第十二章、推荐系统简介

赵涵

2023 年 5 月 11 日

这一章我们介绍推荐系统（recommend system）。推荐系统的应用场景非常多，比如，当你看了一部电影并认为该电影是一部高分电影，那么电影网站会给你推荐与该电影类似的电影；当你浏览购物网站时，搜索了一个词条后，该网站会不停地向你推送相关的产品；当你浏览视频网站时，给你推送的广告要尽可能地符合你当前的需求（视频会员广告推送免广告，汽车广告可以迎合有买车需求的用户）；广告投放如何获得最大化利润；视频引流与推广，为了获得最大化观看时长等等。虽然推荐系统并不具备很高的学术价值，但是具备非常大的商业价值。所以对推荐系统进行一些简单地介绍。推荐系统有三类：

- 基于内容的推荐系统。
- 协同过滤推荐系统。
- 混合的推荐系统。

接下来我们一一进行介绍。

1 推荐系统的提出

通过一个例子，说清楚推荐系统面对的问题。目前有四位用户，他们分别为Alice, Bob, Carol, Dave。对于五部电影，他们都有着各自的评价，我们采用最常见的评价方式，分为0星到5星，0 星代表该用户非常不喜欢，5星代表该用户极度推荐。那么我们给出如下的打分表格：

movie	Alice	Bob	Carol	Dave
Titanic	5	5	0	0
Ghost	5	?	?	0
Roman Holiday	?	4	0	?
Rush Hour	0	0	5	4
The Fast and the Furious	0	0	5	?

这里的？表示当前列用户没有观看过当前行对应的电影。为了接下来描述问题清晰，我们引入一些记号：

1. n_u表示用户的数量，这里$n_u = 4$。
2. n_m表示电影的数量，这里$n_m = 5$。
3. $r(i, j) = 1$，如果用户j观看过电影i，那么该值为1。
4. $y^{(i,j)}$用来表示用户j对电影i的评分（必须在$r(i, j) = 1$为1的前提下才有意义）。

2 特征的使用

对于一部电影来说，当然也有特征，这里我们简单引入两个特征，分别为爱情和动作，分别用x_1和x_2表示。那么对于上表，我们可以进行一些扩充：

movie	Alice	Bob	Carol	Dave	x_1	x_2
Titanic	5	5	0	0	0.9	0
Ghost	5	?	?	0	1.0	0.01
Roman Holiday	?	4	0	?	0.99	0
Rush Hour	0	0	5	4	0.1	1.0
The Fast and the Furious	0	0	5	?	0	0.9

特征下对应的数值越大，表示属于该类型的可能性越大。对应一个用户对一部电影的评分，我们可以用最简单的内积形式表达：$w^{(j)^T}x^{(i)}+b^{(j)}$，这里w表示用户的参数，通过内积，得出对一部电影的评分。比如，我们给定用户1的参数向量为$w^{(1)}=(5\ 0)^T$，$b^{(1)}=0$，电影3的特征向量为$x^{(3)}=(0.99\ 0)^T$，那么内积的结果为4.95，是符合我们上表给出的5分的评价的。那么对于用户没有看过的电影，就可以通过这种方式进行预测该用户的评分，然后推荐给他预测分数最高的前几部电影。

3 协同过滤算法

首先我们讨论对用户1参数的学习，即w和b，我们采用最简单的均方误差作为损失函数，有如下的表达式：

$$J(w^{(1)},b^{(1)})=\frac{1}{2m^{(1)}}\sum_{i:r(i,1)=1}(w^{(1)^T}x^{(i)}+b^{(1)}-y^{(i,1)})^2+\frac{\lambda}{2m^{(1)}}\sum_{k=1}^{n}(w_k^{(1)})^2 \tag{1}$$

那么对于所有的用户，我们可以把所有的用户损失函数求和，就构成总体的目标函数，即：

$$J(w^{(1)},b^{(1)},...,w^{(n_u)},b^{(n_u)})=\frac{1}{2m^{(j)}}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}(w^{(j)^T}x^{(i)}+b^{(j)}-y^{(i,j)})^2+\frac{\lambda}{2m^{(j)}}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(w_k^{(j)})^2 \tag{2}$$

