买三种行为数据整合在一个目标函数中。此外，对于每个用户 u，式(3-17)还考虑了两种类型的有序信息。一种位于式(3-17)的左边，即 $i\in\mathcal{J}_u^{\text{E}}\backslash\mathcal{J}_u^{\text{P}}$($0<r_{ui}<5$)和 $j\in\mathcal{J}\backslash\mathcal{J}_u^{\text{E}}$($r_{uj}$ 未知，即用户 u 没有给物品 j 评分)，另一种位于式(3-17)的右边，即 $i\in\mathcal{J}_u^{\text{P}}$($r_{ui}=5$)和 $j\in\mathcal{J}\backslash\mathcal{J}_u^{P}$($r_{uj}\neq5$)。

对于(u,i,j)三元组，在式(3-17)中 f_{uij} 的模型参数的梯度如下：

$$\nabla\boldsymbol{U}_{u\cdot}=\frac{\partial f_{uij}}{\partial\boldsymbol{U}_{u\cdot}}=-\sigma(\hat{r}_{uj}-\hat{r}_{ui})(\boldsymbol{V}_{i\cdot}-\boldsymbol{V}_{j\cdot})+\alpha_{\text{u}}\boldsymbol{U}_{u\cdot} \tag{3-18}$$

$$\nabla\boldsymbol{V}_{i\cdot}=\frac{\partial f_{uij}}{\partial\boldsymbol{V}_{i\cdot}}=-\sigma(\hat{r}_{uj}-\hat{r}_{ui})\boldsymbol{U}_{u\cdot}+\alpha_{\text{v}}\boldsymbol{V}_{i\cdot} \tag{3-19}$$

$$\nabla\boldsymbol{V}_{j\cdot}=\frac{\partial f_{uij}}{\partial\boldsymbol{V}_{j\cdot}}=-\sigma(\hat{r}_{uj}-\hat{r}_{ui})(-\boldsymbol{U}_{u\cdot})+\alpha_{\text{v}}\boldsymbol{V}_{j\cdot} \tag{3-20}$$

$$\nabla b_i=\frac{\partial f_{uij}}{\partial b_i}=-\sigma(\hat{r}_{uj}-\hat{r}_{ui})+\beta_{\text{v}}b_i \tag{3-21}$$

$$\nabla b_j=\frac{\partial f_{uij}}{\partial b_j}=-\sigma(\hat{r}_{uj}-\hat{r}_{ui})(-1)+\beta_{\text{v}}b_j \tag{3-22}$$

HoToR(mod.)的算法流程如算法 3-3 所示。它主要包含两层循环，其中内层循环使用采样权衡参数 $\lambda_{\text{mod.}}$ 来控制式(3-16)中浏览行为和购买行为三元组的采样比例，它们分别对应于式(3-17)中目标函数相加的左右部分。

算法 3-3　HoToR(mod.)算法

1. 输入：评分数据$\mathcal{R}$和采样权衡参数 $\lambda_{\text{mod.}}$
2. **for** $t=1,2,\cdots,T$ **do**
3.　　**for** $t_2=1,2,\cdots,|\mathcal{R}|$ **do**
4.　　随机在 $[0,1)$ 中产生随机数 r
5.　　**if** $r<\lambda_{\text{mod.}}$ **then**
6.　　随机采样一个(用户,物品)对$(u,i)\in\mathcal{E}\backslash\mathcal{P}$
7.　　随机采样一个物品 $j\in\mathcal{J}\backslash\mathcal{J}_u^{\text{E}}$
8.　　**else**
9.　　随机采样一个(用户,物品)对$(u,i)\in\mathcal{P}$
10.　　随机采样一个物品 $j\in\mathcal{J}\backslash\mathcal{J}_u^{\text{P}}$
11.　　根据采样得到的三元组(u,i,j)计算梯度
12.　　更新模型参数 Θ
13.　　**end for**
14. **end for**

值得注意的是，BPR++算法[11]也包含了这种具有两层循环和一个采样权衡参数的算法框架，它与 HoToR(mod.)的主要区别在于成对偏好学习中建模的三元组：

$$\lambda_{\text{BPR++}}(u,i,j)^{\text{E}'} \text{ vs. } (1-\lambda_{\text{BPR++}})(u,i,j)^{\text{R}} \tag{3-23}$$

