中国科学: 信息科学 2020 年 第 50 卷 第 10 期: 1544–1558
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: jpingcai@163.com

收稿日期: 2019–04–20; 修回日期: 2019–05–28; 接受日期: 2019–07–16; 网络出版日期: 2020–10–15

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

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均存在影响评分的潜在因

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, 2020, 50: 1544–1558, doi: 10.1360/SSI-2019-0107
Cai J P, Lei Y Q, Chen M M, et al. Efficient solution of the SVD recommendation model with implicit feedback (in Chinese). Sci Sin Inform, 2020, 50: 1544–1558, doi: 10.1360/SSI-2019-0107

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 $\boldsymbol{V} \approx \boldsymbol{U}^{\mathrm{T}}\boldsymbol{M}$, 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 [6] 完善了上述推荐模型并提出了 BiasSVD 推荐模型. 该模型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 [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 [13] 提出的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, 通过算法层面的设计以提升带有隐式反馈的 SVD 推荐模型的算法求解效率具有较大的研究空间. 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:

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

之后, Paterek [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

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

其中 $\boldsymbol{1}$ 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 [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

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

其中 $\boldsymbol{w} \in \mathbb{R}^{m\times 1}$ 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 [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:

$$\boldsymbol{P}=\left(\boldsymbol{U} + \boldsymbol{Y}\boldsymbol{J}^{\mathrm{T}}\operatorname{diag}(\boldsymbol{w}) + \boldsymbol{Z}\boldsymbol{T}^{\mathrm{T}}\operatorname{diag}(\boldsymbol{\omega})\right)^{\mathrm{T}}\boldsymbol{M} + \boldsymbol{\alpha}\boldsymbol{1}_m{}^{\mathrm{T}} + \boldsymbol{1}_n\boldsymbol{\beta}^{\mathrm{T}} + c, \tag{4}$$

其中 $\boldsymbol{\omega} \in \mathbb{R}^{u\times 1}$, 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 (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.

评估 SVD 推荐模型的推荐效果时, 人们常常采用均方误差 (mean square error, MSE) 作为评分预测能力的衡量指标. 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:

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

其中 $*$ 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 $f(\boldsymbol{x})$ 相应地划分为 $f(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_p)$ 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:

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

其中 $\boldsymbol{\xi}_i$ 代替 $\boldsymbol{x}_i$ 作为待优化向量, 其余分块均视为常量. 其算法结构如算法 1 所示.

Algorithm 1 Block coordinate descent method

1: Initialize $\boldsymbol{x}^{(0)} \leftarrow [\boldsymbol{x}_1{}^{(0)}, \boldsymbol{x}_2{}^{(0)}, \ldots, \boldsymbol{x}_p{}^{(0)}]$, $t \leftarrow 0$;
2: **while** stopping criteria have not been satisfied **do**
3: **for** each $i \in \{1, 2, \ldots, p\}$ **do**
4: $\boldsymbol{x}_i{}^{(t+1)} \leftarrow \arg\min_{\boldsymbol{\xi}_i} f(\boldsymbol{x}_1{}^{(t+1)}, \ldots, \boldsymbol{x}_{i-1}{}^{(t+1)}, \boldsymbol{\xi}_i, \boldsymbol{x}_{i+1}{}^{(t)}, \ldots, \boldsymbol{x}_p{}^{(t)})$;
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 $O(1/k)$ 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文献 [18,19] 关于 BCD 方法在此类问题的应用已做了理论研究, 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, 越来越多带有隐式反馈特征的 SVD 模型模型被提出. 因此, 关于带有隐式反馈的 SVD 推荐模型梯度求解理论需求日益凸显. 基于矩阵建模的思想, 本节将结合矩阵微积分 [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 $\boldsymbol{P}=f(\boldsymbol{\xi})$. SVD 推荐模型的求解采用均方误差表达式 (5) 最小化函数 $E(\boldsymbol{\xi})$, 其表达式如下:

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

其中 $\mathrm{reg}(\boldsymbol{\xi})$ 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 $\mathrm{reg}(\boldsymbol{\xi})=\frac{1}{2}\boldsymbol{\xi}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\kappa})\boldsymbol{\xi}$, $\boldsymbol{\kappa}$ 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.

