中国科学 信息科学
SCIENTIA SINICA Informationis

Efficient solution of SVD recommendation model with implicit feedback

蔡剑平, 雷蕴奇, 陈明明, 王宁 and 张双越

Citation: 中国科学: 信息科学; doi: 10.1360/SSI-2019-0107

View online: http://engine.scichina.com/doi/10.1360/SSI-2019-0107

Published by the 《中国科学》杂志社

中国科学: 信息科学

SCIENTIA SINICA Informationis

论文

带有隐式反馈的SVD推荐模型高效求解算法

蔡剑平[1*], 雷蕴奇[2], 陈明明[1], 王宁[1], 张双越[1]

1. 厦门华厦学院信息与智能机电学院, 厦门361021
2. 厦门大学信息学院, 厦门361005
* 通信作者. E-mail: caijp@hxxy.edu.cn

国家自然科学基金面上项目(批准号: 61671397)和福建省中青年教师教育科研项目(批准号: JT180779) 资助项目

摘要 作为推荐系统的重要组成部分, 协同过滤已成为了当今主流的推荐方法之一. 其中基于潜在因子的协同过滤常采用SVD推荐模型分析用户喜好. 近年来, 随着SVD推荐模型研究的深入, SVD++, TrustSVD等一类带有隐式反馈的SVD推荐模型被相继提出. 此类模型能更有效地从有限的数据源中挖掘有用信息并取得了较好的效果, 因此受到了人们广泛关注. 然而, 现有文献大多关注于模型设计, 缺乏专门针对带有隐式反馈的SVD推荐模型的高效求解算法. 为此, 本文首先研究了一般性的SVD推荐模型梯度求解框架, 然后以SVD++推荐模型为突破口, 基于块梯度下降方法设计了高效求解算法BCDSVD++并解决了容量矩阵求逆, 稀疏数据优化处理等两个关键问题. 实验表明, 本文所设计的BCDSVD++算法具有比传统的并行梯度下降法更高效的求解效率及收敛能力.

关键词 SVD推荐模型, 隐式反馈, SVD++, 块坐标下降法, 协同过滤

1 引言

随着互联网技术的发展, 推荐系统已成为电子商务领域中不可或缺的重要组成部分. 通过海量用户数据分析, 推荐系统能够挖掘用户的潜在消费需求并向其推荐电影, 音乐, 书籍等商品. 面对激烈的商业竞争环境, 推荐系统能否高效, 精准地抓住用户需求对于电商平台能否提升用户服务质量, 获取更多客户至关重要. 作为一种重要的推荐策略, 协同过滤[1]通过大量收集用户的偏好或评价数据预测用户对各商品评价, 然后再根据预测评分向用户提供个性化的推荐服务. 相比于其他推荐策略, 协同过滤方法仅需少量用户评价信息就能实现较精准推荐且推荐结果具有新颖性, 适用于推荐复杂程度高的商品. 因此该推荐策略被广泛应用于Amazon[2], Netflix[3]和Spotify[4]等知名企业.

作为一类重要的协同过滤方法, 基于奇异值分解(Singular Value Decomposition, SVD)的推荐模型[5]利用潜在因子分析的方法预测用户评分. 该模型认为每个用户及物品存在均存在影响评分的潜

引用格式: 蔡剑平, 雷蕴奇, 陈明明, 等. 带有隐式反馈的SVD推荐模型高效求解算法. 中国科学: 信息科学, 在审文章
Cai J P, Lei Y Q, Chen M M, et al. Efficient solution of SVD recommendation model with implicit feedback (in Chinese). Sci Sin Inform, for review

在因子. 例如, 在电影的推荐场景中, 电影的潜在因子所代表特征可以是类型, 风格, 年代等, 当用户感兴趣的特征与电影的特征相符时通常会给出更高的评价. 奇异值分解过程以用户-物品评分矩阵$\boldsymbol{V}$作为输入, 矩阵中元素表示用户对物品的评分, 不过通常只有极少的评分是已知的. 然后通过特定的优化算法将已评分矩阵$\boldsymbol{V}$分解为用户特征矩阵$\boldsymbol{U}$以及物品特征矩阵$\boldsymbol{M}$使得$\boldsymbol{V} \approx \boldsymbol{U}^T\boldsymbol{M}$, 特征矩阵的每一列分别表示用户或物品的特征向量.

随着协同过滤技术的发展, Paterek等人[6]完善了上述推荐模型并提出了BiasSVD推荐模型. 该模型在考虑了用户和物品中影响评分的独立因素, 更加细致地刻画了用户及物品的潜在特征, 是最经典的传统SVD推荐模型之一. 受模型限制, 上述SVD推荐模型只能利用已有的评分信息训练模型. 然而, 有些关联因子已被论证对于预测用户评价行为是有益的, 传统的SVD推荐模型难以充分利用这些信息实现更精准的评分预测. 考虑了其他因素, Koren等人[7]利用了用户行为的隐式反馈并提出了SVD++推荐模型. 结合邻域模型与潜在因子模型的优势, SVD++推荐模型能够根据更丰富的隐式反馈推断用户偏好并通过用户行为分析其潜在意见. 例如, 某用户看了某位导演拍的许多部电影表明该用户可能喜欢该导演拍的电影. 由于考虑用户隐式反馈, SVD++相比于传统的SVD能够更加深入地挖掘数据中的潜在信息, 本文称此类SVD推荐模型为带有隐式反馈的SVD推荐模型. 除了SVD++推荐模型, TrustSVD[8]也是一种重要的带有隐式反馈的SVD推荐模型. 该模型是SVD++推荐模型的拓展并进一步考虑了社交网络中信任关系的隐式反馈, 不仅获得了更准确的评分预测能力, 而且有效解决了数据冷启动问题. 此外, 还有许多带有隐式反馈的SVD推荐模型. 如拓展了Social MF[9]并考虑隐藏主题的MR3++推荐模型[10], 利用了双信任机制的EITrustSVD模型[11]等.

目前, 大多数关于SVD推荐模型的研究聚焦于问题分析及模型设计并采用梯度下降法求解模型. 作为经典的优化方法, 梯度下降法适用性强, 广泛应用于各类优化问题[12]. 然而, 其不足之处在于梯度下降法难以根据具体问题设计针对性的高效求解方法. 采用了梯度下降法实现的SVD推荐算法往往存在较大的性能提升空间. 针对传统SVD推荐模型, Hu等人[13]提出了基于交替最小二乘法(Alternating Least Squares,ALS)极大地提升了模型求解性能. 其思想是通过交替地利用最小二乘法优化各个特征向量直至算法收敛. 作为一种特殊的块坐标下降法, 交替最小二乘法具有一定的局限性, 仅适用于求解传统的SVD推荐模型, 难以应用于带有隐式反馈的SVD推荐模型, 但其思想对于进一步研究带有隐式反馈的SVD推荐模型的高效算法具有重要的借鉴意义. 实际应用中, 带有隐式反馈的SVD推荐模型常采用梯度下降法结合高性能计算平台求解模型[14~16]. 然而, 此类方法实际以昂贵的计算资源为代价换取高性能计算能力, 并非通过针对性的算法研究实现计算性能的提升. 因此, 带有隐式反馈的SVD推荐模型仍待进一步研究高效模型求解算法. 为此本文研究了面向任意SVD推荐模型的梯度求解框架并基于块坐标下降法(block coordinate descent,BCD) [17~19]设计了SVD++推荐模型的高效求解算法. 研究过程中, 本文以矩阵理论为基础建立算法模型并结合矩阵微积分[20~22], 矩阵求逆[23, 24]等相关理论实现算法设计. 矩阵理论不仅能更加简洁地描述算法模型而且处处体现了系统性的研究思想. 基于上述思路, 本文完成了以下研究工作:

1) 基于SVD推荐模型求解问题研究了一般化的梯度求解框架. 在此基础上, 针对SVD++推荐模型求解了关于各个特征矩阵(向量)的梯度并对各梯度的性质进行理论分析.

2) 基于上述分析结果研究了算法设计过程中关于容量矩阵求逆以及稀疏矩阵优化两个关键科学问题, 通过理论分析和算法设计为高效算法的设计提供支持.

3) 基于块梯度下降法设计了SVD++推荐模型的高效求解算法BCDSVD++, 并采用Movielens

系列数据集及FilmTrust数据集通过实验验证了不同规模下算法的计算性能.

2 预备知识

2.1 基于SVD模型的推荐算法

在众多协同过滤推荐模型中, 基于SVD推荐模型在预测及推荐准确性方面均有优异表现. 该模型将用户及评价物品表示为特征向量并假设用户评分行为取决于用户及相应物品的特征向量的运算结果. 然后利用已有的评分数据求解用户及物品的评分向量, 实现用户对每件物品的评分预测. 此类推荐模型建模过程中, 常将各个用户和物品的特征特征向量按列向量形式横向排列, 记$\boldsymbol{U} \in \mathbb{R}^{f\times u}$和$\boldsymbol{M} \in \mathbb{R}^{f\times m}$分别表示用户以及物品特征矩阵, 其中$u$和$m$为用户和被评价物品的数量, f表示特征向量的维度.

最初的SVD推荐模型[5]假设评分是根据特征向量通过内积计算得到. 采用矩阵形式建模, 记$\boldsymbol{P} \in \mathbb{R}^{u\times m}$表示用户对物品的预测评分, 评分应满足下列表达式:

$$\boldsymbol{P} = \boldsymbol{U}^T\boldsymbol{M}. \tag{1}$$

之后, Paterek等人[6]改进了上述SVD推荐模型并提出BiasSVD模型. 该模型引入偏倚向量有效地反映了用户和物品的内在一致性并提升了评分预测的准确性, 是目前最常用的SVD推荐模型之一. 模型采用$\boldsymbol{\alpha} \in \mathbb{R}^{u\times 1}$, $\boldsymbol{\beta} \in \mathbb{R}^{m\times 1}$, c分别表示用户和物品的偏倚特征向量以及中心化算子, 其中c常取已评分数据的均值, 并认为评分数据不仅取决于用户和被评价物品的特征向量, 还与用户和物品各自的偏倚特征相关. 其预测评分满足如下表达式:

$$\boldsymbol{P} = \boldsymbol{U}^T\boldsymbol{M} + \boldsymbol{\alpha}\mathbf{1}_m{}^T + \mathbf{1}_n\boldsymbol{\beta}^T + c. \tag{2}$$

式中$\mathbf{1}$表示全1列向量. 分析可知, BiasSVD推荐模型能够更好地反映一些与用户或物品相关的个性化特征, 例如一些用户习惯性地给出低的评价或高的评价, 具有较强的应用价值.

