计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.191100181

基于张量分解的排序学习在个性化标签推荐中的研究

杨 洋 邸一得 刘俊晖 易 超 周 维
云南大学软件学院 昆明 650500
(yyangyncn@163.com)

摘 要 标签的使用给系统提供了一个划分并管理用户和物品的途径,而个性化的标签推荐则不仅方便用户输入标签,而且有助于提高系统标签的质量。进而,系统可以获得更多关于用户和物品的信息,提升后续推荐的精度,改善用户体验,因此在淘宝、滴滴等类似的业务场景中具有重要的作用。然而,现有的大多数标签推荐都没有关注推荐列表中的排序问题,列表中过于靠后的标签极易丧失让用户使用的机会,造成用户和物品信息的缺失,阻碍后续的精准推荐。针对上述问题,提出了一种基于张量Tucker分解和列表级排序学习的个性化标签推荐算法,采用优化MAP的方式进行训练,并在Last.fm数据集上进行了仿真实验,不仅验证了算法的有效性,而且充分探讨了学习率、核张量维度等参数对算法的影响。实验结果表明,该算法能较好地优化推荐列表的排序问题,且随列表长度的增加,其性能呈线性下降,算法的实现有利于更好地根据用户喜好来推荐服务。

关键词:张量分解;排序学习;标签推荐;Tucker分解
中图法分类号 TP391

Study on Learning to Rank Based on Tensor Decomposition in Personalized Tag Recommendation

YANG Yang,DI Yi-de,LIU Jun-hui,YI Chao and ZHOU Wei
School of Software,Yunnan University,Kunming 650500,China

Abstract The use of tags provides a way for the system to divide and manage users and items,while personalized tag recommendations not only facilitate users input,but also help to improve the quality of system tags. In turn,the system can obtain more information about users and items,improve the accuracy of subsequent recommendations,improve the user experience. Therefore,it plays an important role in similar business scenarios such as Taobao and Didi. However,most existing tag recommendations do not pay attention to the ranking issues in the recommendation list. The tag that is too late in the list is easy to lose the opportunity for user use,resulting in the lack of information about users and items,and hindering the subsequent accurate recommendation. Aiming at the above problems,a personalized tag recommendation algorithm based on tensor Tucker decomposition and list-wise learning to rank is proposed. The algorithm is trained by optimizing MAP,and the simulation experiment is carried out on Last.fm dataset,which not only verified the effectiveness of the algorithm,but also fully explored the influence of learning rate,the dimension of core tensor and other parameters on the algorithm. Experimental results show that the algorithm can optimize the ranking problem of the recommendation list greatly,and its performance decreases linearly with the increase of the length of the list. The implementation of the algorithm is conducive to better recommendation services according to the user preferences.

Keywords Tensor decomposition,Learning to rank,Tag recommendation,Tucker decomposition

1 引言

标签推荐是通过向用户推荐不同的标签来标注物品[1](如电影,歌曲等),这些标签间接地反映了物品特征和用户兴趣,利用这些标签不仅可以实现物品和用户的分类,还能够为具有相同兴趣的用户推荐物品。然而,现在的标签推荐系统还面临着这样一些问题:1)由于兴趣问题,不同的用户可能会对相同的商品标注上不同的标签[2],因此需要给不同的用户推荐个性化的标签推荐列表。该列表中的标签通常是所有标签中按某种规则排序后的top_K个,代表了当前用户在指定物品上有可能使用的标签。2)一部分用户会因为推荐的标签列表排序不优而丧失标注物品的耐心,最终因为没有提供足够数量的标签而导致推荐系统数据稀疏,推荐质量下降。通常来说,用户仅会关注推荐列表中的前3到5个标签,而很少关注之后的标签,假如给用户推荐的列表中有10个标签且都是用户可能使用的,但是因为排序问题,将用户最终使用的标签排在了末尾而导致用户丧失了耐心,从而错失了标注物品和获取用户兴趣的机会,也为系统后续的推荐带来了困难。

