

# 推荐系统

---

推荐系统是自动联系用户和物品的一种工具，它能够在信息过载的环境中帮助用户发现他们感兴趣的信心，也能将信息推送给它们感兴趣的兴趣用户。

## 推荐系统应用

---

- 电子商务
- 电影和视频网站推荐
- 个性化音乐台
- 社交网络
- 个性化阅读
- 基于位置的服务
- 个性化邮件
- 个性化广告

## 推荐系统评测

---

好的推荐系统不仅仅能够准确预测用户的行为，而且能够扩展用户的事业，帮助用户发现那些他们可能会感兴趣，但却不那么容易发现的东西。

### 实验方法

#### 1. 离线实验

- 1.1 获取用户历史行为
- 1.2 通过历史行为建模
- 1.3 对模型进行评测

#### 2. 用户调查

- 优点：可以获取一些离线实验无法得到的评测信息，如用户的惊喜度。
- 缺点：用户调查成本高，用户需要花大量时间完成一个任务，并且用户调查样本少时不能保证结果的统计意义。

#### 3. 在线实验

ABTest的方式对比不同的推荐算法。

### 评测指标

#### 1. 用户满意度

- 通过购买率度量用户的满意度
- 通过满意和不满意的反馈按钮
- 更一般情况，用点击率、用户停留时间和转化率等指标度量用户的满意度

#### 2. 预测准确度

评分预测：

一般通过均方根误差(RMSE)和平均绝对误差(MAE)计算。

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

MAE采用绝对值计算预测误差：

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

其中 $r_{ui}$ 是用户 $u$ 对物品 $i$ 的实际评分，而 $\hat{r}_{ui}$ 是推荐算法给出的预测评分。

TopN推荐：

TopN推荐的预测一般通过准确率(precision)/召回率(recall)度量。 $R(u)$ 是根据用户在训练集上的行为给用户作出的推荐按列表，而 $T(u)$ 是用户在测试集上的行为列表。

$$Recall = \frac{|R(u) \cap T(u)|}{\sum_{u \in U} |R(u)|}$$
$$Precision = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |R(u)|}$$

有时候为了全面评测TopN推荐的准确率和召回率，一般会选取不同的推荐列表长度 $N$ ，计算一组准确率和召回率，然后画出准确率和召回率的曲线。

### 3. 覆盖率

覆盖率描述一个推荐系统对物品长尾的发掘能力。最简单的定义为推荐系统能够推荐出物品占总物品集合的比例。假设用户集合为 $U$ ，推荐系统给每个用户推荐一个长度为 $N$ 的物品列表 $R(u)$ 。

$$Coverage = \frac{|\bigcup_{u \in U} R(u)|}{|I|}$$

为了更细致的描述推荐系统发掘长尾的能力，需要统计推荐列表中不同物品出现次数的分布，如果所有物品都出现在推荐系统中，并且出现的次数差不多，那么推荐系统发现长尾的能力就越好。

另外两种定义覆盖率的方法：

- 信息熵

$$H = - \sum_{i=1}^n p(i) \log p(i)$$

- Gini系数

$$G = \frac{1}{n-1} \sum_{j=1}^n (2j - n - 1) p(i_j)$$

### 4. 多样性

多样性描述了推荐列表中物品两两之间的不相似性。因此，多样性和相似性是对应的。 $s(i, j) \in [0, 1]$ 定义了物品 $i$ 和 $j$ 之间的相似度，那么用户 $u$ 的推荐列表 $R(u)$ 的多样性定义为：

$$Diversity(R(u)) = 1 - \frac{\sum_{i,j \in R(u), i \neq j} s(i, j)}{\frac{1}{2} |R(u)| (|R(u)| - 1)}$$

所有用户推荐列表的平均值：

$$Diversity = \frac{1}{|U|} \sum_{u \in U} Diversity(R(u))$$

关于推荐系统多样性最好达到什么程度，举例说明。假设用户80%的时间看动作片，20%的时间看动画片。4种不同的推荐列表：A列表中有10部动作片，没有动画片；B列表中有10动画，0动作；C列表8动作，2动画；D列表5动画，5动作。这个例子中，一般认为C列表是最好的，具有一定的多样性，又考虑到了用户的主要兴趣。