然而上述模型仅考虑了基本的用户及物品特征, 忽略了隐式反馈对用户评价行为的影响. 为此, Koren等人[7]提出的SVD++推荐模型考虑了用户评价行为的隐式反馈并引入了特征矩阵$\boldsymbol{Y} \in \mathbb{R}^{f\times m}$表示评价行为背后反映出的用户偏好特征. 基于该思想, SVD++模型假设预测的评分矩阵满足如下表达式:

$$\boldsymbol{P} = \left(\boldsymbol{U} + \boldsymbol{Y}\boldsymbol{J}^T\operatorname{diag}(\boldsymbol{w})\right)^T\boldsymbol{M} + \boldsymbol{\alpha}\mathbf{1}_m{}^T + \mathbf{1}_n\boldsymbol{\beta}^T + c. \tag{3}$$

式3中$\boldsymbol{w} \in \mathbb{R}^{m\times 1}$为系数向量, 其各元素的计算方式为$w_i = \left(\sum_{j=1}^{u}\boldsymbol{J}_{ij}\right)^{-\frac{1}{2}}$, 代表每部电影被评价次数的$-\frac{1}{2}$次方. 相比于式2, 该式在用户特性矩阵$\boldsymbol{U}$的基础上引入用户评分的隐式反馈特征, 更加细致地刻画了用户特征.

进一步地, Guo等人[8]考虑了用户信任的隐式反馈并提出了TrustSVD推荐模型. 该模型以特征矩阵$\boldsymbol{Z} \in \mathbb{R}^{f\times u}$表示各个用户间的信任特征, 结合用户信任关系矩阵$\boldsymbol{T} \in \mathbb{R}^{u\times u}$建立SVD推荐模型如下:

http://engine.scichina.com/doi/10.1360/SSI-2019-0107

$$\boldsymbol{P}=\left(\boldsymbol{U}+\boldsymbol{Y}\boldsymbol{J}^{T}\mathrm{diag}\left(\boldsymbol{w}\right)+\boldsymbol{Z}\boldsymbol{T}^{T}\mathrm{diag}\left(\boldsymbol{\omega}\right)\right)^{T}\boldsymbol{M}+\boldsymbol{\alpha}\mathbf{1}_{m}{}^{T}+\mathbf{1}_{n}\boldsymbol{\beta}^{T}+c. \tag{4}$$

式中$\boldsymbol{\omega}\in\mathbb{R}^{u\times 1}$, 即每个用户与其他用户信任度之和的$-\frac{1}{2}$次方所组成的向量. 由式3和式4可知, SVD++和TrustSVD推荐模型虽然考虑了不同的隐式反馈内容, 但两者具有数学性质上的一致性, 均通过将隐式特征乘上稀疏矩阵的结果作为用户特征的一部分. 此类带有隐式反馈的SVD推荐模型中, SVD++推荐模型最基础也最具代表性, 关于该模型的理论及算法成果可利用数学性质的一致性推广至其他带有隐式反馈的SVD推荐模型. 因此, SVD++推荐模型的理论及高效算法的研究是当前亟待完成的重要工作.

为评估SVD推荐模型的评分预测能力, 人们常常采用均方误差作为衡量指标对比预测评分与已有的用户评分进行评价. 均方误差低表示模型所预测的评分接近于真实评分, 模型的预测能力强, 反之则说明模型预测能力差. 采用矩阵建模, 一般化SVD推荐模型的均方误差表达式如下:

$$\mathrm{mse}=\frac{1}{2}\mathrm{trace}\left(\left(\boldsymbol{J}*\left(\boldsymbol{V}-\boldsymbol{P}\right)\right)^{T}\left(\boldsymbol{J}*\left(\boldsymbol{V}-\boldsymbol{P}\right)\right)\right). \tag{5}$$

式中$*$为Hadamard积运算符[22]; $\boldsymbol{V}\in\mathbb{R}^{u\times m}$为用户评分矩阵, 矩阵中每个元素表示用户对相应物品的评分, 如未评分则记为0; $\boldsymbol{J}\in\{0,1\}^{u\times m}$为指示矩阵, 指示用户是否曾经对相应物品评分, 已评分记为1, 反之为0. 由于实际数据中只有少量用户对物品有评分, $\boldsymbol{V}$,$\boldsymbol{J}$通常为高度稀疏的矩阵. 因此, 如何高效处理稀疏矩阵也是SVD推荐模型研究过程中亟待解决的重要问题之一.

2.2 块坐标下降法

作为一种经典的优化方法, 块坐标下降法(BCD)被广泛应用于诸多优化问题中. 其核心思想是将待优化向量$\boldsymbol{x}$分为p块, 记为$\boldsymbol{x}=[\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_p]$. 然后将关于$\boldsymbol{x}$的目标函数$\mathrm{f}\left(\boldsymbol{x}\right)$被相应地划分为$\mathrm{f}\left(\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_p\right)$并依次求解以第$i$块待优化向量为变量的子问题. 对各个法块均求解下列优化表达式:

$$\min_{\boldsymbol{\xi}_i}\mathrm{f}\left(\boldsymbol{x}_1{}^{(t+1)},\ldots,\boldsymbol{x}_{i-1}{}^{(t+1)},\boldsymbol{\xi}_i,\boldsymbol{x}_{i+1}{}^{(t)},\ldots,\boldsymbol{x}_p{}^{(t)}\right).$$

式中$\boldsymbol{\xi}_i$代替$\boldsymbol{x}_i$作为待优化向量, 其余分块均视为常量. 其算法结构通常表示如下:

算法1 块坐标下降法

1: 初始化$\boldsymbol{x}^{(0)}\leftarrow\left[\boldsymbol{x}_1{}^{(0)},\boldsymbol{x}_2{}^{(0)},\ldots,\boldsymbol{x}_p{}^{(0)}\right]$,$t\leftarrow 0$;
2: **while** 未达到终止条件 **do**
3: **for** each $i\in\{1,2,\ldots,p\}$ **do**
4: $\boldsymbol{x}_i{}^{(t+1)}\leftarrow\arg\min_{\boldsymbol{\xi}_i}\mathrm{f}\left(\boldsymbol{x}_1{}^{(t+1)},\ldots,\boldsymbol{x}_{i-1}{}^{(t+1)},\boldsymbol{\xi}_i,\boldsymbol{x}_{i+1}{}^{(t)},\ldots,\boldsymbol{x}_p{}^{(t)}\right)$;
5: **end for**
6: $t\leftarrow t+1$;
7: **end while**
8: **return** $\boldsymbol{x}^{(t)}$;

相比于其他优化算法, 基于BCD的思想所设计的算法具有速度快, 稳定性强, 实现简单等优点. 当目标函数为分块强凸时, 利用凸函数的性质, BCD算法能够实现$O\left({}^{1}/_{k}\right)$的快速收敛[17]. 尤其是当凸优化问题能够通过解析方法直接求得时, 子问题可直接根据解析式求得最优解, 该过程往往十分高效且不存在理论误差. 由于SVD推荐模型求解问题本质上是一类非负矩阵求解问题且已有文

献[18, 19]关于BCD方法在此类问题的应用做了理论研究, 表明BCD方法对于SVD推荐模型的求解具有较强的理论可行性.

3 SVD推荐模型的梯度求解框架

作为反映目标函数特征的基本要素, 梯度对于许多最优化问题的求解至关重要, SVD推荐模型也不例外. 其求解方法主要分为两类: 一类是根据梯度的极限定义检验每个变量增加极小量后对目标函数的影响, 并据此求得当前的数值梯度; 另一类是通过数学分析方法求得梯度的解析式, 然后代入当前变量值求得数值梯度. 这两类方法在实际应用中都有所涉及. 前者的优势在于能够实现任意目标函数的梯度求解且避免了复杂的数学分析过程. 然而, 该方法无法得知梯度的解析式, 是一种近似的数值求解方法, 存在着求解效率低, 难以获知梯度的内在组成及研究分析等缺陷.

一方面, 实现高效的SVD模型求解算法需要对模型的梯度解析式做必要的数学分析. 另一方面, 带有隐式反馈的SVD推荐模型的复杂性高于传统的SVD推荐模型, 梯度求解困难, 且随着人们研究的深入, 越来越多的隐式反馈特征将被考虑并提出更多的模型, 亟需关于带有隐式反馈的SVD推荐模型梯度求解的理论支持. 因此本章节将基于矩阵建模的思想, 结合矩阵微积分[20~22]的相关理论提出针对任意SVD推荐模型的一般化梯度求解框架, 为梯度求解提供理论支持.

