文章编号:1672-3961(2025)01-0024-06 DOI:10.6040/j.issn.1672-3961.0.2023.285

一种面向矩阵分解模型的推荐系统训练加速方法

段圣宇[1],吴伊宁[1],赛高乐[2]
(1.上海大学计算机工程与科学学院, 上海 200444; 2.深圳技术大学集成电路与光电芯片学院, 广东 深圳 518118)

摘要:为降低矩阵分解(matrix factorization, MF)模型面向推荐系统应用的训练时间,特别针对细粒度稀疏的特征矩阵在训练过程中存在大量无效乘法运算的问题,提出一种基于特征矩阵联合稀疏性进行近似计算的训练加速方法。基于隐因子向量稀疏性强弱基本不变的特点,提出在模型训练初期,根据隐因子向量的稀疏性,对特征矩阵重新排列;在训练过程中,采用早停法,避免无效乘法运算。试验结果表明,模型训练过程中乘法运算次数可最多降低 28.41%,加速前后评分预测值相关系数约 0.95。所提出方法可以保证预测准确性小幅降低的同时,显著减少训练中的乘法运算次数,针对更大规模的矩阵分解模型训练,能实现更好的加速效果。

关键词:推荐系统;矩阵分解;稀疏性;算法加速;近似计算
中图分类号:TP391 **文献标志码**:A
引用格式:段圣宇,吴伊宁,赛高乐. 一种面向矩阵分解模型的推荐系统训练加速方法[J]. 山东大学学报(工学版),2025,55(1):24-29.
DUAN Shengyu, WU Yining, SAI Gaole. Algorithmic acceleration of matrix factorization based recommendation system[J]. Journal of Shandong University (Engineering Science), 2025, 55(1):24-29.

Algorithmic acceleration of matrix factorization based recommendation system

DUAN Shengyu[1], WU Yining[1], SAI Gaole[2]
(1. School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China; 2. College of Integrated Circuits and Optoelectronic Chips, Shenzhen Technology University, Shenzhen 518118, Guangdong, China)

Abstract: In order to reduce the training time of matrix factorization (MF) based recommendation system, specifically considering the fine-grained structured sparsity of the decomposed matrices, which caused unnecessary multiplications and increased the overall time of training process, an algorithmic acceleration method, based on joint sparsity of the decomposed matrices and approximate matrix multiplications, was proposed. According to an observation that the trends of sparsity on all latent vectors generally hold, an algorithm to rearrange the feature matrices during the first a few training epochs, based on joint sparsity, was proposed. An early stop algorithm was applied to eliminate unnecessary multiplications during the training process. The experimental results showed the total number of multiplications could be reduced by up to 28.41%, and the correlation between the predicted ratings produced by the conventional and proposed methods was around 0.95. The acceleration method could greatly reduce the total number of multiplications during MF training process, causing a minimal error, and more multiplications were expected to be eliminated for the recommendation systems with larger scales.

Keywords: recommendation system; matrix factorization; sparsity; algorithmic acceleration; approximate computing

0 引言

推荐系统作为缓解当下信息爆炸的重要工具,在互联网领域得到了广泛的应用,其应用场景包括搜索引擎、社交网络、网上零售、流媒体等[1-2]。协同过滤是目前推荐系统的主流手段,可分为基于内存协同过滤和基于模型协同过滤。前者利用相似度函数找到具有行为共性用户或项目,将在用户行为上“相似”的用户或项目作为推荐结果,其存在内

收稿日期:2023-11-16
基金项目:计算机体系结构国家重点实验室开放课题资助项目(CARCH201909)
第一作者简介:段圣宇(1991—),男,湖北武汉人,讲师,硕士生导师,博士,主要研究方向为人工智能软硬件协同设计、大规模集成电路设计等。E-mail:sduan@ shu.edu.cn