个性化标签推荐的数据是关于"用户-物品-标签"的三元组数据,使用张量的形式对其建模可以较好地保留数据的三维空间结构信息。张量是一个多维数组[3],其多维的结构特征正好可以保留数据中的空间结构信息。通过张量分解方法可以更好地挖掘"用户-物品-标签"这3类实体潜在的语义关联,提高推荐质量。因此,本文提出基于张量分解的排序学习方法来实现个性化标签推荐,主要贡献如下:

1)直接针对标签推荐列表进行优化,有效提高排序质量;

本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
通信作者:周维(zwei@ynu.edu.cn)

2)使用张量对数据建模,充分挖掘数据间的潜在语义关联;3)采用梯度下降的更新方式优化目标函数,并在实验上证明了算法的有效性。

本文第2节介绍相关工作;第3节介绍张量分解和排序学习的一些预备知识,为后面的算法做铺垫;第4节介绍本文算法,并给出优化过程;第5节与其他算法进行对比,验证本文算法的有效性;最后总结全文。

2 相关工作

在标签推荐中,项亮[4]提出的基于物品的流行度标签推荐(Pop)是一种简单有效的方法,其基本思路是将当前物品最流行的标签推荐给用户使用,即当用户给物品打标签时,统计当前物品的所有标签及其使用次数,并推荐其中使用次数最高的top_K个给用户。但是,这种方法没有考虑推荐的个性化,给每个人推荐的都是相同的标签。在这之上,Li等[2]使用张量来构建原始的"用户-物品-标签"三维数据,并采用高阶奇异值分解(HOSVD)的方法进行求解,在降低张量维度的同时提高了个性化标签推荐的准确性。为了给动态增长的用户推荐标签,Liao等[5]基于张量分解提出了面向新用户的增量标签推荐方法,在保持张量分解的优势上简化了新用户的推荐计算。

但是,上述工作都没有关注推荐列表的排序问题。Rendle等[6]为解决排序问题提出了一种通过张量的Tucker分解来优化对级排序学习指标AUC的方式,实现标签推荐,并同时提出一种基于标注的数据解释方案。另外,Shi等[7]将列表级排序学习应用到了隐式反馈数据下基于上下文的推荐中,通过张量的CP分解优化推荐列表的MAP指标,进而优化了列表的排序问题,取得了比传统方法更好的效果。

从上述工作中可以看出,张量和推荐系统相结合的方式已经得到了广泛的应用。但是,上述工作都没有尝试使用张量的Tucker分解来优化列表级排序学习的指标。Rendle等[6]已经证明了Tucker分解具有的良好的性质,而Liu[8]和Li[9]的工作都证明了列表级排序学习的效果优于对级排序学习。

为此,本文采用张量的Tucker分解直接对列表的MAP指标进行优化,通过梯度下降的方式学习得到一个关于原始张量数据列表位置的排序模型,极大地提升了个性化标签推荐列表的MAP指标。

3 预备知识

3.1 张量分解

张量分解分为Tucker分解和CANDECOMP/PARAFAC(CP)分解两种,本文使用的是Tucker分解,因此仅介绍张量的Tucker分解。本文中所用的符号表示如表1所列。

表1 本文符号定义表

Table 1 Table of symbols

定义	符号
张量	$A,B,\cdots$
矩阵	$A,B,\cdots$
向量	$a,b,\cdots$
标量	$a,b,\cdots$
张量积	$\circ$
张量 n-模乘积	$\times_n$

Tucker分解将一个张量分解为一个核张量沿每一维乘以对应的因子矩阵的形式。例如:一个三阶张量 $Y\in\mathbb{R}^{M\times N\times T}$ 的Tucker分解可以表示为:

$$y\approx\mathcal{G}\times\mathbf{U}_1\times_2\mathbf{I}\times_3\mathbf{T} \tag{1}$$