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

$E(\boldsymbol{\xi})$ 的梯度表达式如式 (6) 所示. 式 (6) 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 $E(\boldsymbol{\xi})$ 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:

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

证明 由于 $\boldsymbol{P}=\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}+\cdots$, 因此 $\frac{\partial(\boldsymbol{e}_i{}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{e}_j)}{\partial\boldsymbol{X}}$ 仅与 $\frac{\partial(\boldsymbol{e}_i{}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}\boldsymbol{e}_j)}{\partial\boldsymbol{X}}$ 相关, 即

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

将 $\frac{\partial(\boldsymbol{e}_i{}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{e}_j)}{\partial\boldsymbol{X}}=\boldsymbol{A}^{\mathrm{T}}\boldsymbol{e}_i\boldsymbol{e}_j{}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}$ 代入式 (6) 可得

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

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 $\boldsymbol{P}=\boldsymbol{U}^{\mathrm{T}}\boldsymbol{M}+\boldsymbol{\alpha}\mathbf{1}_m^{\mathrm{T}}+\mathbf{1}_n\boldsymbol{\beta}^{\mathrm{T}}+c$, 其关于用户特征矩阵 $\boldsymbol{U}$ 的子式为 $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{M}$, 由式 (7) 可得 $\boldsymbol{A}=\boldsymbol{I},\boldsymbol{B}=\boldsymbol{M}$. 然后代入式 (7) 可得

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

值得注意的是, 上述表达式直接求出了关于 $\boldsymbol{U}^{\mathrm{T}}$ 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 $\boldsymbol{P}=\left(\boldsymbol{U}+\boldsymbol{Y}\boldsymbol{J}^{\mathrm{T}}\mathrm{diag}\left(\boldsymbol{w}\right)\right)^{\mathrm{T}}\boldsymbol{M}+\boldsymbol{\alpha}\mathbf{1}_m{}^{\mathrm{T}}+\mathbf{1}_n\boldsymbol{\beta}^{\mathrm{T}}+c$. 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作进一步理论分析.

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:

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

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:

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

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

其中 $\oslash$ 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, 难以直接令其梯度为 $\boldsymbol{O}$ 并求得最优解. 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 $\boldsymbol{W}=\boldsymbol{V}-\boldsymbol{\alpha}\boldsymbol{1}_m{}^{\mathrm{T}}-\boldsymbol{1}_n\boldsymbol{\beta}^{\mathrm{T}}-c$, $\boldsymbol{U}'=\boldsymbol{U}+\boldsymbol{Y}\boldsymbol{J}^{\mathrm{T}}\mathrm{diag}\left(\boldsymbol{w}\right)$.

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

上式中, 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:

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

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

考虑了用户评分的隐式反馈, SVD++ 模型的矩阵表达式中特征矩阵 $\boldsymbol{Y}$ 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的特征向量. 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, 本文通过提取 $\frac{\partial E(\boldsymbol{y}_k)}{\partial \boldsymbol{y}_k}$ 中与 $\boldsymbol{y}_k$ 相关的项式得到如下表达式:

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

由推导过程可知, 式 (12) 中的 $\boldsymbol{M}(\boldsymbol{J}*(\boldsymbol{P}-\boldsymbol{V}))^{\mathrm{T}}\mathrm{diag}\left(\boldsymbol{w}\right)\boldsymbol{J}\boldsymbol{e}_k$ 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 $\boldsymbol{y}_k$ 的线性表达式, 最终化简结果如子式 ③ 所示. 式中 $\boldsymbol{Q}=\boldsymbol{J}^{\mathrm{T}}\mathrm{diag}\left(\boldsymbol{w}*\boldsymbol{w}\right)\boldsymbol{J}$ 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:

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

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 $\left(\lambda\boldsymbol{I}+\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}\right)^{-1}$.