存占用率高、难以发现实体间复杂联系等问题[3];后者采用机器学习模型,能够更有效捕捉实体间非线性、隐含联系,其应用与研究更加广泛[3]。

在基于模型的协同过滤中,矩阵分解(matrix factorization, MF)具有较强的可扩展性且易于实现,是目前最普遍的机器学习模型[4]。矩阵分解将用户-项目评分矩阵 $\boldsymbol{R}_{mn}$ 表示成两个低秩的特征矩阵 $\boldsymbol{P}_{mk}$ 和 $\boldsymbol{Q}_{kn}$ 的乘积,其中 m、n 和 k 分别表示用户、项目和隐因子的数量。矩阵分解通过计算 $\boldsymbol{P}_{mk}$ 和 $\boldsymbol{Q}_{kn}$ 的乘积对 $\boldsymbol{R}_{mn}$ 中缺失的数据进行补充,以此预测用户的偏好行为。矩阵分解模型训练过程所包含的乘法运算次数与 m 和 n 的大小呈线性关系,每次迭代都需要计算 $\boldsymbol{P}_{mk}$ 和 $\boldsymbol{Q}_{kn}$ 的乘积,能够计算预测误差,更新特征参数。

在推荐系统中,矩阵分解最初利用奇异值分解(singular value decomposition, SVD)实现[5]。传统 SVD 算法在面临大规模数据时,计算效率低,需要填充评分矩阵中的缺失值,计算准确性受到人为因素的影响。FunkSVD[6]、SVD++[7] 和 BiasSVD[8] 等一系列算法在一定程度上解决了传统 SVD 的这些问题。一些工作对上述算法进行了改进,使基于矩阵分解的推荐算法在预测准确性和运算速度上得到了提高。文献[9]提出利用深度学习优化算法 RMSProp(root-mean-square prop)对传统 FunkSVD 算法进行改进,降低了数据稀疏性,解决了因迭代振荡导致无法收敛的问题,提高预测准确性;利用图形处理器(graphics processing unit, GPU)实现基于 FunkSVD 算法的并行运算,提高了算法运算速度。

矩阵分解模型中,利用随机梯度下降法(stochastic gradient descent, SGD)[10]、坐标梯度下降法(coordinate gradient descent, CGD)[11] 或交替最小二乘法(alternating least squares, ALS)[8] 对特征值进行更新。随机梯度下降法性能取决于学习率设置;坐标梯度下降法容易陷入局部最优解[12];交替最小二乘法收敛速度比较快[13],其算法复杂度较高,处理大批量数据时效率不高。随机梯度下降法的复杂度低,应用最为广泛,本研究采用随机梯度下降法构建矩阵分解模型。

为合理地设置随机梯度下降法的学习率,文献[14]提出了一种自适应学习率策略-AALRSMF,其在训练过程中自适应调整学习率,对超参数选择具有更强鲁棒性。

传统随机梯度下降法串行运算特性,使其在多数据流计算机上难以利用并行处理技术实现加速运算。为了提高基于矩阵分解模型的推荐系统训练效率,针对 GPU 或分布式计算而提出的改进的随机梯度下降法成为目前提高算法运算速度的主流途径[12,15]。上述矩阵分解模型的训练加速方法依赖于多部件、多设备的并行处理,不适用于数据规模大、硬件资源有限的应用领域。

随着互联网产业规模的快速扩大,推荐系统的信息总量和模型尺寸也不断增长[16]。为实现快速、准确的模型构建与训练,在新数据、新信息不断产生背景下满足模型快速更新、迭代需求,本研究提出一种面向矩阵分解模型的推荐系统训练加速方法。该方法采用近似计算,通过分析矩阵 $\boldsymbol{P}_{mk}$ 和 $\boldsymbol{Q}_{kn}$ 的联合稀疏性,对其中稀疏的隐因子向量进行早停法(early stop)处理,在保证高预测准确性的同时,降低矩阵分解模型的训练时间。提出的方法在算法层面避免矩阵分解模型训练过程中的无效运算,适用于不同硬件资源要求的场景,具有更强的普适性。