其中,$\mathbf{U}\in\mathbb{R}^{M\times P}$,$\mathbf{I}\in\mathbb{R}^{N\times Q}$,$\mathbf{T}\in\mathbb{R}^{T\times R}$ 是对应维度上的因子矩阵;核张量 $\mathcal{G}\in\mathbb{R}^{P\times Q\times R}$ 表示了因子矩阵中对应成分的交互程度。通常来说,核张量的大小 P,Q,R 小于原张量的大小 M,N,T,因此核张量 G 也可以视作是原张量 Y 的压缩表示。

Tucker分解示意图如图1所示。

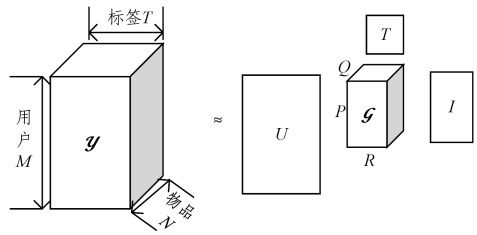

图1 张量Tucker分解示意图

Fig. 1 Decomposition diagram of tensor Tucker

3.2 排序学习

排序学习是一种监督学习方法,利用已有的数据集训练一个相关的排序模型,而从解决排序问题。与传统排序模型相比,其优势在于对众多排序特征进行组合优化,对相应的大量参数自动进行学习,最终得到一个高效精准、更加优化的排序模型。排序学习的通用模型如图2[9]所示。

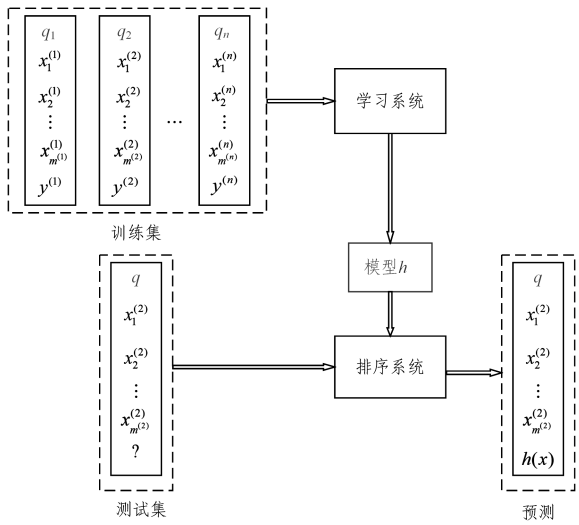

图2 排序学习通用模型图

Fig. 2 General framework of learning to rank

排序学习按照输入样本的不同,可分为3类:点级(Pointwise)、对级(Pairwise)、列表级(Listwise)。本文使用的是列表级的排序学习方法,因此此处仅介绍列表级的排序方法。列表级排序学习相比点级和对级排序学习而言,直接对待排序列表的位置进行优化。目前主要有两种优化方式:1)直接针对排序的评价指标进行优化,例如常用的有MAP(Mean Average Precision),NDCG(Normalized Discounted Cumulative Gain)等。其中,MAP在个性化标签推荐中的定义为:

$$MAP=\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\frac{\sum_{i=1}^{T}\frac{y_{mni}}{p_{mni}}\sum_{j=1}^{T}\mathbb{I}(p_{mnj}\leqslant p_{mni})y_{mnj}}{\sum_{i=1}^{T}y_{mni}} \tag{2}$$

其中，M 表示用户数，N 表示物品数，T 表示标签数；y_{mni} 表示在张量 Y 中用户 m 在物品 n 上是否使用过标签 i，使用过为 1，未使用过为 0；p_{mni} 为用户 m 和物品 n 得到的列表上标签 i 的位置；$\mathbb{I}$ 为指示函数，满足括号内的条件则值为 1，不满足则为 0。2）构造损失函数优化，例如：RankCosine[10] 使用正确排序与预测排序的分值向量的余弦夹角来表示损失函数，ListNet[11] 使用正确排序和预测排序的排列概率分布间的 KL 距离作为损失函数[12]。

