基于数据流和点对点网络的分布式推荐算法

丛义昊,于艳华
北京邮电大学 计算机学院,北京 100876

摘 要:推荐算法是数据挖掘中应用最广泛的算法之一,目前的推荐算法主要是针对静态数据的,缺乏对动态数据的适应性,基于数据流的推荐算法是解决这一问题的方法。针对目前在分布式平台中采用参数服务器控制模型训练存在的滞后梯度和掉队者问题,提出了一种新的使用点对点参数交换网络代替参数服务器的方法,并在训练过程中引入遗忘策略和异常评分检测能力。在新的分布式流计算框架Flink上进行设计实现,并在经典的MovieLens-1m数据集上进行了实验。实验结果表明,该算法能够在保证推荐准确率的同时,降低一半通讯开销。

关键词:在线矩阵分解;流计算;分布式协同过滤;点对点网络
文献标志码:A **中图分类号:**TP301.6 doi:10.3778/j.issn.1002-8331.1710-0160

丛义昊,于艳华.基于数据流和点对点网络的分布式推荐算法.计算机工程与应用,2019,55(1):64-69.
CONG Yihao, YU Yanhua. Online distributed recommendation algorithm based on data stream and peer-to-peer network. Computer Engineering and Applications, 2019, 55(1):64-69.

Online Distributed Recommendation Algorithm Based on Data Stream and Peer-to-Peer Network

CONG Yihao, YU Yanhua
School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract: Recommendation algorithm is one of the most widely used algorithms in data mining. However, recent studies focus on static data and lack the adaptability to dynamic data. Recommendation algorithm based on data stream is the solution to this problem. Aiming at the straggler and delayed-gradient problems in using parameter server to control model training in distributed platform, a new method of using peer-to-peer parameter exchange network is proposed, and the forgetting strategy and anomaly detection ability are introduced in the training process. Algorithm is implemented on Flink and experiments on Movielens-1m. Experimental results show that the algorithm can reduce the communication cost by half, while ensuring the accuracy of recommendation.

Key words: online matrix factorization; stream computing; distributed collaborative filtering; peer-to-peer network

1 引言

推荐系统在如今的互联网产品和应用中被广泛采用,包括日常使用的电子商务产品推荐、相关搜索、话题推荐和社交网络当中的交友推荐等。现有推荐算法包括基于内容的推荐、协同过滤推荐等多种推荐算法。矩阵分解(Matrix Factorization,MF)[1]作为协同过滤推荐算法的一个分支,由于其具有很高的推荐准确度,在推荐系统中被广泛使用。随着互联网技术的发展,信息总量呈爆发性增长态势,且越来越多的数据以数据流的形式到来,为推荐系统提出了新的挑战。如何实现在推荐系统中对海量、动态、无限的数据流进行快速挖掘,并充分利用数据流的时间特性提高推荐的准确性,成为近年来推荐系统的研究热点[2-7]。

文献[8]提出了一种通过共享内存和随机梯度下降(Stochastic Gradient Descent,SGD)实现单机并行矩阵分解的方法,通过无锁更新模型参数,能够获得接近线性的加速比。但该文提出的方式未对模型参数更新提供同步控制(详见3.2节),无法确保损失函数收敛,并且单机共享模型难以推广到分布式环境中。文献[9]提出了一种基于参数服务器和BSP(Bulk Synchronous

基金项目:国家自然科学基金(No.61702046)。
作者简介:丛义昊(1993—),男,硕士,研究领域为大数据与数据挖掘,E-mail: congyihao@126.com;于艳华(1974—),女,博士,副教授,研究领域为大数据与数据挖掘。
收稿日期:2017-10-18 **修回日期:**2017-12-20 **文章编号:**1002-8331(2019)01-0064-06
CNKI **网络出版:**2018-06-28, http://kns.cnki.net/kcms/detail/11.2127.TP.20180627.1642.002.html

Parallel,块同步并行)方式的分布式矩阵分解方案,由于有“超步”的存在,不会引起死锁,但会为矩阵分解引入“掉队者(straggler)”问题,降低训练效率。文献[10]提出了一种基于参数服务器的分布式异步随机梯度下降(Asynchronous SGD,ASGD)方案,使模型训练效率大大提高,但会为矩阵分解引入滞后梯度(delayed gradient)问题,且尚未有理论证明算法一定收敛。文献[11-13]提出了一种基于参数服务器和SSP(Stale Synchronous Parallel)方式的分布式矩阵分解方案,通过限制参数陈旧度(bounded stale),证明算法一定会收敛,但算法属于宽松一致性(relaxed consistency)模型,仍不能完全避免滞后梯度问题。文献[3]和[4]通过在推荐系统中引入艾宾浩斯遗忘曲线以挖掘时间特性对用户兴趣的影响。但该模型只能应对缓慢的用户兴趣变化,无法应对突变型用户兴趣漂移,且无法检测和处理异常评分对模型的影响。

