推荐系统课程实验报告

基于用户、物品的协同过滤算法

霍超凡

1 实验概述

基于用户、物品的协同过滤算法是一种最简单、单纯的算法，它的思想很简单，利用已知的领域数据去插值得到未知的评分。因为它是利用邻域信息来插值计算的，所以这种方法有称为基于领域的协同过滤算法；同时无论是在建立模型过程中，还是预测评分过程中，模型内部都需要能够随机访问相似度矩阵中的任意元素，这个相似度矩阵通常是很大的一个矩阵，非常占内存，因此这种算法又叫做一种 Memory-based 的协同过滤算法。本次实验以这种算法为主线，探讨了诸如相似度矩阵访问和存储结构的一些实现细节，并在 MovieLens 数据集上测试了算法的性能，但是由于时间、精力有限，没有探讨相似度计算公式对结果的影响、算法中热门物品对推荐结果的影响和如何惩罚热门物品、如何将聚类算法融入该方法中以减少计算复杂度、解决稀释问题等问题。

2 算法细节

2.1 算法步骤

基于用户、物品的协同过滤算法分成两大步骤，第一步计算用户与用户（物品与物品）之间的相似度得到相似度矩阵，第二步根据已知评分和相似度加权插值得到未知评分。实验中采用了两种相似度计算公式：cosine 相似度和 pearson 相关系数，给定评分矩阵R，其中矩阵中的每一个元素$\mathrm{R_{ui}}$表示用户 u 对物品 i 的评分，用户u与用户v之间 cosine 相似度如下计算：

$$\mathrm{cosine(u,v)} = \frac{R_u \cdot R_v}{|R_u| \cdot |R_v|}$$

其中，$\mathrm{R_u}$是用户u对所有物品的评分所组成的一个向量。Pearson 相关系数是一种经过归一化的 cosine 相关系数，这个归一化步骤将所有的评分向量的均值归一化到 0，方差归一化到 1，设经过归一化后的操作得到的向量为

$$\hat{R}_u = \frac{R_u - \bar{R}_u}{|R_u - \bar{R}_u|}$$

即将评分分量减去均值后再除以方差。Pearson 相关系数的计算方式为

$$\mathrm{pearson(u,v)} = \frac{\hat{R}_u \cdot \hat{R}_v}{|\hat{R}_u| \cdot |\hat{R}_v|}$$

实验中只使用了这两种计算方式，没有探讨其它计算方式和机器学习算法聚类算法的距离计算公式。

在得到相似度后计算未知评分，这个评分的计算方式如下

$$\mathrm{rating(u,i)} = \bar{\mathrm{R}}_{\mathrm{u}} + \frac{\sum_v sim(u,v) \cdot (R_{vi} - \bar{R}_v)}{\sum_v sim(u,v)}$$

$\mathrm{sim(u,v)}$可以是 cosine 相似度计算方式或 pearson 系数计算公式，$\bar{\mathrm{R}}_{\mathrm{u}}$是用户u的评分均值。

2.2 数据集的存储结构和访问

在算法模型建立之前，我们通常需要加载数据集，数据集是用户对物品的评分，其逻辑结构是一张二维表，这张表通常非常大，而且很稀疏。如何存储和检索这张二维表是我们一个头疼的问题。在实验过程中使用两种方式存储和检索二维表，一种是以矩阵的方式存储二维矩阵的每一个元素，这种存储方式检索速度非常快，但是十分占用空间，在数据集 MovieLens100K 和 MovieLens1M 可以使用这种方式存储，而 MovieLens10M 在我笔记本的内存存储不下。另外一种方式是稀疏矩阵的存储方式，像关系型数据库一样，逐条记录存储，为了便于检索需要在一些字段中建立索引，这种方式可以极大的节约空间，但是检索过程需要遍历索引，效率不如第一种$\mathrm{O(1)}$的访问效率。具体实现中，使用 numpy.ndarray 存储完整矩阵，使用 pandas.DataFrame 存储第二种方式描述的矩阵。在实验过程中 MoiveLens1M 使用第二种方式在建立模型时十分耗时，可能需要数小时才能计算完相似度矩阵，考虑到这点，我只在数据集 MovieLens100K 和 MovieLens 上运行了算法，其存储方式均为第一种方式。