其中，$(u,i,j)^{\text{E}'} \in \{(u,i,j) \mid (u,i) \in \mathcal{E};\ j \in \mathcal{J} \setminus \mathcal{J}_u^{\text{E}}\}$，$(u,i,j)^{\text{R}} \in \{(u,i,j) \mid (u,i,r_{ui}),\ (u,j,r_{uj}) \in \mathcal{R};\ r_{ui} > r_{uj}\}$。可以看到，式(3-23)和式(3-16)是不一样的。它们的区别主要有两点。首先，HoToR(mod.)关注建模用户对交互过物品与未交互物品的偏好，而BPR++侧重于捕捉用户对交互过物品的偏好。其次，HoToR(mod.)通过加权策略在目标函数中融入了评分行为数据，有助于准确地建模用户对物品的偏好。

3.4.3 基于邻域的整全迁移排序

除了基于模型的整全迁移排序算法，HoToR 还提出了一个基于邻域的整全迁移排序算法 HoToR(nei.)，该算法的基本思想是通过无缝地结合用户的浏览、评分和购买行为来学习他们对物品的偏好。具体来说，因为基于物品的单类协同过滤(item-based one-class collaborative filtering，IOCCF)算法[12]的有效性和简单性，HoToR(nei.)采用 IOCCF 算法来同时建模这三种行为数据，其中用户 u 对物品 i 的预测公式如下：

$$\hat{r}_{ui} = \lambda_{\text{nei.}} \sum_{k \in \mathcal{J}_u^{\text{E}} \cap \mathcal{N}_i^{\text{E}}} \bar{r}_{uk} s_{ki}^{\text{E}} + (1-\lambda_{\text{nei.}}) \sum_{k \in \mathcal{J}_u^{\text{P}} \cap \mathcal{N}_i^{\text{P}}} s_{ki}^{\text{P}} \tag{3-24}$$

其中，$\lambda_{\text{nei.}}$ ($0 \leqslant \lambda_{\text{nei.}} \leqslant 1$)是加权置信度，其既结合了式(3-24)左边基于浏览行为和评分行为的预测公式，又结合了右边基于购买行为的预测公式。在式(3-24)中，$\bar{r}_{uk} = (2^{r_{uk}} - 1)/2^5$ 表示基于用户 u 对物品 k 的评分 r_{uk} 所估计的权重[4]，用于缓解浏览行为中用户偏好的不确定性。$s_{ki}^{\text{E}} = \dfrac{|\mathcal{U}_k^{\text{E}} \cap \mathcal{U}_i^{\text{E}}|}{|\mathcal{U}_k^{\text{E}} \cup \mathcal{U}_i^{\text{E}}|}$ 和 $s_{ki}^{\text{P}} = \dfrac{|\mathcal{U}_k^{\text{P}} \cap \mathcal{U}_i^{\text{P}}|}{|\mathcal{U}_k^{\text{P}} \cup \mathcal{U}_i^{\text{P}}|}$ 分别表示物品 k 和物品 i 基于浏览行为数据和基于购买行为数据的相似度，使用杰卡德指数(Jaccard index)进行计算。$\mathcal{N}_i^{\text{E}}$ 和 $\mathcal{N}_i^{\text{P}}$ 表示物品 i 分别基于浏览行为数据和基于购买行为数据的最近邻居集合。

在式(3-24)中，当 $\lambda_{\text{nei.}} = 0$ 时，HoToR(nei.)算法仅使用购买行为数据 $\mathcal{P} = \{(u,i) \mid (u,i,5) \in \mathcal{R}\}$，式(3-24)等价于基于购买行为数据的 IOCCF 算法，即 IOCCF(P)，其中 $\hat{r}_{ui}^{\text{P}} = \sum_{k \in \mathcal{J}_u^{\text{P}} \cap \mathcal{N}_i^{\text{P}}} s_{ki}^{\text{P}}$；当 $\lambda_{\text{nei.}} = 1$ 时，HoToR(nei.)算法结合浏览行为数据 $\mathcal{E} = \{(u,i) \mid (u,i) \in \mathcal{R}\}$ 和评分行为数据 $\mathcal{S} = \mathcal{R}$，式(3-24)等价于基于加权的浏览行为数据的 IOCCF 算法，即 W-IOCCF(E)，其中 $\hat{r}_{ui}^{\text{E}} = \sum_{k \in \mathcal{J}_u^{\text{E}} \cap \mathcal{N}_i^{\text{E}}} \bar{r}_{uk} s_{ki}^{\text{E}}$；当 $0 < \lambda_{\text{nei.}} < 1$ 时，HoToR(nei.)算法利用浏览、评分和购买三种行为数据来学习用户的偏好。