1　推荐系统中的细粒度稀疏特征矩阵

稀疏性,指数据或模型参数中零元素的数量。稀疏矩阵普遍存在于包括矩阵分解、神经网络等的机器学习算法的应用中[17-18]。在推荐系统中,稀疏数据由用户-项目序列中未发生交互的项目所产生,即评分矩阵 $\boldsymbol{R}_{mn}$ 中初始缺失的元素。推荐系统的目标是通过算法预测并补充 $\boldsymbol{R}_{mn}$ 中缺失元素,数据稀疏性在推荐系统中无可避免。基于矩阵分解的推荐系统中,稀疏特征存在于特征矩阵 $\boldsymbol{P}_{mk}$ 和 $\boldsymbol{Q}_{kn}$ 中,指的是特征矩阵中存在较多零元素隐因子向量。特征稀疏性反映了特定特征与学习任务间相关程度,其可通过对模型参数、架构、训练方法等合理设计进行调整。提出的方法将利用矩阵分解模型中特征稀疏性,避免模型训练过程中无效运算,缩短训练时间。

一般来说,矩阵分解模型中特征稀疏性随隐因子数量 k 的增加而增大。过小或过大的隐因子数量会导致模型欠拟合或过拟合,降低模型的预测准确性。为实现高准确性的预测,需合理选择隐因子数量。

基于数据集 MovieLens 100k 构建了矩阵分解模型[19]。模型训练参数如表 1 所示。利用构建的模型对测试集中用户评分进行预测,图 1 展示了随着 k 增大,部分用户的预测评分与实际评分之间皮尔森相关系数(Pearson correlation coefficient)的变化。皮尔森相关系数越高,反映了模型针对特定用户进行评分预测任务拟合程度越高,预测评分越接

近实际评分。从图 1 中可以看到,随着 k 增大,模型针对部分用户(如用户 3)评分的预测准确性大幅提升;对其他用户,模型的预测准确性随着 k 增大基本持平,甚至呈下降趋势(如用户 6)。不同用户在训练集中评分数据的稀疏性不同,不同用户评分标准各异,反映了推荐系统中较强的个性化特征。特定模型针对不同用户评分行为进行学习,其拟合程度不同。

表 1 基于数据集 MovieLens 10^5 构建的矩阵分解模型训练参数设置

Table 1 Hyperparameter setups when training MF models for MovieLens 10^5

用户数量	项目数量/个	训练集评分数量/个	隐因子数量/个	迭代次数/次	学习率/%	测试集评分数量/个
943	1 682	90 570	10、20、30、40、50	300	10	9 430

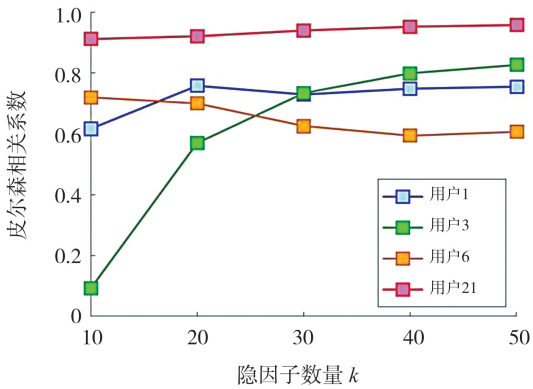
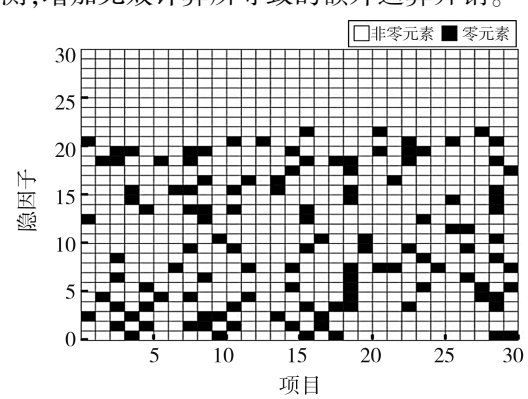


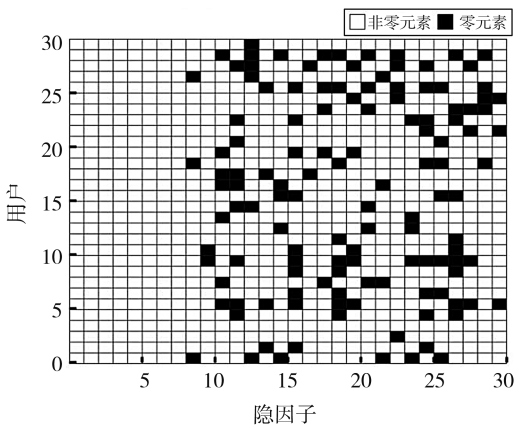
图 1 预测/实际评分皮尔森相关系数与 k 的关系

Fig.1 Pearson correlation coefficient between predicted and actual ratings for different k