4　张量分解在个性化标签推荐中的应用

本节将详细介绍基于张量分解的排序学习在个性化标签推荐中的应用方法，主要包括优化函数的形式化表示、优化函数的光滑化和优化函数求解的过程。

首先明确由核张量和各因子矩阵构建的预测张量 X 中单个值的计算方式：

$$x_{mnt} \approx \mathcal{G} \times_1 \boldsymbol{U}_m \times_2 \boldsymbol{I}_n \times_3 \boldsymbol{T}_t \tag{3}$$

其中，$\boldsymbol{U}_m$，$\boldsymbol{I}_n$，$\boldsymbol{T}_t$ 分别为对应的因子矩阵中的第 m，n，t 行。此外，预测张量 X 是针对列表中元素的排序位置学习得到的模型，其值的大小代表了用户在指定物品上使用该标签的可能性，由此，值比较大的元素的位置也应该比较靠前，也即我们可以将值的大小视为该元素的相对排序位置。

接下来，我们使用预测的值对公式中的 $1/p_{mnt}$ 和指示函数 $\mathbb{I}$ 进行光滑处理。

其中，根据 Chapelle 等的工作[13]，我们使用 Sigmoid 函数 $s(x)=\frac{1}{1+e^{-x}}$ 近似式(2)中的指示函数 $\mathbb{I}$ 来使其光滑，这也是优化中常用的方法之一，其基本思想是：如果元素 j 的排序位置相较于元素 i 是靠前的，那么它在模型中的值也应该是大于元素的。

$$I(p_{mnj} \leqslant p_{mni}) \approx s(x_{mnj} - x_{mni}) = s(x_{mn(j-i)})$$

根据 Shi 等的工作[7]，我们再次使用 Sigmoid 函数来近似 $1/p_{mnt}$，其基本思想是：当 x_{mnt} 的值较大时，$s(x_{mnt})$ 接近于 1，$1/p_{mnt}$ 也接近于 1，而 p_{mnt} 也就越小；相反，当 x_{mnt} 的值较小时，p_{mnt} 也就越大，位置也就越靠后。其公式如下：

$$\frac{1}{p_{mnt}} \approx s(x_{mnt})$$

综上，我们可以得到 MAP 的光滑近似，即优化函数光滑处理后的形式化表示：

$$MAP = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{\sum_{i=1}^{T} y_{mni} s(x_{mni}) \sum_{j=1}^{T} y_{mnj} s(x_{mn(j-i)})}{\sum_{i=1}^{T} y_{mni}} \tag{4}$$

得到了光滑的优化函数后，我们可以使用优化方法对其进行优化，如梯度下降。为了防止过拟合，在优化函数上加上正则化项，可得：

$$L(\mathcal{G}, \boldsymbol{U}, \boldsymbol{I}, \boldsymbol{T}) = \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{\sum_{i=1}^{T} y_{mni} s(x_{mni}) \sum_{j=1}^{T} y_{mnj} s(x_{mn(j-i)})}{\sum_{i=1}^{T} y_{mni}} - \frac{\lambda}{2}(\| \mathcal{G} \|^2 + \| \boldsymbol{U} \|^2 + \| \boldsymbol{I} \|^2 + \| \boldsymbol{T} \|^2) \tag{5}$$