3.4.4 代码实现

HoToR 的算法代码采用 Java 语言编写，工具是 JDK 1.8 和代码编辑器 Eclipse。其中 HoToR(mod.)采样和计算损失函数部分的核心代码如下：

```
public static void HoToR_mod_training()throws IOException{
        for(int iter = 0;iter < num_iterations;iter++){
            for(int iter2 = 0;iter2 < num_train;iter2++){
                int u;
                int i = 0;
                int j = 0;
                float r_ui;
                float bar_r_ui = 1f;
                //产生随机数
                float random = (float)Math.random();
                if(random < lambda_mod) {
                    //随机采样一个(用户,物品)对(u,i)∈ℰ\𝒫
                    int idx;
                    idx = (int) Math.floor(Math.random() * num_train_notFive);
                    u =  indexUserTrainClick[idx];
                    i =  indexItemTrainClick[idx];
                    HashMap<Integer,Float> itemRatingSet_u = TrainData.get(u);
                    r_ui = itemRatingSet_u.get(i);
                    bar_r_ui = (float)((Math.pow(2,r_ui) - 1)/Math.pow(2,5));
                    //随机采样一个物品 j∈𝒥\𝒥_u^E
                    while(true){
                        j = (int)Math.floor(Math.random() * m) + 1;
                        if(ItemTrainingSet.contains(j)&&!itemRatingSet_u.containsKey(j))
                            break;
                    }
                }else{
                    //随机采样一个(用户,物品)对(u,i)∈𝒫
                    int idx;
                    idx = (int) Math.floor(Math.random() * (num_train - num_train_
notFive));
                    u =  indexUserTrainPurchase[idx];
                    i =  indexItemTrainPurchase[idx];
                    bar_r_ui = 1;
                    HashMap<Integer,Float> itemRatingSet_u = TrainData.get(u);
                    //随机采样一个物品 j∈𝒥\𝒥_u^P
                    while(true){
                        j = (int)Math.floor(Math.random() * m) + 1;
                        if(ItemTrainingSet.contains(j)&&(!itemRatingSet_u.containsKey
(j)||itemRatingSet_u.get(j)!= 5.0))
                            break;
                    }
                }

                //计算预测公式和损失函数
                float r_uij = biasV[i] - biasV[j];
                for(int k = 0;k < d;k++){
                    r_uij = r_uij + U[u][k] * (V[i][k] - V[j][k]);
                }
```

```
                    float EXP_r_uij = (float)Math.pow(Math.E,r_uij);
                    float loss_uij =  -1f/(1f + EXP_r_uij);
                }
            }
    }
```

3.4.5 实验设置

对于实验参数的设置，HoToR(mod.)算法设置潜在特征向量的维度 $d=100$ 和学习率 $\gamma=0.01$，迭代次数 T 从{100,500,1000}中选取，正则化项上的权衡参数 α 从{0.1,0.01,0.001}中选取。HoToR(nei.)算法设置最近邻居的数量 $K=100$。权衡参数 $\lambda_{\text{mod.}}$ 和 $\lambda_{\text{nei.}}$ 选取的范围是{0.1,0.2,0.3,…,0.9}。根据验证集上的 NDCG@5 指标选取最优参数值。

3.4.6 讨论

HoToR 构建的是单阶段的算法，与包含两个阶段的 ToR 和包含三个阶段的 CoFiToR 相比，它更便于维护和部署。具体而言，HoToR 模型包含两种变体，即 HoToR(mod.)算法和 HoToR(nei.)算法。为了结合浏览、评分和购买三种类型的行为数据，HoToR(mod.)算法设计了一种新的采样策略，HoToR(nei.)算法则扩展了传统协同过滤方法的预测规则。通过这两种策略，HoToR 不仅利用了浏览行为和购买行为的互补性，而且通过将分数作为置信度权重利用了评分行为数据。

HoToR 将用户的历史评分记录转换为浏览、评分和购买三种行为数据，在未来工作中，可以考虑从其他新的视角来看待用户的历史评分记录，进而从用户的评分数据中学习更丰富的信息。此外，可以将这种看待评分记录的新颖视角应用到基于深度学习的模型中，以更好地捕捉用户对物品的偏好。