## 5. 新颖性

评测新颖度的最简单的方法是利用推荐结果的平均流行度，因为越不热门的物品越有可能让用户觉得新颖。

## 6. 总结

在给定覆盖率、多样性、新颖性等限制条件下，尽可能优化预测准确度。

max 预测准确度  
覆盖率>A  
多样性>B  
新颖性>C

## 评测维度

- 用户维度：主要包括用户的人口统计学信息、活跃度以及是不是新用户等
- 物品维度：物品的属性、流行度平均分以及是不是新加入的物品等
- 时间维度：包括季节，是否为工作日，白天还是晚上等。

如果能够在推荐系统的评测报告中包含不同维度下的系统评测指标，能帮我们找到一个看上去比较弱的算法的优势，发现一个看上去比较强的算法的缺点。

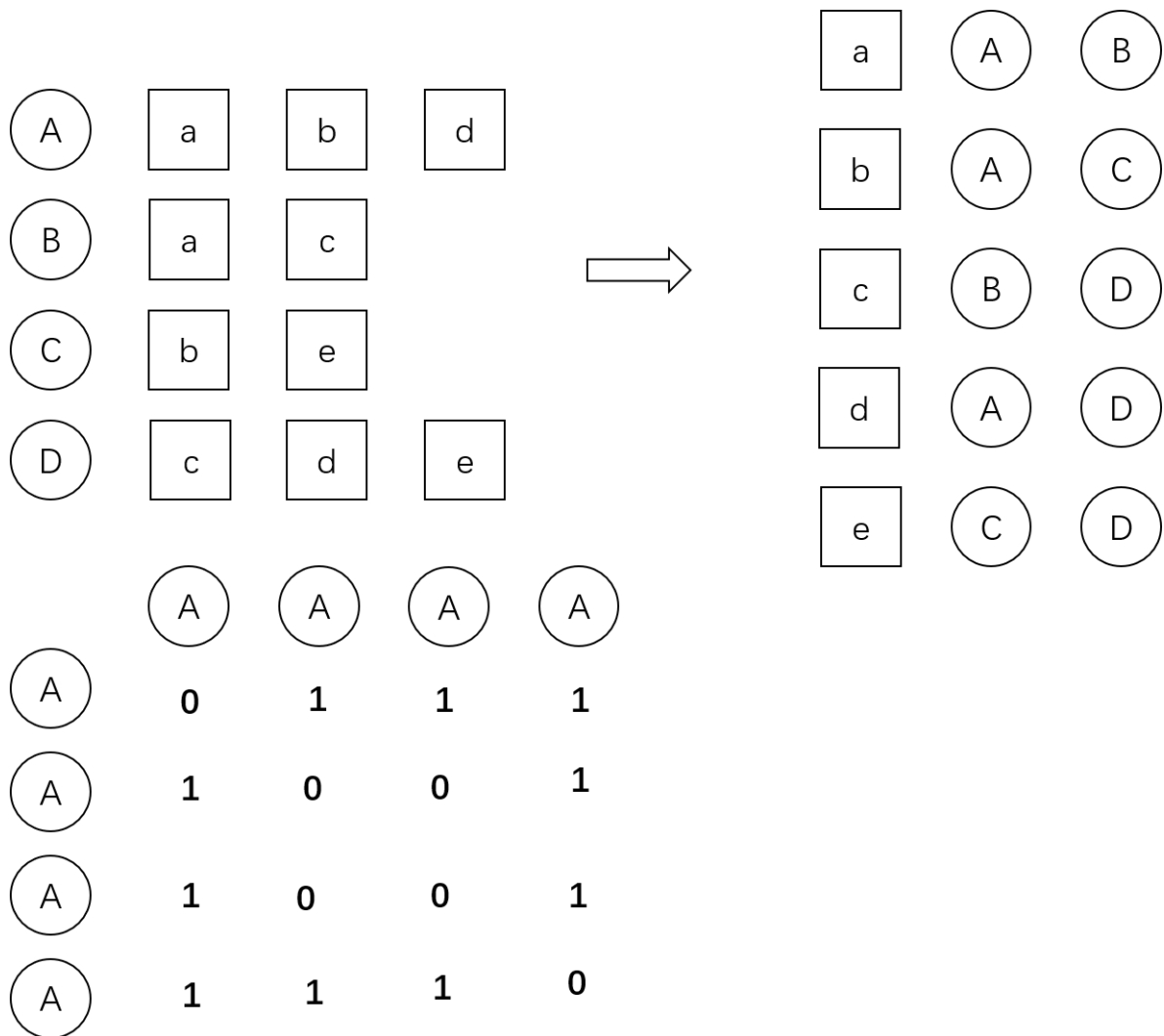
# 推荐系统四种基本算法

## 基于用户的协同过滤算法(collaborative filtering)

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。  
给定用户u和用户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。通过余弦相似度计算用户之间的相似性：

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

事实上，很多用户之间没有对同样的物品产生过行为，很多时候 $|N(u) \cap N(v)| = 0$ 。算法会将很多时间浪费在计算这种用户之间的相似度上。为此，可以首先建立物品到用户的倒排表，对于每个物品都保存对该物品产生过行为的用户列表。扫描倒排表中每个物品对应的用户列表，将用户列表中的每两个用户组合都算作一分，加到用户跟用户的相似矩阵中(初始值为0)，最终得到的用户跟用户的相似矩阵W为余弦相似度的分子部分，将W除以分母可以得到最终的用户兴趣相似度。



得到用户之间的兴趣相似度后，可以计算被推荐用户对每个物品感兴趣的程度：

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