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

根据引理 1 可知式中 $\lambda \boldsymbol{I}+\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$ 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 $\lambda \boldsymbol{I}+\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$, 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 $\boldsymbol{R}_k=\left(\lambda \boldsymbol{I}+\boldsymbol{X}_k{\boldsymbol{X}_k}^{\mathrm{T}}\right)^{-1}$. 则有以下递推关系成立:

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

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

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

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

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 $\boldsymbol{R}_{k+1}=\boldsymbol{R}_k-\frac{\boldsymbol{R}_k\boldsymbol{x}_{k+1}{\boldsymbol{x}_{k+1}}^{\mathrm{T}}\boldsymbol{R}_k}{1+{\boldsymbol{x}_{k+1}}^{\mathrm{T}}\boldsymbol{R}_k\boldsymbol{x}_{k+1}}$.

显然, $\boldsymbol{R}_c$ 即 $(\lambda \boldsymbol{I}+\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}})^{-1}$. 其快速迭代求逆的算法过程如算法 2 所示.

Algorithm 2 Fast inversion algorithm for capacitance matrix

Require: $\lambda\,(\lambda>0)$, $\boldsymbol{X}$;
Ensure: $\boldsymbol{R}_c:\left(\lambda \boldsymbol{I}+\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}\right)^{-1}$;
1: Initialize $\boldsymbol{R}_0=\lambda^{-1}\boldsymbol{I}$ and divide $\boldsymbol{X}$ into column vector group $[\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_c]$;
2: **for** each $i\in\{1,2,\ldots,c\}$ **do**
3: 令 $\boldsymbol{t}=\boldsymbol{R}_{i-1}\boldsymbol{x}_i$;
4: $\boldsymbol{R}_i=\boldsymbol{R}_{i-1}-\frac{\boldsymbol{t}\boldsymbol{t}^{\mathrm{T}}}{1+{\boldsymbol{x}_i}^{\mathrm{T}}\boldsymbol{t}}$; //According to (14).
5: **end for**
6: **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 $O\left(f^2c\right)$. 相比而言, 一般方法的时间复杂度为 $O\left(f^2c+f^3\right)$. 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 $m\times m$ 的大型稠密矩阵. 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 $O\left(f^2m^2\right)$, 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.

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 $\boldsymbol{M}\,\mathrm{diag}\left(\boldsymbol{Q}\boldsymbol{e}_k\right)\boldsymbol{M}^{\mathrm{T}}+k_y\boldsymbol{I}$ 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 (relative standard deviation, 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

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

衡, 与各电影最相近的 k 个物品的特征将成为决定 $\boldsymbol{M}\,\mathrm{diag}\,(\boldsymbol{Q}\boldsymbol{e}_k)\,\boldsymbol{M}^{\mathrm{T}}$ 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.

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) 都为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 $\boldsymbol{J}*(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{M})$ 的 Hadamard 积运算子式. 值得注意的是, 该式首先计算的 $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{M}$ 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 (compressed column storage, CCS) 格式存储稀疏矩阵.

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

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 η 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.

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 $\boldsymbol{Q}*\left(\boldsymbol{M}^{\mathrm{T}}\boldsymbol{Y}\right)$ 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, 本文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.

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

分析可知, 直接计算 $\boldsymbol{M}(\boldsymbol{Q}*(\boldsymbol{M}^{\mathrm{T}}\boldsymbol{Y}))$ 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.

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[1)], 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 @ 3.60 Hz, 16 GB RAM, Nvidia GeForce GTX 1060 6 GB. 数据集采用 MovieLens 系列数据集, 该数据集[2)]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. 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

1) https://github.com/imcjp/BCDSvdppExp.
2) https://grouplens.org/datasets/movielens.