3.5 本章小结

本章讨论了推荐系统中的协同排序问题，即从显式反馈中学习用户对物品的偏好，从而为用户提供个性化的物品排序列表。我们介绍了三个具有代表性的算法，探讨了它们的模型、算法流程、代码实现和实验设置。具体而言，CoFiToR 从浏览、评分和购买的视角看待用户的历史评分记录，并通过顺序递进的方式对其进行建模，从而有效地捕捉了用户的偏好。CCR 在目标函数中结合了用户的内在偏好上下文，并设计了一种巧妙的采样策略来区分正负样本，从而能够准确地为用户推荐物品。HoToR 将用户的多种行为数据整合到一个联合函数中，不仅解决了多阶段模型训练的复杂性问题，而且能够利用完整的行为数据来建模用户的真实偏好。

此外，其他一些算法的目标也是为了解决协同排序问题。这些方法可以分为基于邻域的方法和基于模型的方法。基于邻域的方法根据用户或物品间的相似性来预测用户的偏好。例如，EigenRank[13]和 VSRank[14]关注建模用户对物品的成对偏好，其中 EigenRank

利用用户对物品排序的相关性来测量用户之间的相似度，而 VSRank 则将用户视为文档，并将用户交互过的物品视为词项，然后使用一种新的加权方案来对词项进行加权。为了降低基于成对偏好的方法的训练复杂度，成列协同过滤模型（listwise collaborative filtering，ListCF）[15]根据相似用户在物品排列上的概率分布，直接生成每个用户的物品排序列表。值得注意的是，许多研究人员将注意力转向了基于模型的方法，这类方法的代表性工作有 CoFiRank[16]，它同样基于成列偏好假设，但致力于优化 NDCG（normalized discounted cumulative gain）排序指标。为了准确地学习用户和物品的潜在特征，统一推荐模型（unified recommendation model，URM）[17]在一个框架中结合了基于逐点偏好假设的损失函数和基于成列偏好假设的损失函数。局部协同排序模型（local collaborative ranking，LCR）[18]假设评分矩阵为局部低秩（local low-rank），并结合局部低秩矩阵分解方法来最小化排序损失，而成列局部协同排序模型（listed LCR，LLCR）[19]在 LCR 的基础上进一步提升了模型的效率。此外，低值注入模型（low-value injection，l-injection）[20]旨在找到用户不感兴趣的物品并为其设置低分，不仅解决了数据稀疏的问题，也提高了推荐的准确性。

3.6　参考文献

[1] PAN W，YANG Q，DUAN Y，et al. Transfer learning for behavior ranking [J]. ACM Transactions on Intelligent Systems and Technology，2017，8(5)：65：1-65：23.

[2] DAI W，ZHANG Q，PAN W，et al. Transfer to rank for top-N recommendation [J]. IEEE Transactions on Big Data，2020，6(4)：770-779.

[3] RENDLE S，FREUDENTHALER C，GANTNER Z，et al. BPR：Bayesian personalized ranking from implicit feedback [C]//BILMES J A，NG A Y. Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI'09). Arlington：AUAI Press，2009：452-461.

[4] SHI Y，KARATZOGLOU A，BALTRUNAS L，et al. xCLiMF：Optimizing expected reciprocal rank for data with multiple levels of relevance [C]//YANG Q，KING I，LI Q，et al. Proceedings of the 7th ACM Conference on Recommender Systems (RecSys'13). New York：ACM，2013：431-434.

[5] SALAKHUTDINOV R，MNIH A. Probabilistic matrix factorization [C]//PLATT J C，KOLLER D，SINGER Y，et al. Proceeding of the 20th Annual Conference on Neural Information Processing Systems (NeurIPS'07). New York：Curran Associates Inc.，2007：1257-1264.

[6] CAO Z，QIN T，LIU T，et al. Learning to rank：From pairwise approach to listwise approach [C]//GHAHRAMANI Z. Proceedings of the 24th International Conference on Machine Learning (ICML'07). New York：ACM，2007：129-136.

[7] WU L，HSIEH C，SHARPNACK J. SQL-Rank：A listwise approach to collaborative ranking [C]//DY J，KRAUSE A. Proceedings of the 35th International Conference on Machine Learning (ICML'18). 2018：5311-5320.