假设用户评分矩阵$\boldsymbol{P}$取决于关于特征参数$\boldsymbol{\xi}$(列向量表示)的函数$\boldsymbol{P}=\mathrm{f}(\boldsymbol{\xi})$. SVD推荐模型的求解采用均方误差表达式5最小化函数$\mathrm{E}(\boldsymbol{\xi})$, 其表达式如下:

$$\mathrm{E}(\boldsymbol{\xi})=\frac{1}{2}\mathrm{trace}\left((\boldsymbol{J}*(\boldsymbol{V}-\boldsymbol{P}))^{T}(\boldsymbol{J}*(\boldsymbol{V}-\boldsymbol{P}))\right)+\mathrm{reg}(\boldsymbol{\xi}).$$

式中$\mathrm{reg}(\boldsymbol{\xi})$为关于特征参数$\boldsymbol{\xi}$的正则化函数, 用于避免训练结果的过拟合问题. 正则化函数可表示为$\mathrm{reg}(\boldsymbol{\xi})=\frac{1}{2}\boldsymbol{\xi}^{T}\mathrm{diag}(\boldsymbol{\kappa})\boldsymbol{\xi}$, $\boldsymbol{\kappa}$表示由各个特征参数依次组成的向量. 显然, 其梯度为$\mathrm{diag}(\boldsymbol{\kappa})\boldsymbol{\xi}$. 对于具体的SVD推荐模型, 常采用$k_{*}$表示各特征矩阵的正则化参数.

$$\frac{\partial\mathrm{E}(\boldsymbol{\xi})}{\partial\boldsymbol{\xi}}=\sum_{i,j}\left(\boldsymbol{e}_{i}{}^{T}(\boldsymbol{J}*(\boldsymbol{P}-\boldsymbol{V}))\boldsymbol{e}_{j}\times\frac{\partial\left(\boldsymbol{e}_{i}{}^{T}\boldsymbol{P}\boldsymbol{e}_{j}\right)}{\partial\boldsymbol{\xi}}\right)+\frac{\partial\mathrm{reg}(\boldsymbol{\xi})}{\partial\boldsymbol{\xi}}.\tag{6}$$

$\mathrm{E}(\boldsymbol{\xi})$的梯度表达式如式6所示. 式中$\boldsymbol{e}_{i}$表示第$i$个元素为1, 其余元素均为0的基本列向量. 该式将$\mathrm{E}(\boldsymbol{\xi})$的梯度求解转化为关于每个预测评分的梯度求解. 目前大多数SVD推荐模型的特征矩阵或向量均是关于$\boldsymbol{P}$的线性矩阵表达式, 针对此类推荐模型表达式, 根据式6可进一步得出以下结论.

定理1 对于待求梯度的特征矩阵$\boldsymbol{X}$, 若$\boldsymbol{P}$是关于$\boldsymbol{X}$的线性矩阵表达式, 即$\boldsymbol{P}=\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}+\cdots$, 则有如下梯度表达式成立:

$$\frac{\partial\mathrm{E}(\boldsymbol{X})}{\partial\boldsymbol{X}}=\boldsymbol{A}^{T}(\boldsymbol{J}*(\boldsymbol{P}-\boldsymbol{V}))\boldsymbol{B}^{T}+\frac{\partial\mathrm{reg}(\boldsymbol{X})}{\partial\boldsymbol{X}}.\tag{7}$$

证明 由于$\boldsymbol{P}=\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}+\cdots$, 因此$\frac{\partial\left(\boldsymbol{e}_{i}{}^{T}\boldsymbol{P}\boldsymbol{e}_{j}\right)}{\partial\boldsymbol{X}}$仅与$\frac{\partial\left(\boldsymbol{e}_{i}{}^{T}\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}\boldsymbol{e}_{j}\right)}{\partial\boldsymbol{X}}$相关, 即:

$$\frac{\partial\left(\boldsymbol{e}_{i}{}^{T}\boldsymbol{P}\boldsymbol{e}_{j}\right)}{\partial\boldsymbol{X}}=\frac{\partial\mathrm{trace}\left(\boldsymbol{e}_{i}{}^{T}\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}\boldsymbol{e}_{j}\right)}{\partial\boldsymbol{X}}=\boldsymbol{A}^{T}\boldsymbol{e}_{i}\boldsymbol{e}_{j}{}^{T}\boldsymbol{B}^{T}.$$

http://engine.scichina.com/doi/10.1360/SSI-2019-0107

将$\frac{\partial\left({e_i}^T \boldsymbol{P} e_j\right)}{\partial \boldsymbol{X}} = \boldsymbol{A}^T e_i {e_j}^T \boldsymbol{B}^T$代入式6可得:

$$\begin{aligned}\frac{\partial \mathrm{E}(\boldsymbol{X})}{\partial \boldsymbol{X}} &= \sum_{i,j}\left({e_i}^T(\boldsymbol{J}*(\boldsymbol{P}-\boldsymbol{V}))e_j \times \boldsymbol{A}^T e_i {e_j}^T \boldsymbol{B}^T\right) + \frac{\partial \operatorname{reg}(\boldsymbol{X})}{\partial \boldsymbol{X}} \\ &= \sum_{i,j}\left(\boldsymbol{A}^T e_i {e_i}^T(\boldsymbol{J}*(\boldsymbol{P}-\boldsymbol{V}))e_j {e_j}^T \boldsymbol{B}^T\right) + \frac{\partial \operatorname{reg}(\boldsymbol{X})}{\partial \boldsymbol{X}} \\ &= \boldsymbol{A}^T\left(\sum_i e_i {e_i}^T\right)(\boldsymbol{J}*(\boldsymbol{P}-\boldsymbol{V}))\left(\sum_j e_j {e_j}^T\right)\boldsymbol{B}^T + \frac{\partial \operatorname{reg}(\boldsymbol{X})}{\partial \boldsymbol{X}} \\ &= \boldsymbol{A}^T(\boldsymbol{J}*(\boldsymbol{P}-\boldsymbol{V}))\boldsymbol{B}^T + \frac{\partial \operatorname{reg}(\boldsymbol{X})}{\partial \boldsymbol{X}}.\end{aligned}$$

定理1得证.

由定理1可知, 求解SVD推荐模型梯度过程仅需根据其矩阵表达式代入$\boldsymbol{A}$和$\boldsymbol{B}$即可. 如BiasSVD推荐模型的矩阵表达式为$\boldsymbol{P} = \boldsymbol{U}^T\boldsymbol{M} + \boldsymbol{\alpha}\mathbf{1}_m{}^T + \mathbf{1}_n\boldsymbol{\beta}^T + c$, 其关于用户特征矩阵$\boldsymbol{U}$的子式为$\boldsymbol{U}^T\boldsymbol{M}$, 由式7可得$\boldsymbol{A} = \boldsymbol{I}, \boldsymbol{B} = \boldsymbol{M}$. 然后代入式7可得:

$$\frac{\partial \mathrm{E}(\boldsymbol{U})}{\partial \boldsymbol{U}^T} = (\boldsymbol{J}*(\boldsymbol{P}-\boldsymbol{V}))\boldsymbol{M}^T + k_u\boldsymbol{U}^T \Rightarrow \frac{\partial \mathrm{E}(\boldsymbol{U})}{\partial \boldsymbol{U}} = \boldsymbol{M}(\boldsymbol{J}*(\boldsymbol{P}-\boldsymbol{V}))^T + k_u\boldsymbol{U}.$$

值得注意的是, 上述表达式直接求出了关于$\boldsymbol{U}^T$的梯度. 关于$\boldsymbol{U}$的梯度表达式还需做一次转置处理才能求得.

4 基于块坐标下降法实现高效的SVD++推荐算法

相比于传统的SVD推荐模型, 考虑了隐式反馈的SVD推荐模型复杂得多. 为提升带隐式反馈的SVD推荐模型的求解性能, 本章节将结合上述成果及块坐标下降法研究高效训练算法的实现方法. 研究过程中, 本文着重解决了稀疏矩阵处理, 容量矩阵求逆等关键问题. 作为考虑了隐式反馈的基本模型, SVD++具有一定的代表性, 针对该模型所设计的块坐标下降算法能够反映其他带有隐式反馈的SVD推荐模型求解时所面临的问题. 因此, 本文谨采用SVD++推荐模型展开高效算法研究, 相关研究只需做适当调整即可拓展至其他带有隐式反馈的SVD推荐模型, 例如TrustSVD, EITrustSVD等.

4.1 SVD++推荐模型的梯度及分块最优解

由式3可知, SVD++推荐模型的矩阵表达式为$\boldsymbol{P}=\left(\boldsymbol{U} + \boldsymbol{Y}\boldsymbol{J}^T\operatorname{diag}(\boldsymbol{w})\right)^T\boldsymbol{M} + \boldsymbol{\alpha}\mathbf{1}_m{}^T + \mathbf{1}_n\boldsymbol{\beta}^T + c$. 利用块坐标下降的思想, 本文将将结合梯度求解框架求解各分块子问题的最优解. 经研究发现, 合理地选择子问题的分块方式可使子问题具有凸性, 符合分块强凸子问题的快速收敛要求[17]且能够求出解析表达式并做进一步理论分析.

首先, 利用定理1可求出SVD++模型下关于各个特征矩阵的梯度, 其表达式如下:

$$\begin{cases} \dfrac{\partial \mathrm{E}(\boldsymbol{U})}{\partial \boldsymbol{U}} = \boldsymbol{M}(\boldsymbol{J} * (\boldsymbol{P}-\boldsymbol{V}))^T + k_u \boldsymbol{U}, \\ \dfrac{\partial \mathrm{E}(\boldsymbol{M})}{\partial \boldsymbol{M}} = \left(\boldsymbol{U} + \boldsymbol{Y}\boldsymbol{J}^T \mathrm{diag}(\boldsymbol{w})\right)(\boldsymbol{J} * (\boldsymbol{P}-\boldsymbol{V})) + k_m \boldsymbol{M}, \\ \dfrac{\partial \mathrm{E}(\boldsymbol{Y})}{\partial \boldsymbol{Y}} = \boldsymbol{M}(\boldsymbol{J} * (\boldsymbol{P}-\boldsymbol{V}))^T \mathrm{diag}(\boldsymbol{w})\,\boldsymbol{J} + k_y \boldsymbol{Y}, \\ \dfrac{\partial \mathrm{E}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = (\boldsymbol{J} * (\boldsymbol{P}-\boldsymbol{V}))\,\mathbf{1}_m + k_a \boldsymbol{\alpha}, \\ \dfrac{\partial \mathrm{E}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = (\boldsymbol{J} * (\boldsymbol{P}-\boldsymbol{V}))^T \mathbf{1}_n + k_b \boldsymbol{\beta}. \end{cases}$$

