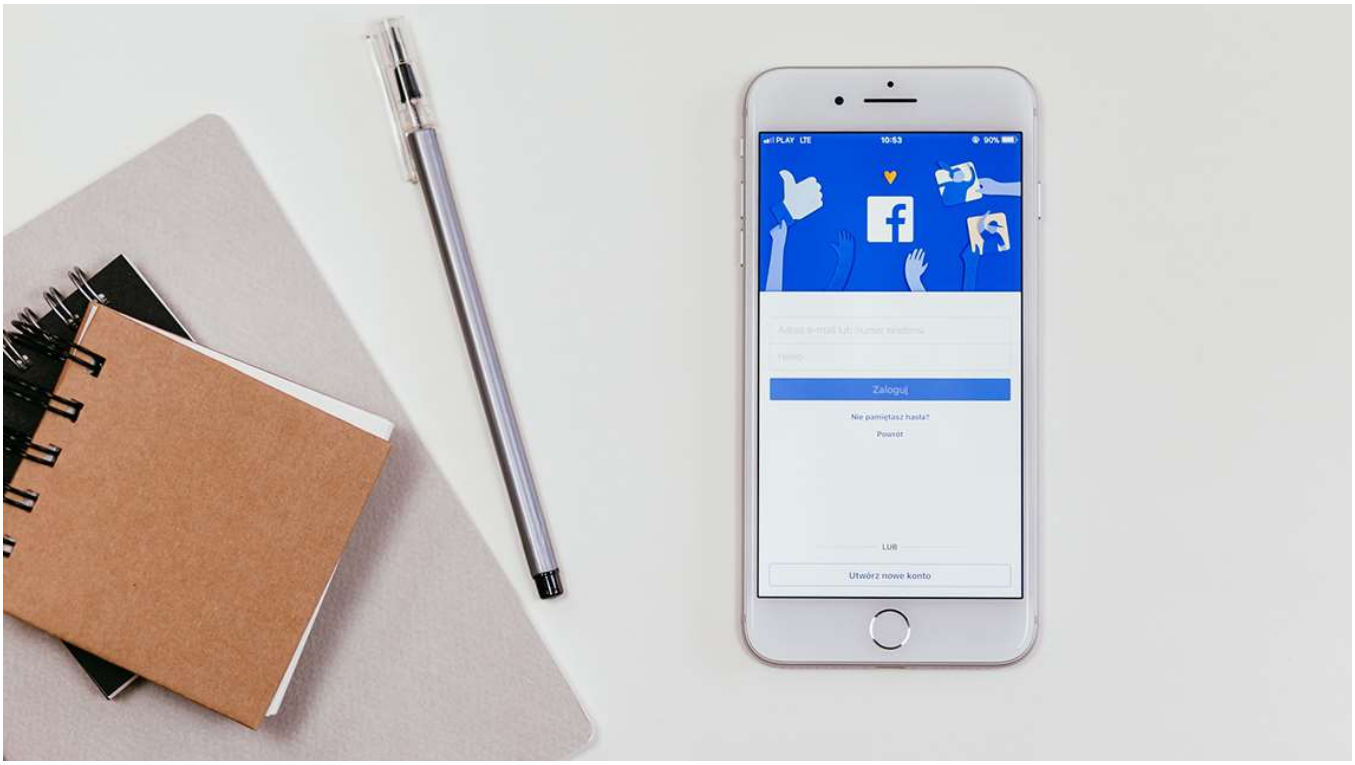
【矩阵分解】Facebook是怎么为十亿人互相推荐好友的

2018-03-28 刑无刀



【矩阵分解】Facebook是怎么为十亿人互相推荐好友的
朗读人：黄洲君　11′49″ | 5.42M

上一篇中，我和你专门聊到了矩阵分解，在这篇文章的开始，我再为你回顾一下矩阵分解。

回顾矩阵分解

矩阵分解要将用户物品评分矩阵分解成两个小矩阵，一个矩阵是代表用户偏好的用户隐因子向量组成，另一个矩阵是代表物品语义主题的隐因子向量组成。

这两个小矩阵相乘后得到的矩阵，维度和原来的用户物品评分矩阵一模一样。比如原来矩阵维度是 m x n，其中 m 是用户数量，n 是物品数量，再假如分解后的隐因子向量是 k 个，那么用户隐因子向量组成的矩阵就是 m x k，物品隐因子向量组成的矩阵就是 n x k。