上述试验结果说明,当矩阵分解模型中隐因子数量 k 固定时,模型在不同用户的评分预测任务中拟合程度不同。各隐因子向量的特征稀疏性也将存在差异。图 2 展示了基于 MovieLens 10^5 并选取 k 等于 30 所构建的矩阵分解模型中的特征矩阵 $\boldsymbol{P}_{mk}$ 和 $\boldsymbol{Q}_{kn}$,其中所有小于阈值 0.06 的元素被定义为零元素。零元素所参与的乘法运算由于其结果为零或接近于零,是无效运算,对矩阵乘法运算结果的影响很小。从图 2 可以看到,零元素在特征矩阵 $\boldsymbol{P}_{mk}$ 和 $\boldsymbol{Q}_{kn}$ 中呈不规则分布,各隐因子向量的稀疏性各异。此现象被称为非结构化的细粒度稀疏性。特征矩阵中细粒度稀疏性使其在算法层面上无法通过特征选择降低运算次数。零元素的位置不可预测,增加无效计算所导致的额外运算开销。



(a) $\boldsymbol{P}_{mk}$

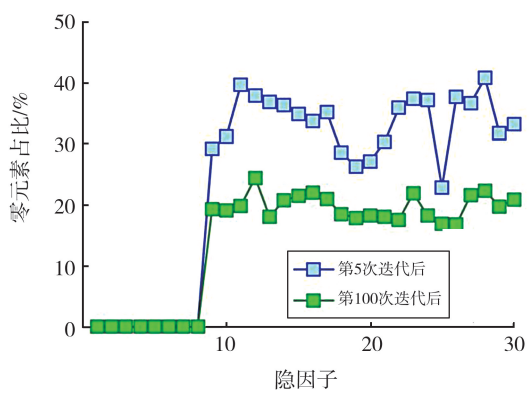


(b) $\boldsymbol{Q}_{kn}$

图 2 特征矩阵中的细粒度稀疏性

Fig.2 Fine-grained sparse structures of $\boldsymbol{P}_{mk}$ and $\boldsymbol{Q}_{kn}$

在矩阵分解模型训练过程中,各隐因子向量稀疏性随着迭代次数变化而变化。图 3 展示了基于 MovieLens 10^5 并选取 k 等于 30,分别经过 5 次和 100 次迭代后特征矩阵中隐因子向量的稀疏程度。从图 3 可以看到,各隐因子向量中零元素的占比随迭代次数增加而减小,反映了特征矩阵的稀疏性呈减小的趋势。各隐因子向量相对于其它隐因子向量,稀疏性强弱基本保持不变:稀疏程度较高的隐因子向量随迭代次数增加,相对于其它隐因子向量,仍保持着较高的稀疏程度。将利用各隐因子向量稀疏性强弱基本不变的特点,对矩阵分解模型的训练实现加速。



(a) $\boldsymbol{P}_{mk}$

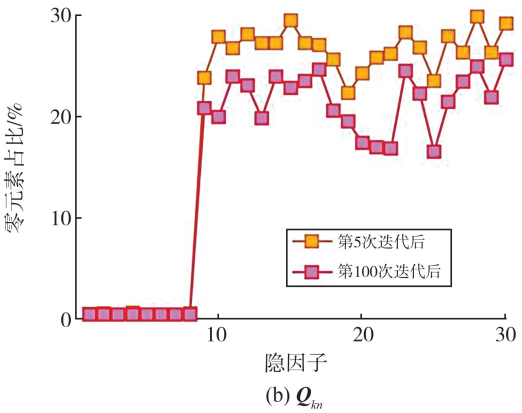


图 3　特征矩阵稀疏性随迭代次数的变化

Fig.3　Variation of feature matrix sparsity with the number of iterations

2　矩阵分解模型加速训练方法

2.1　基于稀疏性特征矩阵重排算法

提出在模型训练初期,根据各隐因子向量稀疏性,对特征矩阵 $\boldsymbol{P}_{mk}$ 和 $\boldsymbol{Q}_{kn}$ 进行重新排列。为综合考虑第 k 个隐因子向量在 $\boldsymbol{P}_{mk}$ 和 $\boldsymbol{Q}_{kn}$ 中的稀疏性,定义了联合特征稀疏性的概念。