为求得各个子问题的最优解, 可尝试令上述每个梯度均为$\boldsymbol{o}$, 然后通过代数运算求得各个分块子问题的最优解. 关于$\boldsymbol{\alpha}$及$\boldsymbol{\beta}$可求得如下最优解表达式:

$$\frac{\partial \mathrm{E}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \boldsymbol{o} \Rightarrow \boldsymbol{\alpha} = \left(\boldsymbol{J} * \left(\boldsymbol{V} - \left(\boldsymbol{U} + \boldsymbol{Y}\boldsymbol{J}^T \mathrm{diag}(\boldsymbol{w})\right)^T \boldsymbol{M} - \mathbf{1}_n \boldsymbol{\beta}^T - c\right)\right)\mathbf{1}_m \oslash (\boldsymbol{J}\mathbf{1}_m + k_a). \tag{8}$$

$$\frac{\partial \mathrm{E}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \boldsymbol{o} \Rightarrow \boldsymbol{\beta} = \left(\boldsymbol{J} * \left(\boldsymbol{V} - \left(\boldsymbol{U} + \boldsymbol{Y}\boldsymbol{J}^T \mathrm{diag}(\boldsymbol{w})\right)^T \boldsymbol{M} - \boldsymbol{\alpha}\mathbf{1}_m{}^T - c\right)\right)^T \mathbf{1}_n \oslash \left(\mathbf{1}_n{}^T \boldsymbol{J} + k_b\right). \tag{9}$$

式中$\oslash$表示同型向量(矩阵)间的对位除法, 利用该运算符能够高效地实现上述表达式的求解. 然而关于特征矩阵$\boldsymbol{U}$, $\boldsymbol{M}$, $\boldsymbol{Y}$的最优解则较为复杂, 直接令其梯度为$\boldsymbol{O}$难以求得最优解. 因此本文按列将其继续划分为多个子块, 求得如下关于特征向量的梯度. 为表示方便, 令$\boldsymbol{W} = \boldsymbol{V} - \boldsymbol{\alpha}\mathbf{1}_m{}^T - \mathbf{1}_n \boldsymbol{\beta}^T - c$, $\boldsymbol{U}' = \boldsymbol{U} + \boldsymbol{Y}\boldsymbol{J}^T \mathrm{diag}(\boldsymbol{w})$.

$$\begin{cases} \dfrac{\partial \mathrm{E}(\boldsymbol{u}_k)}{\partial \boldsymbol{u}_k} = \left(\boldsymbol{M}\mathrm{diag}\left(\boldsymbol{e}_k{}^T \boldsymbol{J}\right)\boldsymbol{M}^T + k_u \boldsymbol{I}\right)\boldsymbol{u}_k - \boldsymbol{M}\left(\boldsymbol{J} * \left(\boldsymbol{W} - \mathrm{diag}(\boldsymbol{w})\,\boldsymbol{J}\boldsymbol{Y}^T \boldsymbol{M}\right)\right)^T \boldsymbol{e}_k, \\ \dfrac{\partial \mathrm{E}(\boldsymbol{m}_k)}{\partial \boldsymbol{m}_k} = \left(\boldsymbol{U}'\mathrm{diag}(\boldsymbol{J}\boldsymbol{e}_k)\,\boldsymbol{U}'^T + k_m \boldsymbol{I}\right)\boldsymbol{m}_k - \boldsymbol{U}'(\boldsymbol{J} * \boldsymbol{V}')\,\boldsymbol{e}_k, \\ \dfrac{\partial \mathrm{E}(\boldsymbol{y}_k)}{\partial \boldsymbol{y}_k} = \boldsymbol{M}(\boldsymbol{J} * (\boldsymbol{P}-\boldsymbol{V}))^T \mathrm{diag}(\boldsymbol{w})\,\boldsymbol{J}\boldsymbol{e}_k + k_y \boldsymbol{y}_k. \end{cases}$$

上述式中, 本文谨对$\boldsymbol{u}_k$及$\boldsymbol{m}_k$做了适当的移项处理, 将梯度表达式转换为$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$的形式. 然后令其梯度等于$\boldsymbol{o}$并求得最优解表达式如下:

$$\boldsymbol{u}_k = \left(\boldsymbol{M}\mathrm{diag}\left(\boldsymbol{e}_k{}^T \boldsymbol{J}\right)\boldsymbol{M}^T + k_u \boldsymbol{I}\right)^{-1} \boldsymbol{M}\left(\boldsymbol{J} * \left(\boldsymbol{W} - \mathrm{diag}(\boldsymbol{w})\,\boldsymbol{J}\boldsymbol{Y}^T \boldsymbol{M}\right)\right)^T \boldsymbol{e}_k. \tag{10}$$

$$\boldsymbol{m}_k = \left(\boldsymbol{U}'\mathrm{diag}(\boldsymbol{J}\boldsymbol{e}_k)\,\boldsymbol{U}'^T + k_m \boldsymbol{I}\right)^{-1} \boldsymbol{U}'(\boldsymbol{J} * \boldsymbol{V}')\,\boldsymbol{e}_k. \tag{11}$$

考虑了用户的评分的隐式反馈, SVD++模型的矩阵表达式中特征矩阵$\boldsymbol{Y}$的涉及了对隐式反馈的关联运算. 模型要求每个用户根据其评价的每部电影求得关于评价行为特征向量. 该运算极大提升分块子问题的复杂性, 无法直接采用类似$\boldsymbol{U}$, $\boldsymbol{M}$的处理方式求$\boldsymbol{Y}$的最优解. 为求$\boldsymbol{y}_k$的最优解表达式,

http://engine.scichina.com/doi/10.1360/SSI-2019-0107

本文通过提取$\frac{\partial \mathrm{E}(\boldsymbol{y}_k)}{\partial \boldsymbol{y}_k}$中与$\boldsymbol{y}_k$相关的项式得到如下表达式:

$$\begin{aligned}
&\frac{\partial \mathrm{E}\left(\boldsymbol{y}_k\right)}{\partial \boldsymbol{y}_k}=\boldsymbol{M}(\boldsymbol{J}*(\boldsymbol{P}-\boldsymbol{V}))^T\mathrm{diag}\left(\boldsymbol{w}\right)\boldsymbol{J}\boldsymbol{e}_k+k_y\boldsymbol{y}_k\\
=&\boldsymbol{M}\left(\boldsymbol{J}^T*\left(\boldsymbol{M}^T\boldsymbol{Y}\boldsymbol{J}^T\mathrm{diag}\left(\boldsymbol{w}\right)\right)\right)\mathrm{diag}\left(\boldsymbol{w}\right)\boldsymbol{J}\boldsymbol{e}_k+k_y\boldsymbol{y}_k \qquad ①\\
&+\boldsymbol{M}(\boldsymbol{J}*(\boldsymbol{P}-\boldsymbol{V}))^T\mathrm{diag}\left(\boldsymbol{w}\right)\boldsymbol{J}\boldsymbol{e}_k-\boldsymbol{M}\left(\boldsymbol{J}^T*\left(\boldsymbol{M}^T\boldsymbol{Y}\boldsymbol{J}^T\mathrm{diag}\left(\boldsymbol{w}\right)\right)\right)\mathrm{diag}\left(\boldsymbol{w}\right)\boldsymbol{J}\boldsymbol{e}_k \qquad ②\\
=&\left(\boldsymbol{M}\mathrm{diag}\left(\boldsymbol{Q}\boldsymbol{e}_k\right)\boldsymbol{M}^T+k_y\boldsymbol{I}\right)\boldsymbol{y}_k+\left(\boldsymbol{M}(\boldsymbol{J}*(\boldsymbol{P}-\boldsymbol{V}))^T\mathrm{diag}\left(\boldsymbol{w}\right)\boldsymbol{J}-\boldsymbol{M}\left(\boldsymbol{Q}*\left(\boldsymbol{M}^T\boldsymbol{Y}\right)\right)\right)\boldsymbol{e}_k. \qquad ③
\end{aligned} \tag{12}$$

由推导过程可知, 式12中的$\boldsymbol{M}(\boldsymbol{J}*(\boldsymbol{P}-\boldsymbol{V}))^T\mathrm{diag}\left(\boldsymbol{w}\right)\boldsymbol{J}\boldsymbol{e}_k$通过分离其中与$\boldsymbol{Y}$相关的子部分获得子式①, 然后再减去该子式获得子式②. 所得子式②的两项式中$\boldsymbol{Y}$的部分相互抵消, 因此实际上子式②是与$\boldsymbol{Y}$无关的常数项. 进一步对子式①化简可提取关于$\boldsymbol{y}_k$线性表达式, 最终化简结果如式12部分③所示. 式中$\boldsymbol{Q}=\boldsymbol{J}^T\mathrm{diag}\left(\boldsymbol{w}*\boldsymbol{w}\right)\boldsymbol{J}$是关于$\boldsymbol{J}$的二次型矩阵. 利用该结果, 通过移项可得关于$\boldsymbol{y}_k$的最优解表达式:

$$\boldsymbol{y}_k=\left(\boldsymbol{M}\mathrm{diag}\left(\boldsymbol{Q}\boldsymbol{e}_k\right)\boldsymbol{M}^T+k_y\boldsymbol{I}\right)^{-1}\left(\boldsymbol{M}\left(\boldsymbol{Q}*\left(\boldsymbol{M}^T\boldsymbol{Y}\right)\right)-\boldsymbol{M}(\boldsymbol{J}*(\boldsymbol{P}-\boldsymbol{V}))^T\mathrm{diag}\left(\boldsymbol{w}\right)\boldsymbol{J}\right)\boldsymbol{e}_k. \tag{13}$$

相比于$\boldsymbol{u}_k$和$\boldsymbol{m}_k$, $\boldsymbol{y}_k$的最优解表达式更加复杂且计算量更大, 需进一步研究有效的计算方法. 作为重要的推荐特征, 评分特征矩阵$\boldsymbol{Y}$的求解过程具有一定的代表性, 其求解方法可拓展至其他带有隐式反馈的推荐模型. 设计求解算法之前, 文本将首先关注以下几个关键问题.