2.3 相似度矩阵的计算与存储

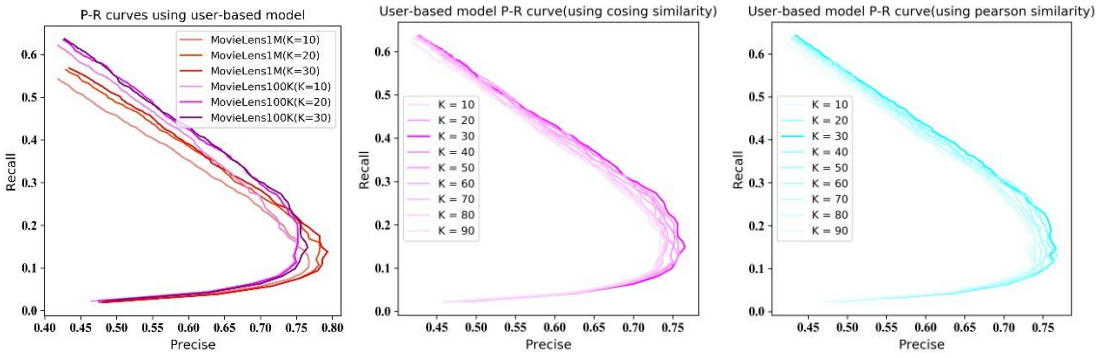
参考大多数算法的实现，只存储相似度较高的 K 中物品（或用户）之间的相似度，以基于用户的协同过滤算法来说，对于一个用户 u 来说，其它用户 v 与他之间的相似度大多数基本接近于 0，他们在最后的插值中影响不大，我们只存储和用户 u 相似度较高的 K 个用户，这种方法叫做 k 近邻或者 topk 方法。存储结构上使用稀疏矩阵的存储方式，建立一个字典，字典中的每一项以用户的 ID 为键，值为一个和该用户相似度较高的 K 的用户的 ID 和相似度所组成的列表。

2.4 算法衡量指标

在评价算法时，将数据集按照一定比例（训练数据集：测试数据集=8：2）划分成训练数据集和测试数据集，训练数据集用于建立模型，测试数据就用于测量模型在测试数据集上的评价指标，一种更为科学的方法是使用 k 折交叉验证，但在实验过程中为了简便，我只划分了一次。对于这种具有具体评分的数据集，一个首选的评测指标是预测分值和实际分值之间 MSE，除了这个指标外，还测量了精度、召回率指标，具体操作方式如下，给定一个用户 ID 后算法会给这个用户一个推荐列表，这个列表是经过排序的列表，越靠前的评分越高。在这个推荐列表中计算精确度和召回率，我们规定实际评分大于等于 3 视为正例，其余均视为负例（实际上，为了使数据结果更好看，可以将那些没有评分的排除出去，但我没有排除）。推荐列表中不考虑用户是否已经购买或评价过，所以训练集中的物品也会出现在推荐列表。

3 实验结果

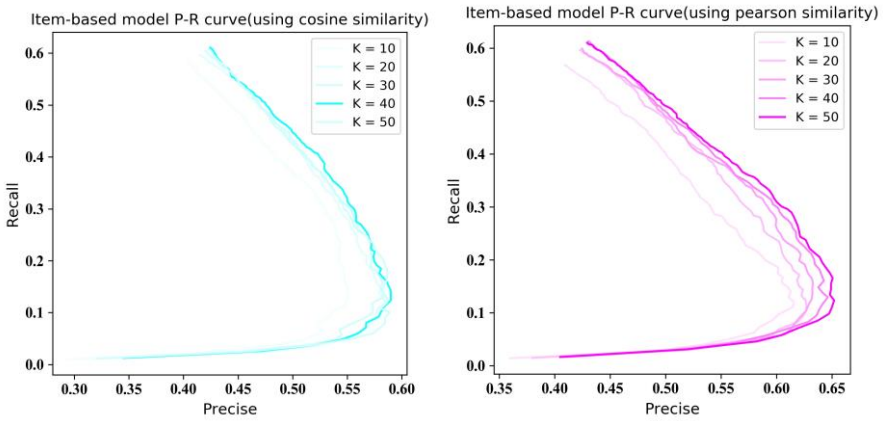
首先，在数据集 MovieLens100K 和 MovieLens1M 中评测算法，如图 1(a)所示，似乎数据集规模大一点效果会好一点，差距不是非常大。敏锐的读者可能已经明显感受到这个 PR 曲线怎么和正常的 PR 曲线不一样，这也是另我困惑的地方，~~我们推荐算法总是把评分较低的放在前面，以至于刚开始的精确度很低，再往后猛增，之后正常，我也不清楚自己算法那里出现了问题，这个（也包括之后）PR 曲线是经过平均处理的，一开始不等于精度 0，这说明有些用户推荐对了，对有些用户推荐了不合适的物品，并且这些物品还排在 top1 的位置。~~



(a) (b) (c)

图 1：图 1(a)是基于用户协同过滤算法在数据集 MovieLens100K 和 MovieLens1M 上的 PR 曲线，图 1(b)(c)是基于用户协同过滤算法使用不同的相似度计算公式在数据集 MovieLens100K 上的 PR 曲线。

在 2.3 节所介绍的 k 近邻相似度中 K 这个超参量对结果存在影响，如图 1(b)(c)所示，K=30 最合适，但也不完全是这样，基于物品的协同过滤的 K 值比较“迷”，如图 2 所示，



(a) (b)