其中， $S(u, K)$ 包含和用户 $u$ 兴趣最接近的 $K$ 个用户， $N(i)$ 是对物品 $i$ 有过行为的用户集合， $w_{uv}$ 是用户 $u$ 和用户 $v$ 的兴趣相似度， $r_{vi}$ 代表用户 $v$ 对物品 $i$ 的兴趣，如果是隐反馈数据，这个 $r_{vi} = 1$ 。计算出用户对物品的兴趣程度之后，选出该用户未购买过的TopN物品即可。

## 基于物品的协同过滤算法

1. 计算物品之间的相似度。
2. 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。

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

这里 $|N(i)|$ 变成了喜欢物品 $i$ 的用户数。 $w_{ij}$ 计算的方式也是先建立用户到物品的倒排表，再根据倒排表进行计算。

得到 $w_{ij}$ 之后，可以计算用户 $u$ 对一个物品 $j$ 的感兴趣程度：

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

$N(u)$ 是用户喜欢物品的集合， $S(j, K)$ 是和物品 $j$ 最相似的 $K$ 的物品的集合。这里跟UserCF有所不同。最终推荐的时候，对于用户历史上感兴趣的每一个物品，取出它们相似度最高的前 $K$ 个物品，共得到 $N(u) * K$ 个物品，排序后取去除用户产生行为后的TopN的物品作为推荐结果。

## 优化方法

1. UserCF算法中，用户u和v的公共物品，本来每次得分都为1，每多一个就多加一分，现在对每个物品的这1分进行惩罚，该物品购买的人越多，惩罚的力度越大。

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

ItemCF算法中，是对物品i和j的公共用户，没多一个多加一分，对这个1分进行惩罚，除上该公共用户购买物品的总数N(u)，购买物品越多，惩罚力度越大。

2. 对相似度进行归一化

$$\hat{w}_{ij} = \frac{w_{ij}}{\max_j w_{ij}}$$

对于相似矩阵 $w_{ij}$ ，有A、B和C三种物品，可能A是热门产品，A对应的行向量的值都会偏大，通过除以每一行的最大值进行归一化，可以一定程度的消除这种热门和冷门之间的差异，提升推荐系统的多样性和覆盖率。