4.2 容量矩阵求逆问题的分析及优化

最优解表达式10,11,13中逆矩阵运算部分实际为容量矩阵[24]求逆问题. 式中的容量矩阵求逆运算问题形式上相当于求解表达式: $\left(\lambda\boldsymbol{I}+\boldsymbol{X}\boldsymbol{X}^T\right)^{-1}$.

引理1 [23] 当$\lambda>0$, 矩阵$\lambda\boldsymbol{I}+\boldsymbol{X}\boldsymbol{X}^T$为正定矩阵.

根据引理1可知式中$\lambda\boldsymbol{I}+\boldsymbol{X}\boldsymbol{X}^T$可逆. 常规计算要求该式求解需先进行乘法运算然后再求逆. 然而, 该做法并未充分发挥该式的计算潜能. 为此, 本文基于Sherman-Morrison公式[23]提出了定理2并据此提出了更高效的迭代求逆算法.

定理2 当$\lambda>0$, 对于矩阵$\lambda\boldsymbol{I}+\boldsymbol{X}\boldsymbol{X}^T$, 若$\boldsymbol{X}$的列数为$c$, $\boldsymbol{x}_k$和$\boldsymbol{X}_k$分别表示$\boldsymbol{X}$的第k列向量以及由$\boldsymbol{X}$的前k列组成的矩阵且$\boldsymbol{R}_k=\left(\lambda\boldsymbol{I}+\boldsymbol{X}_k{\boldsymbol{X}_k}^T\right)^{-1}$. 则有以下递推关系成立:

$$\boldsymbol{R}_{k+1}=\boldsymbol{R}_k-\frac{\boldsymbol{R}_k\boldsymbol{x}_{k+1}{\boldsymbol{x}_{k+1}}^T\boldsymbol{R}_k}{1+{\boldsymbol{x}_{k+1}}^T\boldsymbol{R}_k\boldsymbol{x}_{k+1}},0\leqslant k<c. \tag{14}$$

证明 根据Sherman-Morrison公式可知: 当$\boldsymbol{A}$和$\boldsymbol{A}+\boldsymbol{u}\boldsymbol{v}^T$可逆时, 有等式$\left(\boldsymbol{A}+\boldsymbol{u}\boldsymbol{v}^T\right)^{-1}=\boldsymbol{A}^{-1}-\frac{\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}}{1+\boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}}$成立, 其中$\boldsymbol{u}$, $\boldsymbol{v}$为列向量.

根据矩阵乘法运算原理有: $\lambda\boldsymbol{I}+\boldsymbol{X}\boldsymbol{X}^T=\lambda\boldsymbol{I}+\sum_{i=1}^{c}\boldsymbol{x}_i{\boldsymbol{x}_i}^T$. 即$\boldsymbol{R}_k=\left(\lambda\boldsymbol{I}+\sum_{i=1}^{k}\boldsymbol{x}_i{\boldsymbol{x}_i}^T\right)^{-1}$.

由$\boldsymbol{R}_k$求得$\boldsymbol{R}_{k+1}=\left({\boldsymbol{R}_k}^{-1}+\boldsymbol{x}_{k+1}{\boldsymbol{x}_{k+1}}^T\right)^{-1}$.

运用Sherman-Morrison公式, 并将${\boldsymbol{R}_k}^{-1}$代入式中的$\boldsymbol{A}$, 将$\boldsymbol{x}_{k+1}$代入$\boldsymbol{u}$, $\boldsymbol{v}$可得: $\boldsymbol{R}_{k+1}=\boldsymbol{R}_k-\frac{\boldsymbol{R}_k\boldsymbol{x}_{k+1}{\boldsymbol{x}_{k+1}}^T\boldsymbol{R}_k}{1+{\boldsymbol{x}_{k+1}}^T\boldsymbol{R}_k\boldsymbol{x}_{k+1}}$

定理2得证.

显然, $\boldsymbol{R}_c$即$\left(\lambda \boldsymbol{I}+\boldsymbol{X} \boldsymbol{X}^T\right)^{-1}$. 其快速迭代求逆的算法过程如下:

算法2 容量矩阵快速迭代求逆算法

输入: $\lambda\,(\lambda>0)$,$\boldsymbol{X}$;

输出: $\boldsymbol{R}_c:\left(\lambda \boldsymbol{I}+\boldsymbol{X} \boldsymbol{X}^T\right)^{-1}$;

初始化$\boldsymbol{R}_0=\lambda^{-1} \boldsymbol{I}$, $\boldsymbol{X}$划分为列向量组$[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_c]$;

for each $i \in\{1,2, \ldots, c\}$ **do**

  令$\boldsymbol{t}=\boldsymbol{R}_{i-1} \boldsymbol{x}_i$;

  $\boldsymbol{R}_i=\boldsymbol{R}_{i-1}-\frac{\boldsymbol{t} \boldsymbol{t}^T}{1+\boldsymbol{x}_i{}^T \boldsymbol{t}}$; //根据式14实现

end for

return $\boldsymbol{R}_c$;

相比于传统算法, 算法2的实现过程简洁高效且时间复杂度仅为$O\left(f^2 c\right)$. 相比而言, 一般方法的时间复杂度为$O\left(f^2 c+f^3\right)$. 因此该算法具有更高的求解效率, 尤其是当模型定义的特征数量较多时该算法能更显著地提升求解效率.

在特征维度f一定的情况下, 算法2所消耗的时间取决于列数c. 应用于式10,11,13, 算法2完成一次$\boldsymbol{U}$, $\boldsymbol{M}$求解需要处理的列数等于$\boldsymbol{J}$中非零元个数. 由于$\boldsymbol{J}$通常为高度稀疏的矩阵, 实际上该算法完成一次求解所需处理的数据量并不多, 能够很快地完成求解任务. 同理, 完成$\boldsymbol{Y}$的求解需处理列数等于$\boldsymbol{Q}$中非零元个数. 然而, 作为$\boldsymbol{J}$的二次型矩阵, $\boldsymbol{Q}$并不稀疏, 而是大小为$m \times m$大型稠密矩阵. 以数据集MovieLens 10M为例, 由表1可知其关于$\boldsymbol{J}$的数据密度仅为1.34%, 而$\boldsymbol{Q}$的数据密度却高达85.6%. 求解$\boldsymbol{Y}$复杂度实际接近于$O\left(f^2 m^2\right)$, 因此该部分运算耗时将成为算法效率的瓶颈, 仍需进一步寻找优化策略.

表 1 数据集稀疏程度分析表

Table 1 Sparsity Analysis Table of Data Sets

Data sets	Data density of $\boldsymbol{J}$	Data density of $\boldsymbol{Q}$	RSD	Data covered up to 60%		Data covered up to 80%	
				topK	Proportion of topK	topK	Proportion of topK
Movielens 100K	6.30%	69.6%	116%	240	14.3%	447	26.6%
Movielens 1M	4.47%	82.4%	132%	555	15.0%	1064	28.7%
Movielens 10M	1.34%	85.6%	205%	981	9.19%	2130	19.9%
Movielens 20M	0.54%	40.1%	178%	1147	4.29%	2511	9.39%
Movielens Last	0.18%	31.6%	217%	1342	2.50%	3233	6.00%

注意到$\boldsymbol{Q}$实际上是根据用户评分行为计算的物品相似度矩阵, 物品间的相似度取决于它们被共同用户评分的数量. 对于式13中$\boldsymbol{M}\,\mathrm{diag}\left(\boldsymbol{Q} \boldsymbol{e}_k\right) \boldsymbol{M}^T+k_y \boldsymbol{I}$部分, 与物品$k$相似度越高, 相应物品的特征向量对结果影响越大, 反之, 相似度低的物品影响极小, 甚至可以忽略不计. 研究表明, 实际数据中物品间相似度存在巨大差异且各物品均只与少量物品具有显著的相似性. 表1采用相对标准偏差(RSD)度量了Movielens数据集中每部电影相对于其他电影的相似性差异, 可知各数据集的RSD均高于100%甚至Movielens 10M与Movielens Last高达205%(即相似度的标准差是平均相似度的2.05倍) 及217%. 表明数据中各电影与其他电影的相似度存在严重不平衡, 与各电影最相近的k个物品的特征将成为决定$\boldsymbol{M}\,\mathrm{diag}\left(\boldsymbol{Q} \boldsymbol{e}_k\right) \boldsymbol{M}^T$ 计算结果的主要因素, 其他不相似的物品对于计算结果的影响有限但却需要与相似物品同等的计算资源.

http://engine.scichina.com/doi/10.1360/SSI-2019-0107

根据上述分析, 本文提出了一种有效提升式13中关于容量矩阵求逆计算效率的方案. 其思想是每次计算容量矩阵仅选取与物品k相似度最高的topK件物品的特征向量进行计算而将其他物品的特征向量忽略. 由表1可知, 只需选取少量相似物品即可涵盖60%及80%的数据. 其中, Movielens 10M只需取前9%的相似物品就能涵盖60%的数据且涵盖80%的数据仅需要用前20%的相似物品, 显著符合“二八法则”. 虽然该方案在一定程度上影响了$\boldsymbol{y}_k$求解精度, 不过却能大幅提升算法效率. 因此, 采用该方案实现算法效率的提升具有较强的合理性.

4.3 稀疏数据的优化处理

作为一类数据高度稀疏的算法, SVD推荐模型求解算法的稀疏数据处理能力至关重要. 根据表1可知, Movielens系列数据集仅为0.18%-6.3%的数据密度, 数据稀疏性是算法的重要特征. 采用矩阵建模处理稀疏数据的优势在于拥有诸多文献[25, 26]针对稀疏矩阵的存储和计算提供理论支持, 并且已有许多编程语言(如Python, Matlab等)都稀疏矩阵提供了专业支持. 然而这些稀疏矩阵的研究成果面向了一般性的矩阵运算问题, 对于本文涉及的具体问题仍需结合已有成果进行合理的代数优化才能实现稀疏数据的高效处理.