Algorithm 3 Solving algorithm based on block coordinate descent for SVD++ (BCDSVD++)

Require: score matrix $\boldsymbol{V}$, $\boldsymbol{V}'$, indication matrix $\boldsymbol{J}$, $\boldsymbol{J}'$, maximum number of iterations iter, topK, regularization parameters k_u, k_m, k_a, k_b, k_y;

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

// The following is the algorithm preprocessing and initialization process, constant data should be completed in the process as far as possible.

Initialize the feature matrices $\boldsymbol{U}^{(0)}$, $\boldsymbol{M}^{(0)}$, $\boldsymbol{\alpha}^{(0)}$, $\boldsymbol{\beta}^{(0)}$ randomly and ensure the elements in them satisfy normal distribution $\mathcal{N}(0,1)$, but initialize $\boldsymbol{Y}^{(0)} = \boldsymbol{O}$;

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

Calculate $\boldsymbol{w}$ to satisfy $w_i = (\sum_{k=1}^{u} j_{ik})^{-\frac{1}{2}}$, then let$\boldsymbol{J}_w = \mathrm{diag}\,(\boldsymbol{w})\,\boldsymbol{J}$ and solve $\boldsymbol{Q} = \boldsymbol{J}_w^{\mathrm{T}}\boldsymbol{J}_w$;

Copy $\boldsymbol{Q}$ to $\boldsymbol{Q}^*$, and set the numbers which are not in maximum topK number of each column $\boldsymbol{Q}^*$ to 0, then multiply the corresponding coefficients for each column of $\boldsymbol{Q}^*$ to keep the sum of each column the same as the corresponding column of $\boldsymbol{Q}$; // See the discussion on optimization of matrix inversion problem in Subsection 4.2.

while $t <$ iter **do**

  // The following solves $\boldsymbol{U}^{(t+1)}$ according to (10), and using matrix $\boldsymbol{B}$ to represent temporary matrix in solving process.

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

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

    Select the columns from $\boldsymbol{M}^{(t)}$ that satisfies $j_{ki} = 1, 1 \leqslant i \leqslant m$ to form a new matrix $\boldsymbol{M}^*$;

    // The process of solving linear equation here and below can be realized according to Theorem 2 and Algorithm 2.

    Solve the linear equation about $\boldsymbol{u}_k^{(t+1)}$: $(\boldsymbol{M}^*\boldsymbol{M}^{*\mathrm{T}} + k_u\boldsymbol{I})\boldsymbol{u}_k^{(t+1)} = \boldsymbol{b}_k$; //$\boldsymbol{b}_k$ is kth column of $\boldsymbol{B}$.

  end for

  Combine $\boldsymbol{u}_k^{(t+1)}$ to make $\boldsymbol{U}^{(t+1)} = [\boldsymbol{u}_1^{(t+1)}, \boldsymbol{u}_2^{(t+1)}, \ldots, \boldsymbol{u}_u^{(t+1)}]$;

  // The following solves $\boldsymbol{M}^{(t+1)}$ according to (11).

  Solve $\boldsymbol{U}'^{(t+1)}$=$\boldsymbol{U}^{(t+1)} + \boldsymbol{Y}^{(t)}\boldsymbol{J}_w^{\mathrm{T}}$, and then let $\boldsymbol{B} = \boldsymbol{U}'^{(t+1)}(\boldsymbol{J} * (\boldsymbol{V} - \boldsymbol{\alpha}^{(t)}\boldsymbol{1}_m{}^{\mathrm{T}} - \boldsymbol{1}_n\boldsymbol{\beta}^{(t)\mathrm{T}} - c))$;

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

    Select the columns from $\boldsymbol{U}'^{(t+1)}$ that satisfies $j_{ik} = 1, 1 \leqslant i \leqslant u$ to form a new matrix $\boldsymbol{U}^*$;

    Solve the linear equation about $\boldsymbol{m}_k^{(t+1)}$: $(\boldsymbol{U}^*\boldsymbol{U}^{*\mathrm{T}} + k_m\boldsymbol{I})\boldsymbol{m}_k^{(t+1)} = \boldsymbol{b}_k$; //$\boldsymbol{b}_k$ is kth column of $\boldsymbol{B}$.

  end for

  // The following solves $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ according to (8) and (9).