目前基于矩阵分解的分布式推荐系统,主要使用参数服务器控制模型的训练过程,这使得算法难以在去中心化的环境中使用。并且缺乏对于用户兴趣漂移和异常评分的检测与处理。

本文针对以上问题,提出一种自适应分布式数据流环境的SGD(Distributed Streaming SGD,DS-SGD)算法。面对海量、动态、无限的数据流,通过点对点(Peer-to-Peer,P2P)网络控制模型训练过程,对矩阵分解模型实时增量更新,避免了引入参数服务器带来的性能瓶颈,并显著降低通信开销;通过控制训练数据的迭代次数,支持One Pass和任意次数迭代计算;并充分利用用户评分数据的时间戳信息,对用户的兴趣漂移进行快速响应。与经典的基于矩阵分解的推荐算法进行对比,实验结果表明,算法能够显著提升推荐系统时效性,同时保证推荐准确度。

2 带偏置的矩阵分解

矩阵分解模型的基本原理是通过将评分矩阵分解为两个低阶矩阵,用两个因子矩阵的乘积来逼近原来的评分矩阵,训练的目标是使两个因子矩阵的乘积和评分矩阵之间的误差最小化。考虑用户数为m,物品数为n的场景,评分矩阵可以表示为 $R\in R^{m\times n}$,问题表示为求两个低阶矩阵 $U\in R^{k\times m}$ 和 $V\in R^{k\times n}$,使得 $R\approx U^{\mathrm{T}}V$,最优化问题等价于:

$$\arg\min_{\substack{U\in R^{k\times m}\\V\in R^{k\times n}}}\sum\left(R_{ij}-\left(U^{\mathrm{T}}V\right)_{ij}\right)^2 \tag{1}$$

为了防止过拟合问题的出现,对 U、V 矩阵引入惩罚因子以减小影响,调整后的目标函数如式(2)所示:

$$\arg\min_{\substack{U\in R^{k\times m}\\V\in R^{k\times n}}}\sum\left(R_{ij}-\left(U^{\mathrm{T}}V\right)_{ij}\right)^2+\lambda\|U\|^2+\lambda\|V\|^2 \tag{2}$$

对于上述最优化问题,经常使用SGD来求解,用户隐向量和物品隐向量的更新方法为:

$$\begin{cases}u_i\leftarrow u_i-2\eta\left(\left(u_i{}^{T}v_j-R_{ij}\right)v_j+\dfrac{\lambda}{m}u_i\right)\\ v_j\leftarrow v_j-2\eta\left(\left(u_i{}^{T}v_j-R_{ij}\right)u_i+\dfrac{\lambda}{n}v_j\right)\end{cases} \tag{3}$$

在实际生活中,还存在一种现象:某个用户对每个物品的评分普遍偏高;或某个物品比其他物品相比有较高的评分。这就涉及到了用户偏好问题,即用户具有不同的评分习惯,或者某个物品普遍受到欢迎。通常的做法是在所有物品预测平均分 μ 的基础上,加上用户偏好 b_i 和物品偏好 b_j 来修正这个问题,修正后的评分为:

$$\hat{r}_{ui}=\mu+b_i+b_j+u^{\mathrm{T}}v \tag{4}$$

加入了用户偏好之后的损失函数修正为:

$$\arg\min_{\substack{U\in R^{k\times n}\\V\in R^{k\times m}}}\sum\left(R_{ij}-\mu-b_i-b_j-\left(U^{T}V\right)_{ij}\right)^2+\lambda\left(\sum_i b_i{}^2+\sum_j b_j{}^2+\|U\|^2+\|V\|^2\right) \tag{5}$$

式中,$\sum_i b_i{}^2$ 表示所有用户偏好的平方和,$\sum_j b_j{}^2$ 表示所有物品偏好的平方和。对修正后的损失函数求解,基本能消除用户偏好带来的影响。

3 基于数据流和点对点网络的推荐算法

3.1 训练集与模型分布

传统的矩阵分解算法的评分数据和推荐模型是集中式存储的,这使得算法难以在去中心化的环境中使用[14]。很多算法在启动阶段即需要用户数量、物品数量等信息,而在数据流环境中,用户数量、物品数量等是动态变化的,并且分布式环境中计算节点数目也可能随着计算需求而动态变化,所以训练集与模型的分布需要具有根据计算节点的增减而动态调整的能力。DS-SGD算法选用一致性哈希算法作为 U、R 矩阵的数据划分算法,将数据流形式到来的数据经过哈希函数集的处理映射到由虚拟节点构成的哈希环上,保证了计算节点数变化时,训练集和模型的移动代价最小。