请注意，我们忽略了常数项 $\frac{1}{MN}$，因为该项对优化没有影响。

分别对参数 G，$\boldsymbol{U}_m$，$\boldsymbol{I}_n$，$\boldsymbol{T}_i$ 求偏导可得：

$$\frac{\partial L}{\partial \mathcal{G}} = \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{1}{\sum_{i=1}^{T} y_{mni}} \sum_{i=1}^{T} y_{mni} [s'(x_{mni}) \sum_{j=1}^{T} y_{mnj} s(x_{mn(j-i)}) (\boldsymbol{U}_m \circ \boldsymbol{I}_n \circ \boldsymbol{T}_i) + s(x_{mni}) \sum_{j=1}^{T} y_{mnj} s'(x_{mn(j-i)}) (\boldsymbol{U}_m \circ \boldsymbol{I}_n \circ \boldsymbol{T}_j - \boldsymbol{U}_m \circ \boldsymbol{I}_n \circ \boldsymbol{T}_i)] - \lambda \mathcal{G} \tag{6}$$

$$\frac{\partial L}{\partial \boldsymbol{U}_m} = \sum_{n=1}^{N} \frac{1}{\sum_{i=1}^{T} y_{mni}} \sum_{i=1}^{T} y_{mni} [s'(x_{mni}) \sum_{j=1}^{T} y_{mnj} s(x_{mn(j-i)}) (\mathcal{G} \times_2 \boldsymbol{I}_n \times_3 \boldsymbol{T}_i) + s(x_{mni}) \sum_{j=1}^{T} y_{mnj} s'(x_{mn(j-i)}) (\mathcal{G} \times_2 \boldsymbol{I}_n \times \boldsymbol{T}_j - \mathcal{G} \times_2 \boldsymbol{I}_n \times_3 \boldsymbol{T}_i)] - \lambda \boldsymbol{U}_m \tag{7}$$

$$\frac{\partial L}{\partial \boldsymbol{I}_n} = \sum_{m=1}^{M} \frac{1}{\sum_{i=1}^{T} y_{mni}} \sum_{i=1}^{T} y_{mni} [s'(x_{mni}) \sum_{j=1}^{T} y_{mnj} s(x_{mn(j-i)}) (\mathcal{G} \times_1 \boldsymbol{U}_m \times_3 \boldsymbol{T}_i) + s(x_{mni}) \sum_{j=1}^{T} y_{mnj} s'(x_{mn(j-i)}) (\mathcal{G} \times_1 \boldsymbol{U}_m \times_3 \boldsymbol{T}_j - \mathcal{G} \times_1 \boldsymbol{U}_m \times_3 \boldsymbol{T}_j)] - \lambda \boldsymbol{I}_n \tag{8}$$

$$\frac{\partial L}{\partial \boldsymbol{T}_i} = \sum_{m=1}^{M} \sum_{n=1}^{N} \frac{y_{mni} (\mathcal{G} \times_1 \boldsymbol{U}_m \times_2 \boldsymbol{I}_n)}{\sum_{i=1}^{T} y_{mni}} [s'(x_{mni}) \sum_{j=1}^{T} y_{mnj} s(x_{mn(j-i)}) + (s(x_{mnj}) - s(x_{mni})) \sum_{j=1}^{T} y_{mnj} s'(x_{mn(j-i)})] - \lambda \boldsymbol{V}_i \tag{9}$$

其中，$s'(x_{mni}) = s(x_{mni})(1 - s(x_{mni}))$ 是 Sigmoid 函数的导数。

Tucker 张量分解优化 MAP 如算法 1 所示。

算法 1　Tucker 张量分解优化 MAP(TD-MAP)

输入：训练集 Y，正则化参数 λ，学习率 η，最大迭代次数 itermax

输出：核张量 $\mathcal{G}$，因子矩阵 $\mathbf{U}$，$\mathbf{I}$，$\mathbf{T}$

初始化：随机初始化核张量 $\mathcal{G}^{(0)}$ 和因子矩阵 $\mathbf{U}^{(0)}$，$\mathbf{I}^{(0)}$，$\mathbf{T}^{(0)}$，当前迭代轮数 k=0，由式(2)计算的初始 MAP 值 $p^{(0)}$

repeat

　for m=1,2,…,M do

　　$U_m^{(k+1)} = U_m^{(k)} + \eta \frac{\partial L}{\partial U_m^{(k)}}$ 由式(7)

　for n=1,2,…,N do

　　$I_n^{(k+1)} = I_n^{(k)} + \eta \frac{\partial L}{\partial I_n^{(k)}}$ 由式(8)

　for i=1,2,…,T do

　　$T_i^{(k+1)} = T_i^{(k)} + \eta \frac{\partial L}{\partial T_i^{(k)}}$ 由式(9)