得到的这两个矩阵有这么几个特点：

1. 每个用户对应一个 k 维向量，每个物品也对应一个 k 维向量，就是所谓的隐因子向量，因为是无中生有变出来的，所以叫做“隐因子”；
2. 两个矩阵相乘后，就得到了任何一个用户对任何一个物品的预测评分，具体这个评分靠不靠谱，那就是看功夫了。

所以矩阵分解，所做的事就是矩阵填充。那到底怎么填充呢，换句话也就是说两个小矩阵怎么得到呢？

按照机器学习的套路，就是使用优化算法求解下面这个损失函数：

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - p_u q_i^T)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

这个公式依然由两部分构成：加号左边是误差平方和，加号右边是分解后参数的平方。

这种模式可以套在几乎所有的机器学习训练中：就是一个负责衡量模型准不准，另一个负责衡量模型稳不稳定。行话是这样说的：一个衡量模型的偏差，一个衡量模型的方差。偏差大的模型欠拟合，方差大的模型过拟合。

有了这个目标函数后，就要用到优化算法找到能使它最小的参数。优化方法常用的选择有两个，一个是随机梯度下降（SGD），另一个是交替最小二乘（ALS）。

在实际应用中，交替最小二乘更常用一些，这也是社交巨头 Facebook 在他们的推荐系统中选择的主要矩阵分解方法，今天，我就专门聊一聊交替最小二乘求矩阵分解。

交替最小二乘原理 (ALS)

交替最小二乘的核心是交替，什么意思呢？你的任务是找到两个矩阵 P 和 Q，让它们相乘后约等于原矩阵 R：

$$R_{m\times n} = P_{m\times k} \times Q^T_{n\times k}$$

难就难在，P 和 Q 两个都是未知的，如果知道其中一个的话，就可以按照线性代数标准解法求得，比如如果知道了 Q，那么 P 就可以这样算：

$$P_{m\times k} = R_{m\times n} \times Q^{-1}_{n\times k}$$

也就是 R 矩阵乘以 Q 矩阵的逆矩阵就得到了结果。

反之知道了 P 再求 Q 也一样。交替最小二乘通过迭代的方式解决了这个鸡生蛋蛋生鸡的难题：

1. 初始化随机矩阵 Q 里面的元素值；
2. 把 Q 矩阵当做已知的，直接用线性代数的方法求得矩阵 P；
3. 得到了矩阵 P 后，把 P 当做已知的，故技重施，回去求解矩阵 Q；
4. 上面两个过程交替进行，一直到误差可以接受为止。

你看吧，机器就是这么单纯善良，先用一个假的结果让算法先运转起来，然后不断迭代最终得到想要的结果。这和做互联网 C2C 平台的思路也一样，告诉买家说：快来这里，我们是万能的，什么都能买到！

买家来了后又去告诉卖家们说：快来这里开店，我这里掌握了最多的剁手党。嗯，雪球就这样滚出来了。

交替最小二乘有这么几个好处：

1. 在交替的其中一步，也就是假设已知其中一个矩阵求解另一个时，要优化的参数是很容易并行化的；
2. 在不那么稀疏的数据集合上，交替最小二乘通常比随机梯度下降要更快地得到结果，事实上这一点就是我马上要说的，也就是关于隐式反馈的内容。

隐式反馈

矩阵分解算法，是为解决评分预测问题而生的，比如说，预测用户会给商品打几颗星，然后把用户可能打高星的商品推荐给用户，然而事实上却是，用户首先必须先去浏览商品，然后是购买，最后才可能打分。

相比“预测用户会打多少分”，“预测用户会不会去浏览”更加有意义，而且，用户浏览数据远远多于打分评价数据。也就是说，实际上推荐系统关注的是预测行为，行为也就是一再强调的隐式反馈。