图 2：基于物品的协同过滤推荐算法在 MovieLens100K 上的 PR 曲线，左图（cosine 相似度），右图（pearson 相关系数）

图 3 对比了基于用户和基于物品的协同过滤算法，基于用户的协同过滤算法明显要好于基于物品的协同过滤算法。

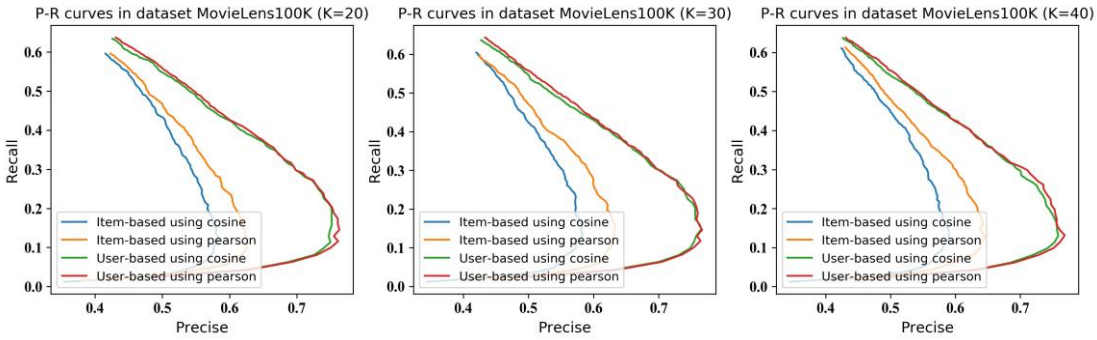


图 3：基于用户和基于物品在数据集 MovieLens 的性能对比

除了 PR 曲线之外，关于 RMSE 的一些结果就不展示了，不知是何原因，RMSE 普遍高在 2.5 左右，我也检查不出模型实现有什么问题。后来，我在对数据访问做出优化时，修改了这两个模型的代码，找出了 RMSE 为何这么高的原因，在重写评价函数时发现了造成以上 PR 曲线出现重大错误的原因。

我们在计算精确度和召回率的过程中，需要遍历推荐列表，检查截至到第 k 项，前 k 个推荐物品中有多少是正例。在遍历时，我使用的 for i range(1, n+1)语句，在写这个语句时考虑到 i 要从 1 开始直到 n，但是在计算 precise 时，把 i 当成计数变量（从 0 开始计数），结果将正样本数目除以 i+1，以致结果不正确，开始 precise 总是从 50%以下起步，往后随着 i 的增加，结果会趋近于正确的结果。这种错误不会影响模型之间的对比，我没有重复实验修改上面的数据。

我们之前已经看到，用于训练的数据集中存在很多评分为 0 的数据，这个评分为 0 的数据对我们结果有很大的影响，以 User-based 模型为例，在计算相似度时，评分为 0 的这个数据是参与运算的，换句话说，把用户对物品的评分真正当成用户对物品 0 评分，实际上，这种情况大概率是用户没有遇到过这样的物品，属于数据的缺失。举例来说均有两个相同偏好的用户，一个用户 A 可能刚刚进入系统，而另一个用户 B 是常驻军，由于两个用户具有相同的偏好，用户 A 喜欢的物品，用户 B 都喜欢，而用户 B 喜欢的物品，用户 A 可能还没有遇到，评分缺失，如果这个评分缺失的值当作 0 参与运算，会使这两个用户的相似度非常低（因为用户 B 喜欢的很多商品，用户 A 的评分为 0），出现矛盾，但是这种计算方式又存在合理性，这种计算方式会削弱热门用户或热门商品的影响力（如果仅仅忽略掉那些评分为 0 的数据，只靠了两者均有评分的情况，那么热门用户会和大多数用户的相似度都很高）。结合相似度计算评分时也会出现这样的抉择，在相似度计算时没有忽略那些评分为零的数据前提下，如果在计算最终评分时，这些为 0 的数据参与运算会使最终的 RMSE 出现偏差，但是精度却很高，相反，如果忽略评分为 0 的数据，会使 RMSE 造成，但是精度非常低。以上两点是我在实验中得到的结论，表 1 中对比了这两种计算评分方式，

表 1：是否忽略评分为 0 的数据对最终结果的影响

	P-10	Freshness-10	RMSE	MAE
忽略评分为 0 的数据	0.101	0.836	1.122	0.891
评分为 0 的数据参与运算	0.857	0.123	2.543	2.235

表 1 中是 User-based 模型在 MovieLens100K 数据集的测评的结果，P-10 表示 top10 中正例的比例，Freshness-10 表示 top10 中那些评分为 0 的比例，P-10 和 Freshness-10 是针对整个数据集计算得到，RMSE 和 MAE 是在测试集上的指标，当我们在计算最终评分时

忽略值为 0 的数据时，模型会推荐那些我们没有遇到过的物品，这些物品或许被和我们相似度较高的用户买过，此时体现了协同过滤的最终目的，当评分为 0 参与运算，推荐的结果都是在数据集中出现的物品，新鲜度低。

参考文献

[1] Deshpande, Mukund & Karypis, George. (2004). Item-based top- N recommendation algorithms. ACM Transactions on Information Systems - TOIS. 22. 143-177. 10.1145/963770.963776.

[2] Sarwar, Badrul & Karypis, George & Konstan, Joseph & Riedl, John. (2001). Item-based Collaborative Filtering Recommendation Algorithms. Proceedings of ACM World Wide Web Conference. 1. 10.1145/371920.372071.

[3] PyCollaborativeFiltering, site: https://github.com/ChangUk/pyCollaborativeFiltering.