定义 1　联合特征稀疏性。

$$\boldsymbol{S}_k = P(|\boldsymbol{p}_k|<T \cup |\boldsymbol{q}_k|<T), \tag{1}$$

式中:$\boldsymbol{S}_k$ 表示 $\boldsymbol{P}_{mk}$ 和 $\boldsymbol{Q}_{kn}$ 中第 k 个隐因子向量的联合特征稀疏性;$\boldsymbol{p}_k$ 和 $\boldsymbol{q}_k$ 分别代表 $\boldsymbol{P}_{mk}$ 中的第 k 列以及 $\boldsymbol{Q}_{kn}$ 中的第 k 行;T 表示零元素阈值:特征矩阵中绝对值小于 T 的元素被定义为零元素;$P(|\boldsymbol{p}_k|<T \cup |\boldsymbol{q}_k|<T)$ 表示 $|\boldsymbol{p}_k|<T$ 且 $|\boldsymbol{q}_k|<T$ 的概率,概率越大,则该隐因子向量的稀疏性越强。

基于 $\boldsymbol{S}_k$ 对特征矩阵进行重排的算法,如算法 1 所示。计算各隐因子向量的 $\boldsymbol{S}_k$。根据 $\boldsymbol{S}_k$,对 $\boldsymbol{P}_{mk}$ 和 $\boldsymbol{Q}_{kn}$ 的列和行分别进行重新排列,分别使 $\boldsymbol{S}_k$ 较小的行和列具有较小的列号和行号。由第 1 节所述,矩阵分解模型训练过程中,各隐因子向量,随迭代次数增加,其稀疏性强弱基本保持不变。特征矩阵的重排算法只需在训练初期执行一次。

算法 1　基于稀疏性的特征矩阵重排算法

输入　$\boldsymbol{P}_{mk}=\{\boldsymbol{p}_1\ \boldsymbol{p}_2\ \cdots\ \boldsymbol{p}_k\}$,$\boldsymbol{Q}_{kn}^{\mathrm{T}}=\{\boldsymbol{q}_1\ \boldsymbol{q}_2\ \cdots\ \boldsymbol{q}_k\}$;

参数零元素阈值 T;

输出　重新排列后的 $\boldsymbol{P}_{mk}$ 和 $\boldsymbol{Q}_{kn}$;

for $i=\{1,\cdots,k\}$

　　计算第 i 个隐因子的联合特征稀疏性 $\boldsymbol{S}_k$;

end for

for $i=\{1,\cdots,k-1\}$

　　for $j=\{2,\cdots,k\}$

　　　　if $S_i<S_j$

　　　　　　在 $\boldsymbol{P}_{mk}$、$\boldsymbol{Q}_{kn}$ 中分别将 $\boldsymbol{p}_i$、$\boldsymbol{q}_i$ 与 $\boldsymbol{p}_j$、$\boldsymbol{q}_j$ 交换位置;

　　　　end if

　　end for

end for

执行算法 1 后,$\boldsymbol{P}_{mk}$ 和 $\boldsymbol{Q}_{kn}$ 满足如下所示的条件:

$$\forall k_1,k_2 \in [1,k] \wedge k_1<k_2 : S_{k_1}<S_{k_2}。 \tag{2}$$

2.2　特征矩阵乘法早停算法

利用算法 1 对特征矩阵进行重新排列,联合特征稀疏性较小的列和行在 $\boldsymbol{P}_{mk}$ 和 $\boldsymbol{Q}_{kn}$ 中将分别具有较小列号和行号。计算机中矩阵乘法运算从较小行号和列号的行列逐行、逐列依次进行执行。针对重排后的 $\boldsymbol{P}_{mk}$ 和 $\boldsymbol{Q}_{kn}$,选取稀疏性较低的元素执行乘法运算。在依次执行某行列向量乘法运算过程中,当出现首个零元素时,对该行列向量乘法运算进行早停处理,将已得到的部分乘积求和结果作为该行列向量乘法运算的近似计算结果。具体算法如下。