  Combining $\boldsymbol{m}_k^{(t+1)}$, we have $\boldsymbol{M}^{(t+1)} = [\boldsymbol{m}_1^{(t+1)}, \ldots, \boldsymbol{m}_m^{(t+1)}]$;

  Let $\boldsymbol{B} = \boldsymbol{J} * (\boldsymbol{V} - \boldsymbol{U}'^{(t+1)\mathrm{T}}\boldsymbol{M}^{(t+1)} - c)$;

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

  // According to (13), solve $\boldsymbol{Y}^{(t+1)}$ by using sparse optimization expression (15) and (16).

  $\boldsymbol{B} \leftarrow \boldsymbol{M}^{(t+1)}(\boldsymbol{Q} * (\boldsymbol{M}^{(t+1)\mathrm{T}}\boldsymbol{Y}^{(t)})) + \boldsymbol{M}^{(t+1)}(\boldsymbol{B} - \boldsymbol{J} * (\boldsymbol{\alpha}^{(t+1)}\boldsymbol{1}_m{}^{\mathrm{T}} + \boldsymbol{1}_n\boldsymbol{\beta}^{(t+1)\mathrm{T}}))^{\mathrm{T}}\boldsymbol{J}_w$;

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

    Let $\boldsymbol{M}^* = \boldsymbol{M}^{(t+1)}\,\mathrm{diag}(\boldsymbol{q}_k^*)$. In this expression, $\boldsymbol{q}_k^*$ denote the kth column of $\boldsymbol{Q}^*$;

    Solve the linear equation about $\boldsymbol{y}_k^{(t+1)}$: $(\boldsymbol{M}^*\boldsymbol{M}^{*\mathrm{T}} + k_y\boldsymbol{I})\boldsymbol{y}_k^{(t+1)} = \boldsymbol{b}_k$; // $\boldsymbol{b}_k$ is kth column of $\boldsymbol{B}$.

  end for

  // Substitute $\boldsymbol{V}'$ and $\boldsymbol{J}'$ as verification set to find the mean square error.

  Using $\boldsymbol{U}^{(t+1)}$, $\boldsymbol{M}^{(t+1)}$, $\boldsymbol{Y}^{(t+1)}$, $\boldsymbol{\alpha}^{(t+1)}$, $\boldsymbol{\beta}^{(t+1)}$ to calculate $\mathrm{mse}^{(t+1)}$;

  // Check whether there has been overfitting.

  If $\mathrm{mse}^{(t+1)} > \mathrm{mse}^{(t)}$, terminate the iteration and exit the loop; Otherwise, let $t \leftarrow t + 1$ and go to next;