　$G^{(k+1)} = G^{(k)} + \eta \frac{\partial L}{G^{(k)}}$ 由式(6)

　k=k+1

　$p^{(k)}$ = MAP 由式(2)

　if $p^{(k)} - p^{(k-1)} \leqslant 0$ then

　　break

　until k≥itermax

return $\mathbf{U}=\mathbf{U}^{(k)}$，$\mathbf{I}=\mathbf{I}^{(k)}$，$\mathbf{T}=\mathbf{T}^{(k)}$

5　实验和分析

5.1　数据集

本文选用 Last.fm 数据集[14] 中的标签数据作为实验数据集，其中包括了 1 892 位用户给 12 523 位歌手使用 9 749 个标签标记的 186 479 条记录。为了避免标签数据过高的稀疏性对实验的影响，我们与之前的工作[15]一致，抽取该数据集的核心子集来进行仿真实验。具体抽取方法为：去除出现次数不大于 70 次的用户和歌手，同时去除出现次数小于 20 次的标签，得到 444 位用户、372 位歌手和 263 个标签的 41 684

个标签记录。占总标记次数的 22.35%，稠密度为 0.0959%。此外，我们从中随机选择 80% 的数据作为训练集，剩余的 20%作为测试集。

5.2 实验设置

本文选用 Pop 方法和 TD 方法作为实验的对比方法。其中，Pop 方法在第 2 节有所介绍，而 TD 方法(Tucker Decomposition)是基础的张量 Tucker 分解，通过 HOOI 方法进行求解，本文使用 tensorly 库[16]中提供的方法进行计算。

5.3 实验结果与分析

(1)与其他方法的对比

我们选取核张量维度为 10，学习率为 0.01，在不同 top_K 的取值下，与 Pop 和 TD 的对比如图 3 所示。

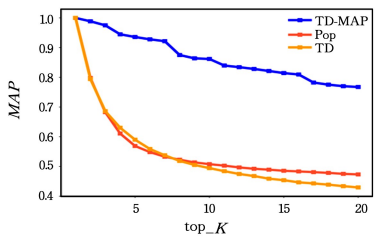

图 3 TD-MAP 与其他方法的效果对比图

Fig. 3 Comparison of TD-MAP and other methods

通过图 3 可以看出，本文方法 TD-MAP 较 Pop 方法和 TD 方法在 top_K>1 的推荐上都有着较好的表现。此外，3 种方法的 MAP 都在随着 top_K 的增大加而减小，这是因为推荐列表变长后召回了更多相关的物品而导致 MAP 相应地下降，分别地，TD-MAP 的 MAP 值随着 top_K 的增大近乎线性地下降，而 Pop 和 TD 方法的在 K 值较小时的效果呈指数型下降，这表明了 TD-MAP 具有较好的稳定性，在 K 值增大时效果也不会剧烈下降。此外，在 K 值较小时，TD 方法的性能优于 Pop 方法，而随着 K 值的增大，Pop 方法好于 TD 方法，这既能表明 Pop 方法的有效性，即给用户推荐当下热门的标签能一定程度上解决标签推荐的问题，也能反映出 Pop 方法比 TD 方法要略稳定一些。