上述最优解表达式8-11,13均涉及了形如$\boldsymbol{J}*\left(\boldsymbol{U}^T\boldsymbol{M}\right)$的Hadamard积运算子式. 值得注意的是, 该式首先计算的$\boldsymbol{U}^T\boldsymbol{M}$是一个$u\times m$的大型稠密矩阵, 然后将其与$\boldsymbol{J}$进行Hadamard积运算实际上是执行了大量元素的置零操作, 整体过程意味大量不必要的运算. 为避免不必要的运算, 式15通过代数方法将其转换为如下计算形式. 为体现稀疏数据的处理, 式中带下划线矩阵表示采用了稀疏方式存储和计算的矩阵, 如采用列压缩存储(CCS)格式存储稀疏矩阵.

$$\underline{\boldsymbol{J}}*\left(\boldsymbol{U}^T\boldsymbol{M}\right)=\sum_{i=1}^{f}\underline{diag\left(\underline{\boldsymbol{e}_i}^T\boldsymbol{U}\right)}\,\underline{\boldsymbol{J}}\,\underline{diag\left(\underline{\boldsymbol{e}_i}^T\boldsymbol{M}\right)}. \tag{15}$$

经分析, 式15左右两边的时间复杂度分别为$O\left(umf\right)$和$O\left(\left(u+m+\eta\right)f\right)$, 其中$\eta$为$\boldsymbol{J}$中非零元素个数. 由于$\boldsymbol{J}$的稀疏度极高, 故$\eta\ll um$, 因此右式的计算量远低于左式. 为避免赘述, 下文的算法重点在于描述各步骤实现的内容, 涉及到与左式相关计算并不按右式展开, 但这并不意味着实际也左式方式计算.

然而关于$\boldsymbol{Y}$的求解过程中所面临的问题更加复杂. 由于$\boldsymbol{Q}$是大型稠密矩阵, 式13中关于子式$\boldsymbol{Q}*\left(\boldsymbol{M}^T\boldsymbol{Y}\right)$的部分应用式15所述的代数方法仍面临大量稠密数据的运算. 为避免关于大型稠密矩阵的运算, 文本将结合式中与之左乘的矩阵$\boldsymbol{M}$实现稀疏优化, 具体优化过程如式16所示, 式中仍采用下划线标识稀疏存储的矩阵.

$$\boldsymbol{M}\left(\boldsymbol{Q}*\left(\boldsymbol{M}^T\boldsymbol{Y}\right)\right)=\sum_{i=1}^{f}\boldsymbol{M}\underline{\mathrm{diag}\left(\boldsymbol{m}_i\right)}\,\underline{\boldsymbol{J}}^T\,\underline{\mathrm{diag}\left(\boldsymbol{w}*\boldsymbol{w}\right)}\,\underline{\boldsymbol{J}}\,\underline{\mathrm{diag}\left(\boldsymbol{y}_i\right)}. \tag{16}$$

分析可知, 直接计算$\boldsymbol{M}\left(\boldsymbol{Q}*\left(\boldsymbol{M}^T\boldsymbol{Y}\right)\right)$所需的时空复杂度分别为$O\left(m^2f\right)$和$O\left(m^2+mf\right)$, 当数据涉及了大量物品时, 关于该式的计算将面临巨大的挑战. 式16将$\boldsymbol{Q}$拆解为关于$\boldsymbol{J}$的表达式并做上述变换后, 其时空复杂度变为$O\left(f^2\left(\eta+m+u\right)\right)$和$O\left(mf+uf\right)$. 相比于左式, 右式在时间效率方面上能更有效地处理更多物品, 适合于大规模稀疏数据. 不过时间效率特征维度呈二次相关, 采用右式时特征维度不宜设置过高. 空间效率方面, 右式仅需要$O\left(mf+uf\right)$的额外空间, 远低于左式, 能够极大地节省空间方面的消耗. 综上所述, 右式提供了一种更加快速有效的稀疏数据处理方案.

4.4 块梯度下降法的设计与实现

经前文分析, 本文求解了各个分块子问题最优解表达式并研究了其中关于容量矩阵求逆及稀疏数据优化处理等关键问题. 基于上述成果, 本小节将设计基于块梯度下降的SVD++推荐模型求解算法BCDSVD++(Block Coordinate Descent for SVD++), 其伪代码如算法3所示. 经研究, 算法训练过程中评分预测误差并非随着训练迭代次数的增加而减小, 当迭代到一定程度后误差反而增加, 表明此时该算法已出现过拟合问题. 为有效处理过拟合问题, 算法将数据集分为训练集和验证集. 记$\boldsymbol{V}'$与$\boldsymbol{J}'$分别表示验证集的评分矩阵与指示矩阵. 每次迭代后, 算法均利用验证集检验当前模型的误差, 直到所验证的误差出现上升时算法停止.

算法3 基块坐标下降法的SVD++推荐模型求解算法BCDSVD++

输入: 评分矩阵$\boldsymbol{V}$, $\boldsymbol{V}'$, 指示矩阵$\boldsymbol{J}$, $\boldsymbol{J}'$, 最大迭代次数iter, topK, 正则化参数k_u,k_m,k_a,k_b,k_y;

输出: $\boldsymbol{U}^{(t)}$,$\boldsymbol{M}^{(t)}$,$\boldsymbol{Y}^{(t)}$,$\boldsymbol{\alpha}^{(t)}$,$\boldsymbol{\beta}^{(t)}$;

// 以下是算法预处理及初始化过程, 常量数据应尽量在该过程完成计算.

随机初始化特征矩阵$\boldsymbol{U}^{(0)}$, $\boldsymbol{M}^{(0)}$, $\boldsymbol{\alpha}^{(0)}$, $\boldsymbol{\beta}^{(0)}$使其中元素符合$\mathcal{N}(0,1)$, 并令$\boldsymbol{Y}^{(0)}=\boldsymbol{O}$;

初始化$c=\frac{{\mathbf{1}_u}^T(\boldsymbol{J}*\boldsymbol{V})\mathbf{1}_m}{{\mathbf{1}_u}^T\boldsymbol{J}\mathbf{1}_m}$, $t=0$, $\boldsymbol{v}_1=\boldsymbol{J}\mathbf{1}_m+k_a$, $\boldsymbol{v}_2=\boldsymbol{J}^T\mathbf{1}_n+k_b$, $\text{mse}^{(0)}=+\infty$;

计算$\boldsymbol{w}$令$w_i=\left(\sum_{k=1}^{u}j_{ik}\right)^{-\frac{1}{2}}$, 令$\boldsymbol{J}_w=\text{diag}(\boldsymbol{w})\,\boldsymbol{J}$并求$\boldsymbol{Q}=\boldsymbol{J}_w^T\boldsymbol{J}_w$;

根据$\boldsymbol{Q}$求得$\boldsymbol{Q}^*$, 并保留每列topK个最大值, 其余置零. 然后$\boldsymbol{Q}^*$各列乘上相应系数保持每列的和与$\boldsymbol{Q}$的对应列相同; //参见4.2节关于求逆问题优化的讨论

while $t<\text{iter}$ **do**

  // 根据公式10求解$\boldsymbol{U}^{(t+1)}$, 矩阵$\boldsymbol{B}$表示求解各特征向量过程中由常量向量组成的临时矩阵.

  令$\boldsymbol{B}=\boldsymbol{M}^{(t)}\left(\boldsymbol{J}*\left(\boldsymbol{V}-\boldsymbol{J}_w\boldsymbol{Y}^T\boldsymbol{M}^{(t)}-\boldsymbol{\alpha}^{(t)}{\mathbf{1}_m}^T-\mathbf{1}_n{\boldsymbol{\beta}^{(t)}}^T-c\right)\right)^T$;

  for each $k\in\{1,2,\ldots,u\}$ **do**

    从$\boldsymbol{M}^{(t)}$中选择满足$j_{ki}=1, 1\leqslant i\leqslant m$的列组成矩阵$\boldsymbol{M}^*$;

    // 此处及下文的解线性方程过程可结合定理2与算法2实现

    求解关于$\boldsymbol{u}_k^{(t+1)}$的解线性方程: $\left(\boldsymbol{M}^*{\boldsymbol{M}^*}^T+k_u\boldsymbol{I}\right)\boldsymbol{u}_k^{(t+1)}=\boldsymbol{b}_k$; //$\boldsymbol{b}_k$表示$\boldsymbol{B}$的第$k$列

  end for

  组合$\boldsymbol{u}_k^{(t+1)}$使得$\boldsymbol{U}^{(t+1)}=\left[\boldsymbol{u}_1^{(t+1)},\boldsymbol{u}_2^{(t+1)},\ldots,\boldsymbol{u}_u^{(t+1)}\right]$;

  // 根据公式11求解$\boldsymbol{M}^{(t+1)}$

  求$\boldsymbol{U}'^{(t+1)}$=$\boldsymbol{U}^{(t+1)}+\boldsymbol{Y}^{(t)}\boldsymbol{J}_w^T$, 令$\boldsymbol{B}=\boldsymbol{U}'^{(t+1)}\left(\boldsymbol{J}*\left(\boldsymbol{V}-\boldsymbol{\alpha}^{(t)}{\mathbf{1}_m}^T-\mathbf{1}_n{\boldsymbol{\beta}^{(t)}}^T-c\right)\right)$;

  for each $k\in\{1,2,\ldots,m\}$ **do**

    从$\boldsymbol{U}'^{(t+1)}$中选择满足$j_{ik}=1, 1\leqslant i\leqslant u$的列组成矩阵$\boldsymbol{U}^*$;

    求解关于$\boldsymbol{m}_k^{(t+1)}$的线性方程: $\left(\boldsymbol{U}^*{\boldsymbol{U}^*}^T+k_m\boldsymbol{I}\right)\boldsymbol{m}_k^{(t+1)}=\boldsymbol{b}_k$; //$\boldsymbol{b}_k$表示$\boldsymbol{B}$的第$k$列

  end for

  // 根据公式8,9求解$\boldsymbol{\alpha}$和$\boldsymbol{\beta}$

  组合$\boldsymbol{m}_k^{(t+1)}$可得$\boldsymbol{M}^{(t+1)}=\left[\boldsymbol{m}_1^{(t+1)},\ldots,\boldsymbol{m}_m^{(t+1)}\right]$, 并令$\boldsymbol{B}=\boldsymbol{J}*\left(\boldsymbol{V}-{\boldsymbol{U}'^{(t+1)}}^T\boldsymbol{M}^{(t+1)}-c\right)$;

  $\boldsymbol{\alpha}^{(t+1)}\leftarrow\left(\left(\boldsymbol{B}-\boldsymbol{J}*\left(\mathbf{1}_n{\boldsymbol{\beta}^{(t)}}^T\right)\right)\mathbf{1}_m\right)\oslash\boldsymbol{v}_1$, $\boldsymbol{\beta}^{(t+1)}\leftarrow\left(\left(\boldsymbol{B}-\boldsymbol{J}*\left(\boldsymbol{\alpha}^{(t+1)}{\mathbf{1}_m}^T\right)\right)^T\mathbf{1}_n\right)\oslash\boldsymbol{v}_2$;

  // 根据公式13求解$\boldsymbol{Y}^{(t+1)}$, 可结合稀疏优化表达式15,16实现.