通过式(3)可以看出,SGD算法每轮迭代计算涉及到三个参数 u_i、v_j 和 R_{ij}。训练集 R 与模型 U 经过图1所示的划分后,在任意节点上,至少有 u_i 和 R_{ij} 可以在worker节点本地找到。物品隐向量矩阵 V 无法使用与 U、R 相同的划分方式,因为对于评分 R_{ij} 而言,只能与 u_i、v_j 其一(本文为 u_i)使用相同的分区key,达到本地计算的目的,所以 v_j 必须通过节点间通信来交换。通常的做法是采用参数服务器来控制参数交换过程,详见下一节。

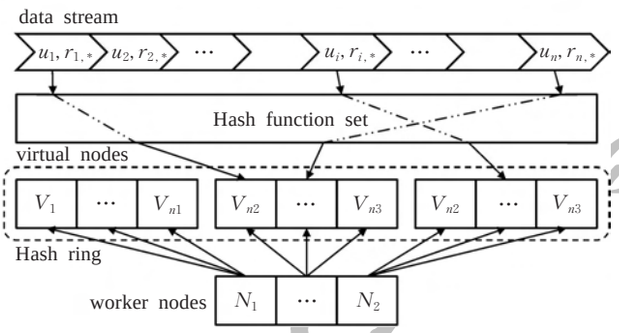

图1 训练集R与模型U的划分

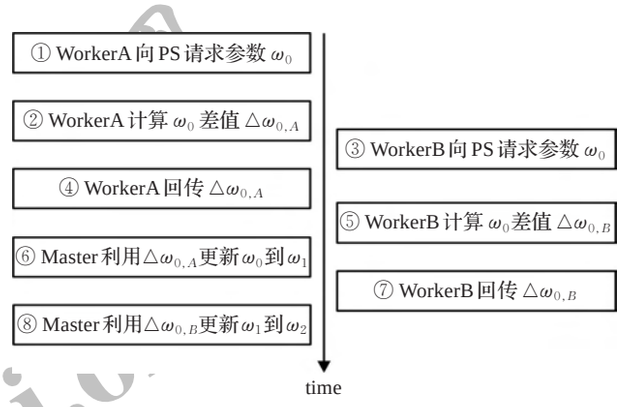

图2 滞后梯度

3.2 点对点参数交换网络

使用SGD求矩阵分解模型实际上是一个顺序更新U、V矩阵的过程。因为SGD本质上是一个迭代收敛式算法,即后一步的更新过程可能会依赖于前一步的结果,所以传统的SGD算法不能直接应用于分布式环境。Gemulla等人证明,只有当并行更新U、V矩阵的过程中使用的训练数据集严格满足式(6)时,才可进行并行模型更新:

$$\begin{cases} i_1 \neq i_2 \neq \cdots \neq i_n \\ j_1 \neq j_2 \neq \cdots \neq j_n \\ (i,j) \in [n] \times [m], R_{ij} \neq 0 \end{cases} \tag{6}$$

当前的工作都是通过使用参数服务器来控制模型的训练过程。其中具有代表性的实现是Google的Jeff Dean提出的DistBelief,主要用于大规模深度学习网络训练。DistBelief将深度学习模型存储在全局的参数服务器中,计算节点间通过参数服务器进行信息传递,是解决SGD和L-BFGS算法分布式训练问题的一种可行方案。

基于参数服务器的矩阵分解有两种方案:一种方案是使用异步SGD控制模型训练,此种方式会受到滞后梯度问题的影响,指的是一个本地的worker未能及时将参数更新值传回全局模型,而其他worker在此期间请求了旧的模型参数并在其基础上进行梯度计算的情况。

如图2所示,WorkerA在第二步进行参数ω_0的梯度计算时,还未及时将差值$\Delta\omega_{0,A}$回传至参数服务器,WorkerB即向参数服务器请求了“过时”的参数ω_0,最终导致在第⑧步时,参数服务器错误的使用了过时的$\Delta\omega_{0,B}$更新了ω_1。参数服务器的一大作用是对参数进行显式的加锁与解锁,对于本例而言,就是在WorkerB向参数服务器请求参数ω_0时,检测到参数正在被WorkerA使用,从而延迟请求的生效,直到第⑥步结束之后才允许重新请求参数ω_1;此外,参数服务器还负责确保同时处于更新状态的参数满足式(6)的条件。

另外一种解决方案是使用BSP或SSP的方式控制模型训练,此方案会受到掉队者问题影响,指的是一轮迭代的时间受制于最慢计算节点的计算速度,在BSP方式中该问题尤为明显。