end while

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

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

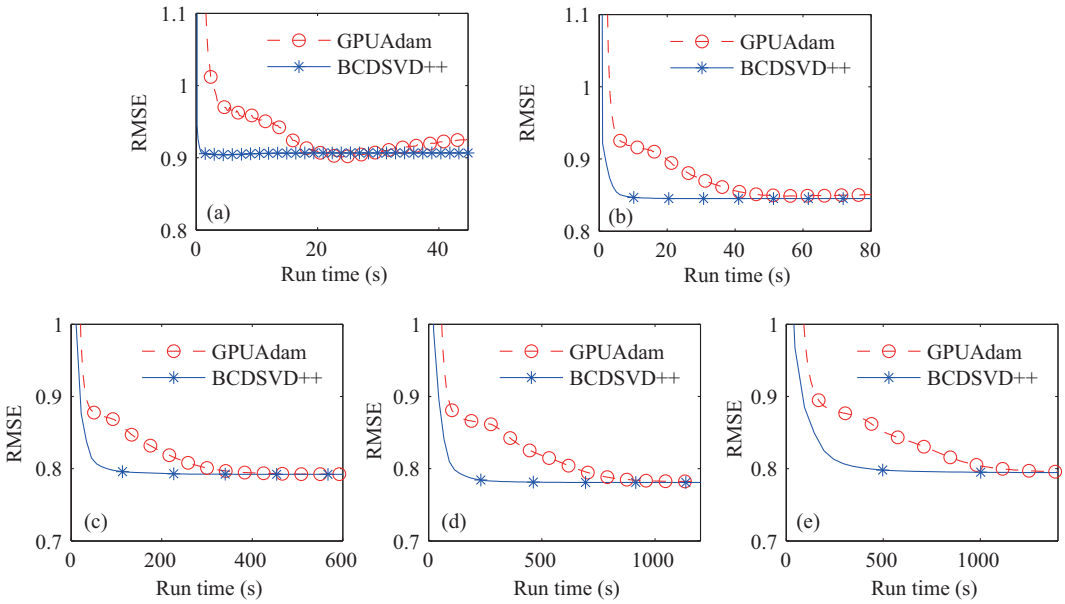


图 1 (网络版彩图) SVD++ 运行结果

Figure 1 (Color online) Results for SVD++. (a) SVD++ for Movielens 100K; (b) SVD++ for Movielens 1M; (c) SVD++ for Movielens 10M; (d) SVD++ for Movielens 20M; (e) SVD++ for Movielens Last

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第 4.4 小节提供的源码附件.

5.2 实验结果

由于对比实验采用了 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 5 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 1.6 s 和 2.1 s 收敛至最低误差, 相比于 GPUAdam 的 12 s 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 [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架以及基于 BCD 推荐模型求解方法, 能够有效适用于多种带有隐式反馈的 SVD 推荐模型, 具

表 2 稳定状态下的 RMSE

Table 2 RMSE under the 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

有一定的推广价值及应用前景.

6 总结

针对 SVD 推荐模型, 本文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. ACM Comput Surv, 2014, 47: 1–45

2 Linden G, Smith B, York J. Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput, 2003, 7: 76–80

3 Bell R M, Koren Y. Lessons from the Netflix prize challenge. SIGKDD Explor Newsl, 2007, 9: 75–79

4 Johnson C C. Logistic matrix factorization for implicit feedback data. In: Proceedings of 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. In: Proceedings of the 7th IEEE International Conference on E-Commerce Technology (CEC'05). New York: IEEE, 2005. 257–264

6 Paterek A. Improving regularized singular value decomposition for collaborative filtering. In: Proceedings of KDD Cup and Workshop, 2007. 5–8

7 Koren Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2008. 426–434

8 Guo G B, Zhang J, Yorke-Smith N. TrustSVD: collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence, Austin, 2015. 123–129

9 Tang J L, Hu X, Gao H J, et al. Exploiting local and global social context for recommendation. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, 2013. 2712–2718

10 Hu G N, Dai X Y, Qiu F Y, et al. Collaborative filtering with topic and social latent factors incorporating implicit feedback. ACM Trans Knowl Discov Data, 2018, 12: 1–30

11 Tian Y, Qin Y B, Xu D Y, et al. TrustSVD algorithm based on double trust mechanism. J Front Comput Sci Tech, 2015, 9: 1391–1397 [田尧, 秦永彬, 许道云, 等. 基于双信任机制的 TrustSVD 算法. 计算机科学与探索, 2015, 9: 1391–1397]

12 Ruder S. An overview of gradient descent optimization algorithms. 2016. ArXiv: 1609.04747

13 Hu Y F, Koren Y, Volinsky C. Collaborative filtering for implicit feedback datasets. In: Proceedings of the 8th IEEE International Conference on Data Mining. Washington: IEEE Computer Society, 2008. 8: 263–272

14 Gates M, Anzt H, Kurzak J, et al. Accelerating collaborative filtering using concepts from high performance computing. In: Proceedings of 2015 IEEE International Conference on Big Data. New York: IEEE, 2015. 667–676