## 隐语义模型(LFM - latent factor model)

LFM核心思想是通过隐含特征联系用户兴趣和物品。对用户对于物品兴趣进行分类，然后从分类中挑选他可能喜欢的物品。LFM中这个分类不是人工定义的，是通过监督学习计算得来的。

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

公式中的 $p_{u,k}$ 和 $q_{i,k}$ 是模型的参数，其中 $p_{u,k}$ 度量了用户u的兴趣和地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。

## 显性反馈和隐性反馈在LFM建模方式

LFM在显示反馈数据(评分或者喜爱程度分类)上解决评分或者分类预测有着较好的精度，可以直接进行建模。但是在隐性反馈数据集中，只有正样本(用户喜欢什么物品)，没有负样本(用户对什么不感兴趣)，因此需要进行负采样。

### 负采样

1. 对于每个用户，都要保证正负样本的平衡(数目类似)。
2. 对于每个用户采样负样本时，要选取那些很热门，而用户却没有行为的物品。  
直接对训练集进行简单随机采样即可，某一物品出现的次数越多，说明它的热门程度越高，被抽中的概率也会变高。

### 模型的损失函数

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

对于隐性反馈数据，经过负采样之后，可以得到一个用户--物品集 $K=\{(u,i)\}$ ，其中如果(u,i)是正样本，则 $r_{ui} = 1$ ，否则 $r_{ui} = 0$ 。 $\lambda \|p_u\|^2 + \lambda \|q_i\|^2$ 是用来防止过拟合的正则项， $\lambda$ 是超参数。可以通过梯度下降的方法进行损失的优化：

$$\frac{\partial C}{\partial q_{uk}} = -2q_{ik} \cdot e_{ui} + 2\lambda p_{uk}$$

$$\frac{\partial C}{\partial q_{ik}} = -2p_{uk} \cdot e_{ui} + 2\lambda q_{ik}$$

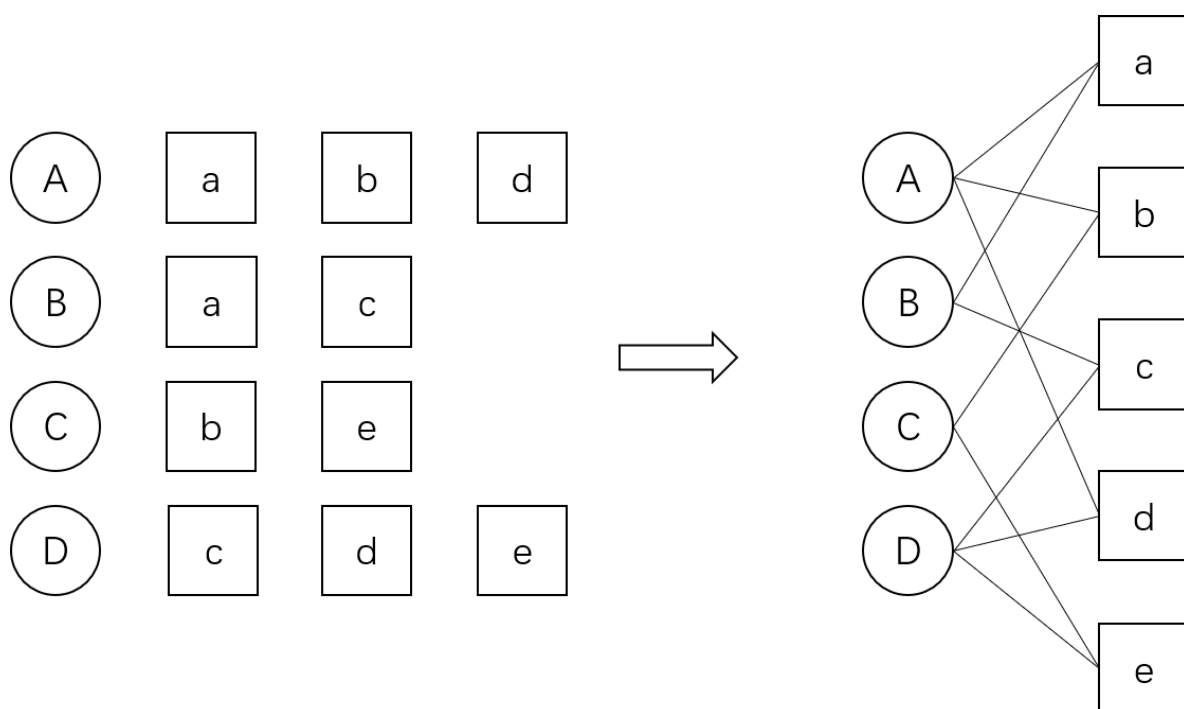
$$p_{uk} = p_{uk} + \alpha(q_{ik} \cdot e_{ui} - \lambda p_{uk})$$