算法 2　特征矩阵乘法的早停算法

输入　$\boldsymbol{P}_{mk}$ 中第 i 行 $\boldsymbol{p}_i$,$\boldsymbol{Q}_{kn}$ 中第 j 列 $\boldsymbol{q}_j$;

参数　零元素阈值 T;

输出　第 i 个用户对第 j 个项目的评分预测 $\boldsymbol{p}_i \cdot \boldsymbol{q}_j$;

$\boldsymbol{p}_i \cdot \boldsymbol{q}_j = 0$

for $t=\{1,\cdots,k\}$

　　if $|p_{it}|<T$ or $|q_{tj}|<T$

　　　　break

　　else

　　　　$\boldsymbol{p}_i \cdot \boldsymbol{q}_j = \boldsymbol{p}_i \cdot \boldsymbol{q}_j + p_{it} \cdot q_{tj}$

　　end if

end for

利用算法1根据隐因子向量稀疏性进行特征矩阵重排,利用算法 2 对特征矩阵乘法实现近似计算,该方法可在降低运算时间同时,实现较小误差。这是因为早停处理后未参与向量乘法运算隐因子,其稀疏性相对更强、零元素比例相对更高,对向量乘法运算结果影响较小。

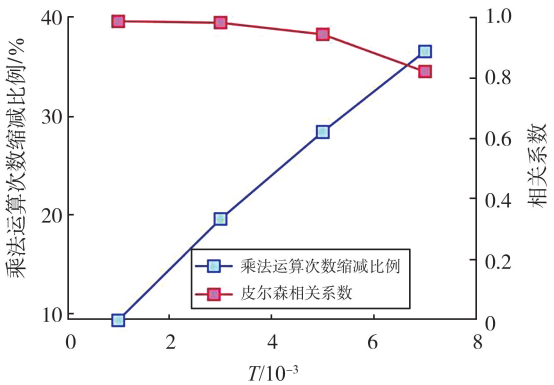
3　试验与结果分析

本章针对 MovieLens 10^5 和 Amazon Appliances

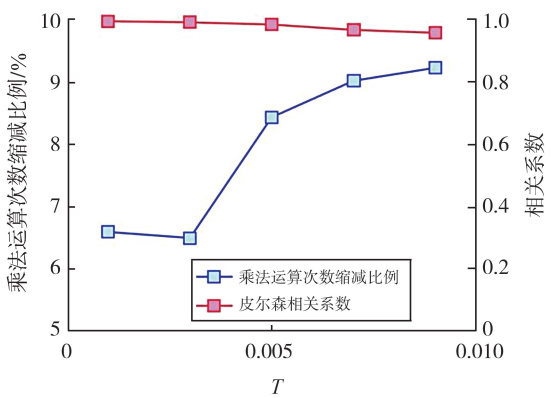
两个数据集,基于矩阵分解开源库 LibMF[20],构建基于矩阵分解模型的推荐系统,通过对比试验测试提出的训练加速方法效果与误差。MovieLens 10^5 数据集相关信息以及训练参数设置如表 1 所示。数据集 Amazon Appliances 包括30 252个用户针对 515 650 个项目进行的 602 776 次评分。针对 Amazon Appliances,矩阵分解模型训练参数设置与 MovieLens 10^5 相同,如表 1 所示。对上述两个数据集,随机选择 80%及 20%的样本分别作为训练集及测试集。

本试验的评估指标是皮尔森相关系数以及乘法运算次数缩减比例。利用皮尔森相关系数,计算未使用和使用训练加速方法所得到的评分预测值之间的相关性,而非两者之间的绝对误差,在不考虑两者评分预测值范围差异的前提下,反映了两者评分预测值的相似程度。测量前后两种情况的乘法运算次数,从而排除程序计算时间随机性影响。