15 Wang Z, Liu Y, Chiu S. An efficient parallel collaborative filtering algorithm on multi-GPU platform. J Supercomput, 2016, 72: 2080–2094

16 Kingma D P, Ba J. Adam: a method for stochastic optimization. 2014. ArXiv: 1412.6980

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. Sci Sin Math, 2016, 46: 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. 中国科学: 数学, 2016, 46: 1499–1506]

18 Xu Y, Yin W. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. SIAM J Imag Sci, 2013, 6: 1758–1789

19 Shi Q, Sun H, Lu S, et al. Inexact block coordinate descent methods for symmetric nonnegative matrix factorization. IEEE Trans Signal Process, 2017, 65: 5995–6008

20 Barnes R J. Matrix differentiation. Springs J, 2006, 2006: 1–9

21 Laue S, Mitterreiter M, Giesen J. Computing higher order derivatives of matrix and tensor expressions. In: Proceedings of Advances in Neural Information Processing Systems, 2018. 2750–2759

22 Magnus J R, Neudecker H. Matrix Differential Calculus With Applications in Statistics and Econometrics. Hoboken: John Wiley & Sons, 2019

23 Sherman J, Morrison W J. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. Ann Math Statist, 1950, 21: 124–127

24 Hager W W. Updating the inverse of a matrix. SIAM Rev, 1989, 31: 221–239

25 Wang S N, Liu J S, Shroff N. Coded sparse matrix multiplication. 2018. ArXiv: 1802.03430

26 Buluç A, Gilbert J R. Parallel sparse matrix-matrix multiplication and indexing: implementation and experiments. SIAM J Sci Comput, 2012, 34: 170–191

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. In: Proceedings of the 21st International Conference on Parallel and Distributed Systems (ICPADS). New York: IEEE, 2015. 682–691

Efficient solution of the SVD recommendation model with implicit feedback

Jianping CAI[1*], Yunqi LEI[2], Mingming CHEN[1], Ning WANG[1] & Shuangyue ZHANG[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: jpingcai@163.com

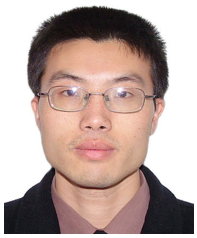
Abstract Collaborative filtering, an important part of the recommendation system, has become a mainstream recommendation method. In collaborative filtering methods based on potential factors, SVD recommendation models are often used to analyze user preferences. With the recent research of SVD recommendation models, some SVD recommendation models with implicit feedback, such as SVD++ and TrustSVD, have been successively proposed. These types of models can more effectively mine useful information from limited data sources and achieve better results than traditional SVD recommendation model, thereby garnering widespread attention. However, most existing papers focus on model design and the lack of efficient algorithms for SVD recommendation models with implicit feedback. Therefore, this paper first studies the general gradient solution framework of the SVD recommendation model. Then, it considers the SVD++ recommendation model as a breakthrough and designs an efficient solution algorithm, namely, BCDSVD++, based on the block coordinate descent method. Furthermore, we solve the two key problems of capacity matrix inversion and sparse data optimization processing. Experiments show that the proposed BCDSVD++ algorithm yields better 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

Jianping CAI was born in 1990. He received his M.S. degree from Fuzhou University in 2016. At present, he is the teaching assistant at the College of Information and Smart Electromechanical Engineering of Xiamen Huaxia University. His current research interests are recommender system, optimization theory, and big data.

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

Mingming CHEN was born in 1979. She is a visiting scholar at the University of Illinois, US (2015). She is a recipient of the educational evaluation expert of Fujian Province. Her research fields include information communication network system, system development of big data, communication network optimization, information storage.



Ning WANG was born in 1979. He is a visiting scholar at the School of Computer Science, Florida International University (2016). He is aeducational evaluation expert of Fujian province (2015). His research fields include data mining, system development of big data, information system engineering, and cloud computing.