那如何从解决评分预测问题转向解决预测行为上来呢？这就是另一类问题了，行话叫做 One-Class。

这是什么意思呢？如果把预测用户行为看成一个二分类问题，猜用户会不会做某件事，但实际上收集到的数据只有明确的一类：用户干了某件事，而用户明确“不干”某件事的数据却没有明确表达。所以这就是 One-Class 的由来，One-Class 数据也是隐式反馈的通常特点。

对隐式反馈的矩阵分解，需要将交替最小二乘做一些改进，改进后的算法叫做加权交替最小二乘：Weighted-ALS。

这个加权要从哪说起？用户对物品的隐式反馈，通常是可以多次的，你有心心念念的衣服或者电子产品，但是刚刚剁完手的你正在吃土买不起，只能每天去看一眼。

这样一来，后台就记录了你查看过这件商品多少次，查看次数越多，就代表你越喜欢这个。也就是说，行为的次数是对行为的置信度反应，也就是所谓的加权。

加权交替最小二乘这样对待隐式反馈：

1. 如果用户对物品无隐式反馈则认为评分是 0；
2. 如果用户对物品有至少一次隐式反馈则认为评分是 1，次数作为该评分的置信度。

那现在的目标函数在原来的基础上变成这样：

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} c_{ui}(r_{ui} - p_u q_i^T)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

多出来的 Cui 就是置信度，在计算误差时考虑反馈次数，次数越多，就越可信。置信度一般也不是直接等于反馈次数，根据一些经验，置信度 Cui 这样计算：

$$c_{ui} = 1 + \alpha C$$

其中阿尔法是一个超参数，需要调教，默认值取 40 可以得到差不多的效果，C 就是次数了。

这里又引出另一个问题，那些没有反馈的缺失值，就是在我们的设定下，取值为 0 的评分就非常多，有两个原因导致在实际使用时要注意这个问题：

1. 本身隐式反馈就只有正类别是确定的，负类别是我们假设的，你要知道，One-Class 并不是随便起的名字；
2. 这会导致正负类别样本非常不平衡，严重倾斜到 0 评分这边。

因此，不能一股脑儿使用所有的缺失值作为负类别，矩阵分解的初心就是要填充这些值，如果都假设他们为 0 了，那就忘记初心了。应对这个问题的做法就是负样本采样：挑一部分缺失值作为负类别样本即可。

怎么挑？有两个方法：

1. 随机均匀采样和正类别一样多；
2. 按照物品的热门程度采样。

请允许我直接说结论，第一种不是很靠谱，第二种在实践中经过了检验。

还是回到初心来，你想一想，在理想情况下，什么样的样本最适合做负样本？

就是展示给用户了，他也知道这个物品的存在了，但就是没有对其作出任何反馈。问题就是很多时候不知道到底是用户没有意识到物品的存在呢，还是知道物品的存在而不感兴趣呢？

因此按照物品热门程度采样的思想就是：一个越热门的物品，用户越可能知道它的存在。那这种情况下，用户还没对它有反馈就表明：这很可能就是真正的负样本。

按照热门程度采样来构建负样本，在实际中是一个很常用的技巧，我之前和你提到的文本算法 Word2Vec 学习过程，也用到了类似的负样本采样技巧。

推荐计算

在得到了分解后的矩阵后，相当于每个用户得到了隐因子向量，这是一个稠密向量，用于代表他的兴趣。同时每个物品也得到了一个稠密向量，代表它的语义或主题。而且可以认为这两者是一一对应的，用户的兴趣就是表现在物品的语义维度上的。

看上去，让用户和物品的隐因子向量两两相乘，计算点积就可以得到所有的推荐结果了。但是实际上复杂度还是很高，尤其对于用户数量和物品数量都巨大的应用，如 Facebook，就更不现实。于是 Facebook 提出了两个办法得到真正的推荐结果。