图 4 展示了在选取 k 等于 30 且选取不同的零元素阈值 T 时,提出的方法在两个数据集上的表现。



(a) MovieLens 100k



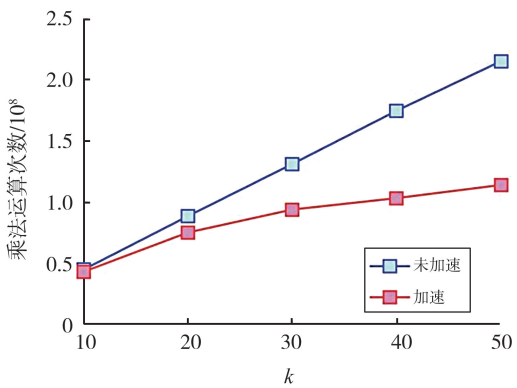
(b) Amazon Appliances

图 4 加速前后乘法运算次数及皮尔森相关系数随阈值 T 的变化

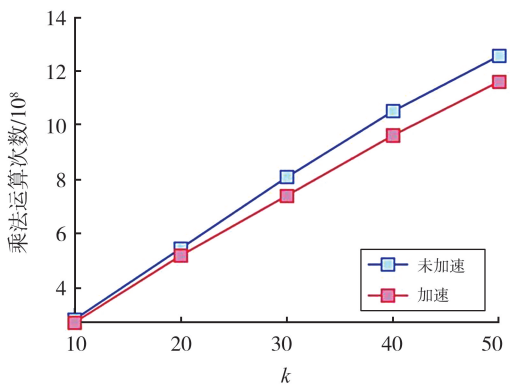
Fig.4 Total number of multiplications and correlation coefficients for different T

可以看到,随着 T 增大,乘法运算次数的缩减比例呈线性增长的趋势,说明了训练模型所需的时间呈不断降低的趋势。随着 T 增大,皮尔森相关系数呈降低趋势,说明提出的加速训练方法中早停算法导致了一定的预测误差。相比于乘法运算次数缩减比例增大,皮尔森相关系数降低并不显著;从图 4 可以看到,当加速前后评分预测值的相关系数在0.95 左右时,在针对 MovieLens 10^5 和 Amazon Appliances 训练模型的过程中,乘法运算次数分别降低了 28.41%和 9.23%。通过对零元素阈值 T 的合理选取,可根据实际需求,在较少的训练时间和较低的误差之间进行折中。

图 5 展示了在选取 T 等于 0.005 且选取不同的隐因子数量 k 时,提出的方法在两个数据集上所实现的加速效果。当 k 较小时,提出的方法并未显著减少乘法运算次数。随着 k 增大,乘法运算次数缩减比例逐渐增大。这是因为当 k 较小时,各隐因子向量相对稠密,在进行向量乘法运算时,只会对少量的零元素执行早停处理;随着 k 增大,隐因子向量更加稀疏,提出的方法对更多零元素执行早停处理,能减少更多的乘法元素次数,实现更好加速效果。



(a) MovieLens 100k



(b) Amazon Appliances

图 5 加速前后乘法运算次数及皮尔森相关系数随阈值 T 的变化

Fig.5 The number of multiplications and Pearson correlation coefficient as a function of the threshold T before and after acceleration

4　结论

提出了一种面向矩阵分解模型的推荐系统训练加速方法。用户评分行为具有较强个性化,在矩阵分解模型中,存在细粒度的特征稀疏性。细粒度稀疏的特征矩阵在训练过程中存在大量由零元素导致的无效乘法运算,增加模型训练所需时间。针对上述问题,提出了基于稀疏性特征矩阵重排算法和特征矩阵乘法早停算法,有效降低零元素导致的无效乘法运算次数,同时保证较小的准确性损失。试验结果表明,在预测准确性小幅降低时,上述方法能够显著减少训练所需乘法运算次数。针对更大规模矩阵分解模型的训练,上述方法能实现更好的加速效果。

参考文献:

[1] Amazon Web Services Inc. Amazon personalize[EB/OL]. (2023-12-16)[2023-02-16]. https://aws.amazon.com/personalize/.

[2] COVINGTON P, ADAMS J, SARGI E. Deep neural networks for YouTube recommendations [C]// Proceedings of the 10th Conference on Recommender Systems. New York, USA: ACM, 2016: 191-198.