(2)核张量的维数对结果的影响

我们在实验过程中固定学习率为 0.01，选取 top_K 为 5，对核张量的维数分别设为 5，10，15，20，25 和 30，实验结果如图 4 所示。

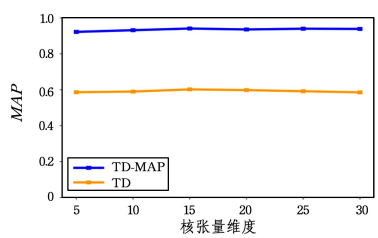

图 4 核张量维度对结果的影响图

Fig. 4 Core tensor dimension and effect

由图 4 可知，核张量维度 5～30 变化时，TD-MAP 和 TD 方法的结果都收敛到了一个值附近，仅有略微的波动来自于随机的初始化，对结果影响非常小。TD 方法是由 tensorly 实现的，在算法收敛时返回，那么造成 TD-MAP 在这些核张量维度的取值都收敛的情况是否是因为过大的学习率带来的呢？我们将继续讨论。

(3)学习率对训练的影响

在该实验中，我们固定核张量为 10，top_K 为 5，取不同的学习率来进行对比，实验结果如图 5 所示。

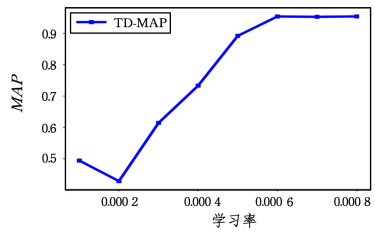

图 5 学习率取值对结果的影响

Fig. 5 Learning rate and effect

图 5 中，当学习率大于 0.0006 时，MAP 已经收敛到了一个值附近，说明算法只需要较小的学习率就能够达到收敛状态。同时，该算法使用较小的学习率就能够收敛，是因为数据过于简单，都是 0/1 型的值，所以拟合该数据不需要太大的学习率，在本文其他实验中我们使用了 0.01 的学习率是为了加快训练的收敛速度。由此，核张量的维度对结果的影响也就很小了，而我们知道核张量的维度是决定张量分解的困难程度的，因此核张量的维度可能会对运行时间产生较大的影响。我们将在下一节中针对核张量维度对事件的影响进行讨论。

(4)运行时间

在该部分实验中，我们固定学习率为 0.01，top_K 为 10，核张量维度分别为 5，10，15，20，25 和 30，实验所使用的机器 CPU 为 Intel Xeon E5-2620，内存为 128 GB。本文算法的总运行时间等于训练时间与测试时间的和，部分核张量维度取值下的运行时间如表 2 所列。

表 2 不同核张量维度与运行时间统计表

Table 2 Different core tensor dimensions and running time statistics

(单位：s)

核张量维度	训练时间	测试时间	总时间
10	6519.93	1.57	6521.50
20	7616.33	1.69	7618.02
30	9715.49	0.98	9716.47

从表 2 中可以看出，算法运行的时间随着核张量维度的增加而增加。其中，因为测试时间仅仅包括了由核张量和因子矩阵构建参数张量的时间和根据该参数张量从测试集中取值的时间，所以花费的时间很少，仅在 1s 左右。因此，我们在此仅关注对训练时间影响，如图 6 所示。

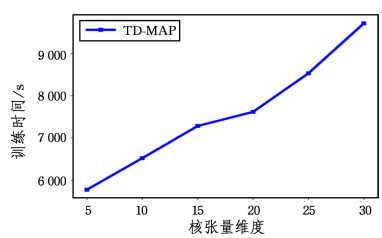

图 6 核张量维度对训练时间的影响

Fig. 6 Effect of core tensor dimension on training time

从图 6 可以看出，算法的训练时间受核张量维度的影响较大，这是由张量分解的性质决定的，需要分解的核张量的维度越大，其分解难度越高，所消耗的时间就越长。

结束语 本文提出了基于张量 Tucker 分解模型来优化

列表级排序的 TD-MAP 方法，不仅充分利用了“用户-物品-标签”这样的标签数据的三维结构信息，而且采用优化列表排序的方式，在个性化标签推荐中取得了不错的效果。此外，在实验部分充分讨论了算法的各个超参数对算法的影响，其中对实验结果影响最大的是 top-K 的取值，这也是比较符合实际的，因为在推荐列表的长度增加的情况下，推荐项随之增多，而效果也有所下降。