第一种，利用一些专门设计的数据结构存储所有物品的隐因子向量，从而实现通过一个用户向量可以返回最相似的 K 个物品。

Facebook 给出了自己的开源实现 Faiss，类似的开源实现还有 Annoy，KGraph，NMSLIB。

其中 Facebook 开源的 Faiss 和 NMSLIB（Non-Metric Space Library）都用到了 ball tree 来存储物品向量。

如果需要动态增加新的物品向量到索引中，推荐使用 Faiss，如果不是，推荐使用 NMSLIB 或者 KGraph。用户向量则可以存在内存数据中，这样可以在用户访问时，实时产生推荐结果。

第二种，就是拿着物品的隐因子向量先做聚类，海量的物品会减少为少量的聚类。然后再逐一计算用户和每个聚类中心的推荐分数，给用户推荐物品就变成了给用户推荐物品聚类。

得到给用户推荐的聚类后，再从每个聚类中挑选少许几个物品作为最终推荐结果。这样做的好处除了大大减小推荐计算量之外，还可以控制推荐结果的多样性，因为可以控制在每个类别中选择的物品数量。

总结

在真正的推荐系统的实际应用中，评分预测实际上场景很少，而且数据也很少。因此，相比预测评分，预测“用户会对物品干出什么事”，会更加有效。

然而这就需要对矩阵分解做一些改进，加权交替最小二乘就是改进后的矩阵分解算法，被 Facebook 采用在了他们的推荐系统中，这篇文章里，我也详细地解释了这一矩阵分解算法在落地时的步骤和注意事项。

其中，我和你提到了针对 One-Class 这种数据集合，一种常用的负样本构建方法是根据物品的热门程度采样，你能想到还有哪些负样本构建方法吗？欢迎留言一起讨论。感谢你的收听，我们下次再见。

精选留言

瑞雪 4

你好，请问如果选取一部分为负样本，其他的缺失值在矩阵分解时是直接使用NaN吗，有点对正负样本分不太清:D

2018-03-28

林彦 3

1. 既然可以根据物品的热门程度选择负样本，是不是类似也可以根据用户的活跃程度选择负样本?
2. 是不是可以借鉴之前基于内容推荐的方法，先找出和当前用户或场景类似内容的用户或场景中的热门物品的交互作为负样本？这里用户或场景可以用各种距离度量的方式选出k个最相邻的。基于内容相似度找和正样本最相邻的物品作为负样本可能也可以。
3. 负样本从概率分布中采样，概率分布的参数让整体的期望值和真实值尽可能接近。这样交互次数多的有更大概率被选中，或者可以看成赋予了更大权重。
4. 引入一个概率参数变量，有交互的概率为p(i, j)，预测交互值为1；无交互的概率为p(i, j)，预测交互值为0。除了计算用户和物品隐变量外，把用户和物品隐变量固定后再估算这个概率参数。

2018-03-28

Drxan 1

大神，如果要对负样本进行采样的话，是不是就无法用矩阵分解了

2018-03-28

曾阿牛 1

在构建点击率预估模型时，仅将正样本附近未点击的样本视为负样本。样本量大时，剔除一段时间内没有转化行为的用户数据（包括正负样本）

2018-03-28

易初 0

用户 和 物品 是一个pair ， 用dssm 深度语意匹配网络是不是更好

2018-05-22

Lz 0

按照facebook的场景，如果是对用户推荐用户，m，n两轴都是用户的话，算法依然适用么？

2018-05-09

yalei 0

通常用户需要一个“入口”才能浏览商品详情，这个入口可大部分情况下是搜索结果和算法推荐。可以设置曝光埋点，再结合点击埋点来找到真正的负样本（有曝光而无点击的样本）

2018-04-10

森林 0

目标函数里置信度C是1+aC，如果我们挑负样本的话，负样本的次数是啥？

2018-04-02

哎哎哎 0

用户向量存在内存中的话，如何存的下呢？还有就是这种方式对用户向量的更新如果不是实时的话，是否线上服务的用户miss会很高啊，这种情况应该怎么处理呢？

2018-03-31

Drxan 0

大神，如果要对负样本进行采样的话，是不是就无法用矩阵分解了

2018-03-28