$$q_{ik} = q_{ik} + \alpha(p_{uk} \cdot e_{ui} - \lambda q_{ik})$$

LFM模型中的主要参数:

- 隐特征的分數K
- 学习率 $\alpha$
- 正则参数 $\lambda$
- 负样本\正样本的比例ratio

## 基于图的模型



其中圆形节点代表用户，方形节点代表物品，边代表用户对物品的行为。

## 基于随机游走的PersonalRank算法

### 迭代算法

上面的二分图表示用户A对物品a d都感兴趣，最简单的情况PersonalRank假设每条边感兴趣的程度是相同的，现在要对用户A进行推荐，实际就是计算A对所有物品感兴趣的程度。在PersonalRank算法中不区分用户节点和物品节点，问题就转化成：对节点A来说，节点A B C a b c d的重要度各是多少。重要度用PR来表示。

初始赋予 $PR(A) = 1$ ,  $PR(i \neq A) = 0$ 即对于A来说，他自身的重要度为满分，其他节点的重要度均为0。

然后开始在图上游走。每次都是从PR不为0的节点开始游走，往前走一步。继续游走的概率是 $\alpha$ ，停留在当前节点的概率是 $1 - \alpha$ 。

第一次游走，从A节点以各自50%的概率走到了a和d，这样a和d就分得了A的部分重要度， $PR(a) = PR(d) = \alpha * PR(A) * 0.5$ 。最后PR(A)变为 $1 - \alpha$ 。第一次游走结束后PR不为0的节点有A a d。

第二次游走，分别从节点A a d开始，有a d A B D五个节点获得了权重得分

$$PR(a) = \frac{1}{2}PR(A) + \frac{1}{2}PR(B)$$

$$PR(d) = \frac{1}{2}PR(A) + \frac{1}{3}PR(D)$$

...

计算根节点PR(A)时：

$$PR(A) = 1 - \alpha + \frac{1}{2}PR(a) + \frac{1}{2}PR(d)$$

迭代公式为：

$$PR(v) = \begin{cases} \alpha \sum_{v' \in in(v)} \frac{PR(v')}{|out(v')|} & (v \neq v_u) \\ (1 - \alpha) + \alpha \sum_{v' \in in(v)} \frac{PR(v')}{|out(v')|} & (v = v_u) \end{cases}$$

## 迭代算法思考

1. 这个迭代公式在迭代过程的每一轮中加和是否能保证为1

系统每一轮的计算中，流动到下一轮的PR总量是上一轮PR总量的 $\alpha$ ，因此只要每轮再加上 $1 - \alpha$ 的值是一定可以保证为1

2.  $1 - \alpha$ 为什么要加在root节点

在第一轮迭代的时候，很明显是在root节点加上 $1 - \alpha$ ，但是在整个迭代状态的中间某一轮结束时，此时节点的PR值加和为1，我们也看不出哪个节点是root节点了，整个系统经过下一轮PR值的流出后，只有 $\alpha$ 的PR剩余，那么剩下的 $1 - \alpha$ 的PR值应该加到哪个节点？

3. 换一种思路，每轮迭代每个节点的PR值的 $\alpha$ 流出到下一个节点，剩余的 $1 - \alpha$ 是由各个节点所保留的PR值的和为 $1 - \alpha$ ，应该算到各个节点上，而不是root节点上。这样同样可以维持每轮迭代之后系统总和为1的条件。