[3] 王磊,熊于宁,李云鹏,等. 一种基于增强图卷积神经网络的协同推荐模型[J]. 计算机研究与发展, 2021, 58(9): 1987-1996.
WANG Lei, XIONG Yuning, LI Yunpeng, et al. A collaborative recommendation model based on enhanced graph convolutional neural network [J]. Journal of Computer Research and Development, 2021, 58(9): 1987-1996.

[4] HE Xiangnan, LIAO Lizi, ZHANG Hanwang, et al. Neural collaborative filtering[C]//Proceedings of the 26th International Conference on World Wide Web. New York, USA: ACM, 2017: 173-182.

[5] RESNICK P, VARIAN H R. Recommender systems[J]. Communications of the ACM, 1997, 40(3): 56-58.

[6] SALAKHUTDINOV R, MNIH A. Probabilistic matrix factorization[C]//Proceedings of the 20th International Conference on Neural Information Processing Systems. New York, USA: ACM, 2007: 1257-1264.

[7] KOREN Y. Factor in the neighbors: Scalable and accurate collaborative filtering [J]. ACM Transactions on Knowledge Discovery from Data, 2010, 4(1): 1-24.

[8] KOREN Y, BELL R, VOLINSKY C. Matrix factorization techniques for recommender systems [J]. Computer, 2009, 42(8): 30-37.

[9] YUE Xiaochen, LIU Qicheng. Parallel algorithms of improved FunkSVD based on GPU[J]. IEEE Access, 2022, 10: 26002-26010.

[10] BOTTOU L. Large-scale machine learning with stochastic gradient descent[C]//Proceedings of the 19th International Conference on Computational Statistics. Berlin, Germany: Springer, 2010: 177-186.

[11] YU H F, HSIEH C J, SI S, et al. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems[C]// 12th International Conference on Data Mining. New Jersey, USA: IEEE, 2012: 765-774.

[12] CHIN W S, ZHUANG Y, JUAN Y C, et al. A fast parallel stochastic gradient method for matrix factorization in shared memory systems [J]. ACM Transactions on Intelligent Systems and Technology, 2015, 6(1): 1-24.

[13] TAN Wei, CAO Liangliang, FONG L. Faster and cheaper: parallelizing large-scale matrix factorization on GPUs [C]//Proceedings of the 25th International Symposium on High-Performance Parallel and Distributed Computing. New York, USA: ACM, 2016: 219-230.

[14] WEI Feng, GUO Hao, CHENG Shaoyin, et al. AALRSMF: An adaptive learning rate schedule for matrix factorization[C]//Asia-Pacific Web Conference. Berlin, Germany: Springer, 2016: 410-413.

[15] XIE X L, TAN W, FONG L, et al. Cumf _ sgd: parallelized stochastic gradient descent for matrix factorization on GPUs [C]//Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing. New York, USA: ACM, 2017: 79-92.

[16] LIAN Xiangru, YUAN Binhang, ZHU Xuefeng, et al. Persia: an open, hybrid system scaling deep learning-based recommenders up to 100 trillion parameters[C]// Proceedings of the 28th SIGKDD Conference on Knowledge Discovery and Data Mining. New York, USA: ACM, 2022: 3288-3298.

[17] DONÀ J, GALLINARI P. Differentiable feature selection, a reparameterization approach[C]//Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference. Berlin, Germany: Springer, 2021: 13-17.

[18] WU Yining, SAI Gaole, DUAN Shengyu. Work-in-Progress: Accelerated matrix factorization by approximate computing for recommendation system [C]//International Conference on Embedded Software. Shanghai, China: IEEE, 2022: 1-2.

[19] HARPER F M, KONSTAN J A. The MovieLens datasets: history and context[J]. ACM Transactions on Interactive Intelligent Systems, 2016, 5(4): 1-19.

[20] CHIN W S, YUAN B W, YANG M Y, et al. LIBMF: a library for parallel matrix factorization in shared-memory systems [J]. Journal of Machine Learing Research, 2016, 17(1): 2971-2977.

(编辑:陈燕)