$\boldsymbol{B} \leftarrow \boldsymbol{M}^{(t+1)}\left(\boldsymbol{Q} * \left(\boldsymbol{M}^{(t+1)^T}\boldsymbol{Y}^{(t)}\right)\right) + \boldsymbol{M}^{(t+1)}\left(\boldsymbol{B} - \boldsymbol{J} * \left(\boldsymbol{\alpha}^{(t+1)}\mathbf{1}_m{}^T + \mathbf{1}_n\boldsymbol{\beta}^{(t+1)^T}\right)\right)^T\boldsymbol{J}_w$;
for each $k \in \{1, 2, \ldots, m\}$ **do**
  令$\boldsymbol{M}^* = \boldsymbol{M}^{(t+1)}\,\mathrm{diag}\,(\boldsymbol{q}_k^*)$, 其中$\boldsymbol{q}_k^*$为$\boldsymbol{Q}^*$的第$k$列;
  求解关于$\boldsymbol{y}_k^{(t+1)}$的线性方程: $\left(\boldsymbol{M}^*\boldsymbol{M}^{*T} + k_y\boldsymbol{I}\right)\boldsymbol{y}_k^{(t+1)} = \boldsymbol{b}_k$;//$\boldsymbol{b}_k$表示$\boldsymbol{B}$的第$k$列
end for
// 该计算步骤需代入验证集相关矩阵$\boldsymbol{V}'$和$\boldsymbol{J}'$求均方误差.
利用$\boldsymbol{U}^{(t+1)}$, $\boldsymbol{M}^{(t+1)}$, $\boldsymbol{Y}^{(t+1)}$, $\boldsymbol{\alpha}^{(t+1)}$, $\boldsymbol{\beta}^{(t+1)}$求得验证集均方误差$\mathrm{mse}^{(t+1)}$;
// 判断是否出现过拟合现象
当$\mathrm{mse}^{(t+1)} > \mathrm{mse}^{(t)}$时, 迭代终止, 退出循环; 否则, $t \leftarrow t+1$, 进入下一次循环;
end while
return $\boldsymbol{U}^{(t)}$,$\boldsymbol{M}^{(t)}$,$\boldsymbol{Y}^{(t)}$,$\boldsymbol{\alpha}^{(t)}$,$\boldsymbol{\beta}^{(t)}$;

该算法通过矩阵形式描述了各个步骤的实现内容, 结构简洁紧凑, 并且在设计过程中充分考虑算法计算效率采用了较高效的实现方案. 值得注意的是, 伪代码重点在于描述各步骤的实现内容, 步骤内部的实现未必完全根据其表达式计算而可能采用更高效的计算方案. 例如, 一些稀疏优化的处理细节并未体现于算法中. 根据上述代码所实现的可执行源码提供于GitHub网站https://github.com/imcjp/BCDSvdppExp, 读者可自行下载.

5 算法评估

为有效评估算法优劣, 本文通过实验检验算法3的性能. 实验除了评估算法的准确性, 稳定性等因素外还考虑了算法达到稳定状态所消耗的时间. 算法性能高低决定着算法处理数据的效率, 高效的算法意味着使用更少的计算资源且能够处理更大规模的数据.

5.1 实验背景

作为对比实验, 本文谨选择了TensorFlow框架下基于Adam优化算法[16]实现的SVD++推荐模型进行比较. 其原因在于TensorFlow下的Adam算法是基于梯度下降的思想改进的高效求解算法且是当前主流的高性能并行优化算法. 实验环境采用Intel Core i5-8600K CPU @ 3.60Hz, 16GB RAM, Nvidia GeForce GTX 1060 6GB. 数据集采用MovieLens系列数据集, 该数据集是一个历史悠久的经典电影评分数据集且数据规模从10万到2000万均有覆盖, 能够有效验证不同数据规模下的性能, 该数据集提供于https://grouplens.org/datasets/movielens. 其中, 最新数据集Movielens Last于2018年9月更新, 包含2700万条评分数据并涉及28万个评分用户以及5万部电影. 对于SVD++推荐模型而言, 该数据集是一个巨大的挑战. 此外, 本文还引入了FilmTrust数据集用于进一步验证上述理论成果. 一方面, 将该数据集应用于算法3, 表明其能够适应于各类不同的数据集; 另一方面, 基于上述梯度求解框架的理论, 本文将该数据应用于针对TrustSVD模型拓展设计BCDTrustSVD, 充分验证了该理论的一般适用性. BCDTrustSVD可通过对式4以及算法3的相关理论成果进行适当拓展实现, 因此本文将不再对其实现细节做更多赘述. 具体的算法实现及实验结果可参见上述本文提供的源码附件.

5.2 实验结果

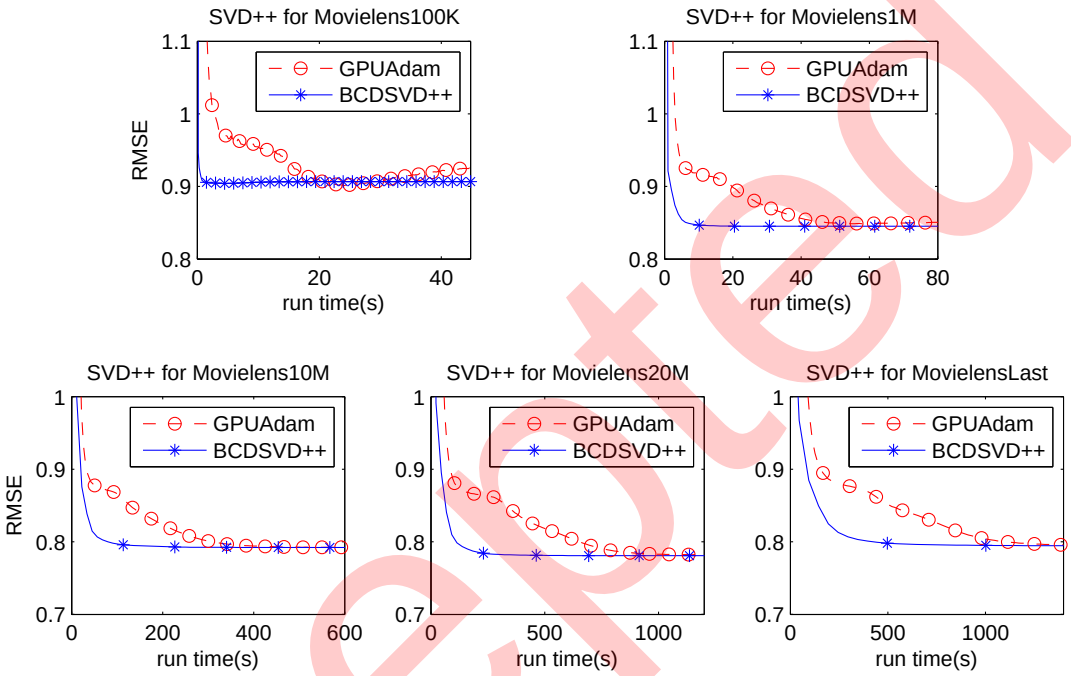


图 1 SVD++运行结果

Figure 1 Results for SVD++

由于对比试验采用了GPU计算框架下的Adam算法, 因此将其记为GPUAdam. 实验采用RMSE (RMSE $=\sqrt{\frac{\text{mse}}{m}}$, m为验证集样本数)衡量算法每次预测评分的准确性, 并记录下了每次迭代距离实验开始的时间点. 两者的对比实验结果如图1所示, 图中横坐标表示算法运行所消耗的时间, 纵坐标表示算法运行过程中的RMSE值, 节点标识算法每完成10次迭代对应的耗时及RMSE值.

由图1看出, BCDSVD++在五个Movielens数据集上均能快速收敛, 而GPUAdam的收敛过程则略显平缓, 需多次迭代才能完全收敛. 从收敛结果看, 两者最终收敛结果相近, 但BCDSVD++具有更高效的收敛能力, 具有更强的运用价值. 不过, 值得注意的是, 本文实验设计的BCDSVD++并未采用任何高性能计算框架且初步研究表明BCDSVD++存在并行计算的可行性. 因此, 进一步研究BCDSVD++并行算法将使SVD++的模型求解性能得到更大的提升.

表2统计实验在达到稳定状态下所需要的迭代次数和所需时间及稳定后的RMSE值, 更加详细地描述了上述实验效果. 经对比可知, 无论是达到稳定状态下的迭代次数还是耗时, BCDSVD++在各个数据集上都具有更好的表现, 从总体上看BCD算法稳定后的RMSE相对更低. 该结果进一步印证了图1所得出的实验结论.