[8] PAN W，MING Z. Collaborative recommendation with multiclass preference context [J]. IEEE Intelligent Systems，2017，32(2)：45-51.

[9] DAI W，PAN W，MING Z. Context-aware collaborative ranking [C]//WANG F L，XIE H，LAM W，et al. Proceedings of 15th Asia Information Retrieval Societies Conference (AIRS'19). Berlin：

Springer,2019: 43-55.

[10] MA W, LIAO X, DAI W, et al. Holistic transfer to rank for top-N recommendation [J]. ACM Transactions on Interactive Intelligent Systems, 2021, 11(1): 8: 1-8: 23.

[11] LERCHE L, JANNACH D. Using graded implicit feedback for Bayesian personalized ranking [C]//KOBSA A, ZHOU M X, ESTER M, et al. Proceedings of the 8th ACM Conference on Recommender Systems (RecSys'14). New York: ACM, 2014: 353-356.

[12] DESHPANDE M, KARYPIS G. Item-based top-N recommendation algorithms [J]. ACM Transactions on Information Systems, 2004, 22(1): 143-177.

[13] LIU N, YANG Q. EigenRank: A ranking-oriented approach to collaborative filtering [C]// MYAENG S, OARD D W, SEBASTIANI F, et al. Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'08). New York: ACM, 2008: 83-90.

[14] WANG S, SUN J, GAO B J, et al. VSRank: A novel framework for ranking-based collaborative filtering [J]. ACM Transactions on Intelligent Systems and Technology, 2014, 5(3): 51: 1-51: 24.

[15] HUANG S, WANG S, LIU T, et al. Listwise collaborative filtering [C]//BAEZA-YATES R, LALMAS MOUNIA, MOFFAT A, et al. Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'15). New York: ACM, 2015: 343-352.

[16] WEIMER M, KARATZOGLOU A, LE Q V, et al. COFI RANK-Maximum margin matrix factorization for collaborative ranking [C]//PLATT J C, KOLLER D, SINGER Y, et al. Proceedings of the 21st Annual Conference on Neural Information Processing Systems (NeurIPS'08). New York: Curran Associates Inc., 2008: 1593-1600.

[17] SHI Y, LARSON M, HANJALIC A. Unifying rating-oriented and ranking-oriented collaborative filtering for improved recommendation [J]. Information Science, 2013, 229: 29-39.

[18] LEE J, BENGIO S, KIM S, et al. Local collaborative ranking [C]//CHUNG C, BRODER A Z, SHIM K, et al. Proceedings of the 23rd International World Wide Web Conference (WWW'14). New York: ACM, 2014: 85-96.

[19] 刘海洋,王志海,黄丹,等.基于评分矩阵局部低秩假设的成列协同排名算法[J].软件学报,2015, 26(11): 2981-2993.

[20] LEE J, HWANG W, PARC J, et al. L-Injection: Toward effective collaborative filtering using uninteresting items [J]. IEEE Transactions on Knowledge and Data Engineering, 2019, 31(1): 3-16.

3.7 习题

1. 本章研究的问题(即基于显式反馈的物品排序问题)与第 2 章研究的问题(即基于显式反馈的评分预测问题)有哪些联系与区别?

2. CoFiToR 算法将用户的购物过程分为三个阶段,它是如何连接这些阶段的?

3. 可以将哪些算法用作 CoFiToR 算法中三个阶段的基础组件?

4. 在 CoFiToR 算法的三个阶段中,候选物品列表由长变短,最终得到 K 个物品(例如 $K=5$)。能否设计一种算法,候选物品列表由短变长,最终得到 K 个物品?请想一想,

关于构建候选物品列表的这两种方式，哪一种更加合理？

5. 除了 CCR 算法中使用的方法，你还能想出哪些方法可以来对用户偏好的上下文进行建模？

6. 请分析 CCR 算法的时间复杂度。

7. 多阶段的模型（例如 ToR 和 CoFiToR）存在哪些问题？HoToR 模型是如何解决这些问题的？

8. 请分析本章重点介绍的三个算法的优缺点。

9. 实现本章重点介绍的三个算法，并比较它们的推荐效果。

10. 在 RecSys、SIGIR 等知名国际会议的论文集中，查阅研究 CR 问题的前沿工作（近 3 年），总结其中的研究动机、主要思路、关键技术和对比实验。