所以，在root节点上 $1 - \alpha$ 只是定义了一种维持系统PR总和稳定的方式，并不是一定要加在root节点上。

经过实验，将 $1 - \alpha$ 留在各个节点上的方式和都留在root节点上的方式都能够收敛，但是结果是有所差别的。 $1 - \alpha$ 留在root节点上的方式，会使需要推荐的User节点得分高很多。

4. 是否一定能收敛

$(1 - \alpha)PR$ 值保留在root节点的方式从实验结果来看可以收敛，但是不清楚收敛条件。保留在各个节点中的方式，可以看作马尔可夫过程，每个节点有一个初始状态，有一个转移状态矩阵，在平稳马尔可夫过程的条件下， $n$ 个转移矩阵相乘， $n$ 趋近无穷时，会得到一个常量矩阵，因此可以收敛。

## 矩阵运算平稳过程

PersonalRank经过多次的迭代游走，使得各节点的重要度趋于稳定，实际上我们根据状态转移矩阵，经过一次矩阵运算就可以直接得到系统的稳态。

$$r = (1 - \alpha)(1 - \alpha M^T)^{-1} r_0$$

## 四种推荐算法的比较

### UserCF和ItemCF的比较

1. UserCF推荐结果着重于反映和用户兴趣相似的小群体的热点，ItemCF推荐结果着重于用户自己的历史兴趣。
2. ItemCF需要维护Item相似度的表，UserCF需要维护User相似度的表，如果Item的维度大或者Item需要经常的变动，应该选择UserCF，反过来应该选择ItemCF。

## LFM和协同过滤方法的比较

1. LFM是一种有监督学习方法，协同过滤属于基于统计的传统方法，LFM的拟合能力会更强。
2. 协同过滤方法需要维护一张比较大的相似度表，空间复杂度远高于LFM。
3. LFM无法实时在线更新参数，用户有新的行为，LFM只能通过重新训练的方法来更新模型。
4. ItemCF有着较好的推荐解释，是基于用户的历史行为的推荐结果。LFM的隐类很难展现给用户。

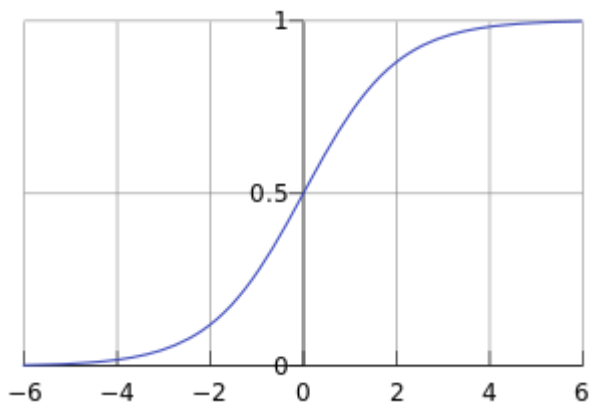
## 基于图的模型

通过随机游走的方式有着良好的可解释性，但时间复杂度上有着明显的缺点，收敛的速度很慢，不仅无法在线提供实时推荐，甚至离线生成推荐结果也很耗时。

## 逻辑回归在推荐的应用

逻辑函数：

$$g(z) = \frac{1}{1 + e^{-z}}$$



逻辑回归模型假设：

$$P(y = 1|x; \theta) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

在推荐的模型中， $\theta$ 是样本的特征， $x$ 是样本在特征下的取值，实质上是将一个线性分类器做了一个非线性的映射，使得它的拟合能力更强了。简单的推荐场景下，假定1为用户的购买行为，0为用户未购买，对模型进行训练，不断更新 $\theta$ 值，使得发生过购买行为的样本模型输出值接近1，未发生购买行为的样本模型输出值接近0，模型性能稳定之后，模型训练完成。预测的时候将预测样本的 $x$ 值代入模型中，当结果大于某个人为设定的阈值时，我们就预测他会有购买行为，反之预测不会购买。