另外，还有一个值得关注的点是算法的运行时间，我们可以很容易地看出算法的测试时间是很短的，但训练部分却是比较耗时的。在实际的应用中，我们一般是离线训练模型，而在线上使用已训练好的模型进行实时的推荐，即我们测试部分的工作。因此，算法在实际使用时的时间还是很快的，但是，为了进一步加强算法的可用性，例如当数据集更新后能够及时地更新模型，我们在后续的工作中将重点关注如何加快算法的训练，缩短训练时间。

参 考 文 献

[1] ZENG H，HU Q，GAN X X. Method for tag recommendation of tensor decomposition based on multiple relationships [J]. Application Research of Computers，2019，36(10)：2907-2910.

[2] LI G，WANG S，LI Z Y，et al. Personalized Tag Recommendation Algorithm Based on Tensor Decomposition [J]. Computer Science，2015，42(2)：267-273.

[3] KOLDA T G，BADER B W. Tensor Decompositions and Applications[J]. Siam Review，2009，51(3)：455-500.

[4] 项亮. 推荐系统实践[M]. 北京：人民邮电出版社，2012.

[5] LIAO Z F，WANG C Q，LI X Q. Tag Recommendation and New User Tag Recommendation Algorithms Based on Tensor Decomposition [J]. Journal of Chinese Computer Systems，2013，34(11)：2472-2476.

[6] RENDLE S，SCHMIDTTHIEME L. Pairwise interaction tensor factorization for personalized tag recommendation[C]// Proceedings of the Third International Conference on Web Search and Web Data Mining. New York：Association for Computing Machinery，2010：81-90.

[7] SHI Y，KARATZOGLOU A，BALTRUNAS L，et al. TFMAP：optimizing MAP for top-n context-aware recommendation[C]// Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York：Association for Computing Machinery，2012：155-164.

[8] LIU T Y. Learning to Rank for Information Retrieval[J]. Foundations and Trends in Information Retrieval，2009，3(3)：225-331.

[9] LI H. Learning to Rank for Information Retrieval and Natural Language Processing[M]. California：Morgan & Claypool Publishers，2011.

[10] LIU J，WU C，XIONG Y，et al. List-wise probabilistic matrix factorization for recommendation [J]. Information Sciences，2014，278：434-447.

[11] CAO Z，QIN T，LIU T，et al. Learning to rank：from pairwise approach to listwise approach[C]// Proceedings of the 24th International Conference on Machine Learning. New York：Association for Computing Machinery，2007：129-136.

[12] HUANG Z H，ZHANG J W，TIAN C Q，et al. Survey on learning-to-rank based recommendation algorithms [J]. Journal of Software，2016，27(3)：691-713.

[13] CHAPELLE O，WU M. Gradient descent optimization of smoothed information retrieval metrics [J]. Information Retrieval，2010：13(3)：216-235.

[14] CANTADOR I N，BRUSILOVSKY P，KUFLIK T. Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2010)[C]// Proceedings of the 2010 ACM Conference on Recommender Systems. New York：Association for Computing Machinery，2010：375-376.

[15] CHEN M M，XUE K J. Personalized Recommendation Algorithm Based on Modified TensorDecomposition Model [J]. Data Analysis and Knowledge Discovery，2017，1(3)：37-45.

[16] JEAN K，YANNIS P，ANIMA A，et al. TensorLy：Tensor Learning in Python[J]. Journal of Machine Learning Research，2019：20：1-6.

YANG Yang，born in 1995，postgraduate. His main research interests include tensor decomposition and distributed computing.

ZHOU Wei，born in 1974，Ph.D，professor，Ph.D supervisor，is a member of China Computer Federation. His main research interests include deep learning and its application in bioinformatics.