其实对于电影的特征，我们也可以进行学习，比如说，上映了一部新电影，这部电影并没有具体的分类，但是去观看的观众，对该电影进行了得分，那么我们就可以默认每个用户的参数向量已经是最优的，通过该参数，去推断带电影的特征向量，即给定w和b，我们去学习x，对于一部电影(假定标号为1)来说，有如下的损失函数：

$$J(x^{(1)})=\frac{1}{2}\sum_{j:r(1,j)=1}(w^{(j)^T}x^{(1)}+b^{(j)}-y^{(1,j)})^2+\frac{\lambda}{2}\sum_{k=1}^{n}(x_k^{(1)})^2 \tag{3}$$

那么对于所有已经打分后的新电影，就可以通过求和，把损失函数表示出来：

$$J(x^{(1)},x^{(2)},...,x^{(n_m)})=\frac{1}{2}\sum_{i=1}^{n_m}\sum_{j:r(i,j)=1}(w^{(j)^T}x^{(i)}+b^{(j)}-y^{(i,j)})^2+\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}(x_k^{(i)})^2 \tag{4}$$

更新方式可以采用梯度下降，先更新w和b，在更新x。协同过滤算法也因此得名，不是一个用户对单一的电影的评价，而是众多用户对一部电影的评价，然后基于此，进行更新。而对一个用户的参数更新，也是由于多个电影的评分去估计，所以叫协同过滤。

4 二进制标签

这里我们介绍了对电影的推荐，如果是对商品的推荐呢？比如用户在浏览商品后，是否会购买商品？是否对该商品感兴趣？用户会在你的商品界面停留30秒以上吗？用户会点进去商品详情页进行仔细浏览吗？对于这一类问题，其实是二进制问题，我们给出三种分类：

1. 1-表示当用户看到该商品后，有意向继续进行了解。
2. 0-表示当前用户没有意愿进行了解。
3. ?-表示当前用户还没查看过该商品。

只要把线性回归的损失函数，改成交叉熵的损失函数，就解决了当前问题，即对于二值标签$y^{(i,j)}$表示向用户j 推荐商品i的概率值，由sigmoid函数得出，$g(w^{(j)^T}x^{(i)}+b^{(j)})$，这里$g(z)$ 表示sigmoid函数。那么总的损失函数具有如下的表达式：

$$L(f_{(w,b,x)}(x),y^{(i,j)})=-y^{(i,j)}\log(f_{(w,b,x)}(x))-(1-y^{(i,j)})\log(1-f_{w,b,x}(x)) \tag{5}$$

同样的，把该用户观看过的所有的视频或者广告进行求和，有：

$$J(w,b,x)=\sum_{(i,j):r(i,j)=1}L(f_{(w,b,x)}(x),y^{(i,j)}) \tag{6}$$

5 均值归一化

之所以要均值归一化，是因为一般用户的参数，初始化为0向量。当一个新电影上映时，对于一个新用户来说，他对所有的电影预测都是0分，这明显是不符合预期的。正如我们在评分网站看到的一样，我们看到的都是对电影的平均分，所以我们在学习参数的时候，一般需要对评分进行均值归一化处理，处理方式非常简单，就是对矩阵的每一行，求平均值，得到一个均值向量μ，然后在训练时，对评分矩阵都减去均值向量。训练结束，然后给出如下的预测：

$$w^{(j)}x^{(i)}+b^{(i)}+\mu_i \tag{7}$$

6 寻找相关特征

对于一个学习后的特征，其实是很难解释的，所谓寻找相关特征，就是如何找到相关的作品来进行推荐，最常用的方法是计算两个作品之间的欧几里得距离，即：

$$\sum_{i=1}^{n}(x_i^{(k)}-x_i^{(i)})^2 \tag{8}$$

但是这种计算方式存在一些问题，比如认为每一维度的权重是相同的。对于协同过滤算法，也有一些局限性：

- 冷启动：对于很少有用户评论的作品，怎么进行排序？对于很少给出评价的用户，如何向他们推荐有意义的作品？
- 没有使用到作品和用户的详细信息：作品的风格，电影明星，影业公司，用户的性别，年龄，位置等等。

对于以上的问题，我们接下来介绍基于内容的推荐。

7 基于内容进行推荐

基于内容的推荐（Content-based filtering），是完全依靠内容进行推荐。给定两个向量，一个是作品的向量，一个是用户的向量，比如作品的向量为$x_m^{(i)}=(x_1,x_2,...,x_u)^T$，这里的每一个分量代表作品的特征，比如年份，流派，平均分等等，同样对于用户也有一个向量$x_m^{(j)}$，里面的分量可以代表年龄，性别，国籍等等。这里需要强调用户与作品的特征向量维度不一定相同。那么怎么找到一种方法让两种维度不同的向量匹配在一起，此时深度学习正好弥补这个空缺。