关于FilmTrust数据集的实验结果表明, 该数据在上述实验环境下应用于BCDSVD++及BCD-TrustSVD分别仅需1.6s和2.1s收敛至最低误差, 相比于GPUAdam的12s收敛时间, 具有较大优势. 且收敛后的均方误差分别稳定于0.796和0.794, 两者均具有较低的误差且TrustSVD模型的实验效果有一定的提升, 与Guo等人[8]提供的实验结果相符. 由此表明, 本文所提出的SVD推荐模型梯度求解框

表 2　稳定状态下的RMSE

Table 2　RMSE under stable state

Data sets	GPU-Adam for SVD++			BCD for SVD++		
	Iterations	Running time(s)	RMSE	Iterations	Running time(s)	RMSE
Movielens 100K	96	21.8	0.903	31	4.63	0.904
Movielens 1M	110	56.2	0.849	35	35.9	0.845
Movielens 10M	133	563	0.792	45	510	0.792
Movielens 20M	134	1166	0.782	40	914	0.781
Movielens Last	115	1588	0.795	27	1355	0.795

架以及基于BCD推荐模型求解方法, 能够有效适用于多种的带有隐式反馈的SVD推荐模型, 具有一定的推广价值及应用前景.

6　总结

针对SVD推荐模型, 文本提出了具有一般意义的SVD推荐模型的梯度求解框架. 利用该梯度框架, 本文针对SVD++推荐模型设计了BCDSVD++并实现模型的高效求解, 且该算法能够拓展至其他隐式反馈的SVD推荐模型, 具有一定的理论及应用价值. 实验结果表明, 即使不采用任何高性能计算框架, BCDSVD++在性能方面仍超过了采用了TensorFlow计算框架实现的Adam算法, 具有更强的计算能力.

另一方面, 相关文献[27]表明, ALS方法可通过并行计算提升算法性能. 同本文设计的算法一样, ALS方法也是基于了块梯度下降的思想. 因此, BCDSVD++具有实现并行计算的可能性. 结合上述实验成果, 并行算法的实现将可能为带有隐式反馈的SVD推荐模型的求解带来远比现有技术高效得多的性能提升. 因此, BCDSVD++具有较强的研究潜力及改进空间.

参考文献

1 Shi Y, Larson M, Hanjalic A. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges[J]. ACM Computing Surveys (CSUR), 2014, 47(1): 3.

2 Linden G, Smith B, York J. Amazon. com recommendations: Item-to-item collaborative filtering[J]. IEEE Internet computing, 2003 (1): 76-80.

3 Bell R M, Koren Y. Lessons from the Netflix prize challenge[J]. SiGKDD Explorations, 2007, 9(2): 75-79.

4 Johnson C C. Logistic matrix factorization for implicit feedback data[J]. Advances in Neural Information Processing Systems, 2014, 27.

5 Zhang S, Wang W, Ford J, et al. Using singular value decomposition approximation for collaborative filtering[C]//Seventh IEEE International Conference on E-Commerce Technology (CEC'05). IEEE, 2005: 257-264.

6 Paterek A. Improving regularized singular value decomposition for collaborative filtering[C]//Proceedings of KDD cup and workshop. 2007, 2007: 5-8.

7 Koren Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model[C]//Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2008: 426-434.

8 Guo, Guibing, Jie Zhang, and Neil Yorke-Smith. TrustSVD: collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. Twenty-Ninth AAAI Conference on Artificial Intelligence. 2015.

9 Tang J, Hu X, Gao H, et al. Exploiting local and global social context for recommendation[C]//Twenty-Third International Joint Conference on Artificial Intelligence. 2013.

10 Hu G N, Dai X Y, Qiu F Y, et al. Collaborative filtering with topic and social latent factors incorporating implicit feedback[J]. ACM Transactions on Knowledge Discovery from Data (TKDD), 2018, 12(2): 23.

11 TIAN Y, QIN Y B, XU D Y, et al. TrustSVD Algorithm Based on Double Trust Mechanism[J].Journal of Frontiers of Computer Science and Technology, 2015, 9(11): 1391-1397. [田尧, 秦永彬, 许道云, 等. 基于双信任机制的TrustSVD 算法[J]. 计算机科学与探索, 2015, 9(11): 1391-1397.]

12 Ruder S. An overview of gradient descent optimization algorithms[J]. arXiv preprint arXiv:1609.04747, 2016.

13 Hu Y, Koren Y, Volinsky C. Collaborative Filtering for Implicit Feedback Datasets[C]//ICDM. 2008, 8: 263-272.

14 Gates M, Anzt H, Kurzak J, et al. Accelerating collaborative filtering using concepts from high performance computing[C]//2015 IEEE International Conference on Big Data (Big Data). IEEE, 2015: 667-676.

15 Wang Z, Liu Y, Chiu S. An efficient parallel collaborative filtering algorithm on multi-GPU platform[J]. The Journal of Supercomputing, 2016, 72(6): 2080-2094.

16 Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

17 Song E B, Shi Q J, Zhu Y M. Acceleration of block coordinate descent method achieves the $O\left(1/k^2\right)$ rate of convergence for a convex function with block coordinate strong convexity[J]. (in Chinese).Sci Sin Math, 2016 (010): 1499-1506. [宋恩彬, 史清江, 朱允民. 分块强凸函数的加速块坐标下降算法的$O\left(1/k^2\right)$收敛率[J]. 中国科学: 数学, 2016 (010): 1499-1506.]

18 Xu Y, Yin W. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion[J]. SIAM Journal on imaging sciences, 2013, 6(3): 1758-1789.

19 Shi Q, Sun H, Lu S, et al. Inexact block coordinate descent methods for symmetric nonnegative matrix factorization[J]. IEEE Transactions on Signal Processing, 2017, 65(22): 5995-6008.

20 Barnes R J. Matrix differentiation[J]. Springs Journal, 2006: 1-9.

21 Laue S, Mitterreiter M, Giesen J. Computing Higher Order Derivatives of Matrix and Tensor Expressions[C]//Advances in Neural Information Processing Systems. 2018: 2750-2759.

22 Magnus J R, Neudecker H. Matrix differential calculus with applications in statistics and econometrics[M]. Wiley, 2019.

23 Sherman J, Morrison W J. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix[J]. The Annals of Mathematical Statistics, 1950, 21(1): 124-127.

24 Hager W W. Updating the inverse of a matrix[J]. SIAM review, 1989, 31(2): 221-239.

25 Wang S, Liu J, Shroff N. Coded sparse matrix multiplication[J]. arXiv preprint arXiv:1802.03430, 2018.

26 Buluç A, Gilbert J R. Parallel sparse matrix-matrix multiplication and indexing: Implementation and experiments[J]. SIAM Journal on Scientific Computing, 2012, 34(4): C170-C191.

27 Winlaw M, Hynes M B, Caterini A, et al. Algorithmic acceleration of parallel ALS for collaborative filtering: Speeding up distributed big data recommendation in spark[C]//2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 2015: 682-691.

Efficient solution of SVD recommendation model with implicit feedback

Cai Jianping[1*], Lei Yunqi[2], Chen Mingming[1], Wang Ning[1] & Zhang Shuangyue[1]

1. *College of Information and Smart Electromechanical Engineering, Xiamen Huaxia University, Xiamen* 361021*, China*;
2. *School of Informatics, Xiamen University, Xiamen* 361005*, China*
* Corresponding author. E-mail: caijp@hxxy.edu.cn

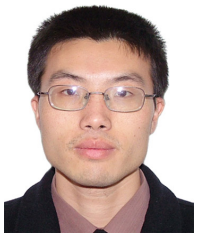
Abstract As an important part of recommendation system, collaborative filtering has become one of the mainstream recommendation methods. In collaborative filtering methods based on potential factors, the SVD recommendation models are often used to analyze user preferences. With the research of SVD recommendation models in recent years, some SVD recommendation models with implicit feedback, such as SVD++, TrustSVD, have been proposed one after another. This kind of models can more effectively mine useful information from limited data sources and achieve better results, so they have drawn widespread attention. However, most of the existing papers focus on model design and lack of efficient algorithms for SVD recommendation models with implicit feedback. Therefore, this paper studies the general gradient solution framework of SVD recommendation model firstly, and then takes the SVD++ recommendation model as a breakthrough, designs an efficient solution algorithm BCDSVD++ based on block coordinate descent method. Besides, we solve the two key problems of capacity matrix inversion and sparse data optimization processing. Experiments show that the proposed algorithm BCDSVD++ has more efficient solution efficiency and convergence ability than the traditional parallel gradient descent method.

Keywords SVD recommendation model, implicit feedback, SVD++, block coordinate descent method, collaborative filtering

Cai Jianping was born in 1990. He received his Master's degree from Fuzhou University in 2016. He is the teaching assistant of the College of Information and Smart Electromechanical Engineering of Xiamen Huaxia University. His current research interests include recommender system, optimization theory and big data.

Lei Yunqi was born in 1963. He received his B. Eng. degree in electronics from University of Science and Technology of China, M. Eng. degree in marine electric engineering from University of The Navy Engineering, China, and the Ph.D. degree in automation from National University of Defense Technology, China, in 1982, 1984 and 1988, respectively. His current research interests include deep learning, computer vision and image processing, big data and cloud computing and computer networks.

Chen Mingming was born in 1979. She is the visiting scholar of University of Illinois in America(2015). She is the dean of the College of Information and Smart Electromechanical Engineering of Xiamen Huaxia University. She is the member of Academic Committee of Xiamen Huaxia University and the recipient of Educational Evaluation Expert of Fujian Province. She received Excellent Teachers Award in Xiamen City(2013). Professor Mingming Chen's research Fields are Information Communication Network System, System Development of Big Data, Communication Network Optimization, Information Storage.



Wang Ning was born in 1979. He is visiting scholar of School of Computer Science, FIU(2016). He is the director of the Engineering Research Center of Fujian Province. He was a recipient of Fujian Provincial Higher Education professional leaders (2014) and Educational Evaluation Expert of Fujian Province(2015). He is the member of Academic Committee of Xiamen Huaxia University and the Executive Director of Systems Engineering Society of Fujian Province. Professor Ning Wang's research fields are Data Mining, System Development of Big Data, Information System Engineering, Cloud Computing.