算法1 Peer-to-Peer Model Exchange

```
for each v_j in IterativeItemStream <key, v_j>
  Get propagate direction from key
  Get lastBinaryIdx from key
  Get lastNode from key
  if v_j reach the border of workers
    change direction
  nextNode=lastNode^(1<<lastBinaryIdx)
  update key by nextNode and direction
  DS_SGD(v_j^t)
  send <key, v_j^{t+1}> to worker_{nextNode}
end
```

在以上方式中,显然参数服务器是分布式系统的瓶颈。由于3.1节通过相同的key对U、R矩阵进行一致性哈希划分,则在分布式并行计算中,式(6)中的第一个条件必然满足,无需参数服务器进行同步控制;对于V矩阵,将其转换为迭代数据流的形式,在所有worker节点间通过算法1定义的点对点参数交换算法进行传播,对于任意$v_j \in V, j \leqslant m$,v_j在全局有且仅有一份副本,v_j的更新始终在worker节点本地进行,从而解决了ASGD的滞后梯度问题。同时点对点参数交换算法还能够保证对于任意v_j,总是能够被传播到所有计算节点,参与所有r_{ij}的训练过程。

在图3中以4节点4参数为例,演示了参数的交换和更新过程。每个节点在更新完本地参数后,通过计算得出下一个传播节点,将持有的参数传递过去,然后重复上述过程。对于2^N个worker的,通过2^N-1次传播,每个参数都会在所有节点上完成一轮迭代计算。

通过本地计算和点对点参数交换,虽然无需引入参数服务器,但模型训练所需的通信总开销并没有减少。由于SGD求解矩阵分解模型的过程中,算法在有限步(N步)内具有可交换性和幂等性,所以可进一步降低节点间通信开销,即无需每次计算之后就传递参数,通过证明发现,可以在worker本地最大迭代N次之后再进行参数交换,仍能保证损失函数收敛。

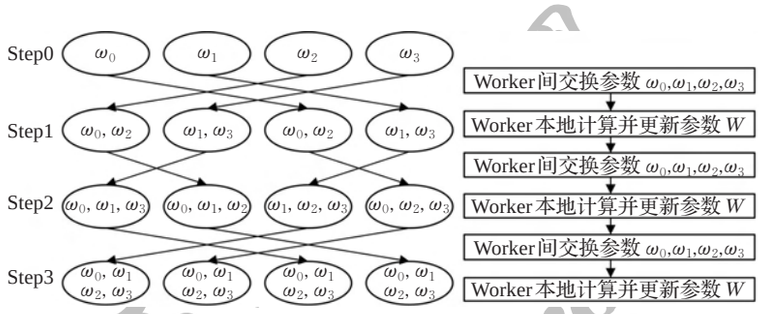

图3 点对点参数交换

SGD算法的可交换性和幂等性表达如式(7)、(8)所示:

$$f\left(f\left(\omega^{t_{n1}},r^{t_{n1}}\right),r^{t_{n2}}\right)=f\left(f\left(\omega^{t_{n2}},r^{t_{n2}}\right),r^{t_{n1}}\right) \tag{7}$$

$$f\left(f^{n}\left(r^{t_n}\right)\right)\approx f\left(f^{n-1}\left(r^{t_{n-1}}\right)\right)\approx\cdots\approx f(r) \tag{8}$$

证明 SGD算法在任意参数交换步长$<N$时,损失函数收敛性不变。

假设worker1、2之间需要交换参数,worker1上现有输入数据r^{t_0}和r^{t_1},worker2具有输入数据r^{t_2}和r^{t_3}。在SGD算法中,将模型参数的更新用式(9)表示:

$$\omega^{t_{i+1}}=\omega^{t_i}+\Delta\omega^{t_i}_{r^{t_j}} \tag{9}$$

代入式(7)、(8)可得:

$$\begin{cases}\Delta\omega^{t_i}_{r^{t_{j1}}}+\Delta\omega^{t_{i+1}}_{r^{t_{j2}}}=\Delta\omega^{t_i}_{r^{t_{j2}}}+\Delta\omega^{t_{i+1}}_{r^{t_{j1}}}\\ \omega^{t_i}+\Delta\omega^{t_i}_{r^{t_j}}\approx\omega^{t_i}\end{cases} \tag{10}$$

由式(10)可得,对于迭代一步交换参数的情况有:

$$\omega_{\mathrm{I}}\approx\omega^{t_0}+\Delta\omega^{t_0}_{r^{t_0}}+\Delta\omega^{t_1}_{r^{t_2}}+\Delta\omega^{t_2}_{r^{t_1}}+\Delta\omega^{t_3}_{r^{t_3}} \tag{11}$$

对于迭代两步交换参数的情况有:

$$\omega_{\mathrm{I}}\approx\omega^{t_0}+\Delta\omega^{t_0}_{r^{t_0}}+\Delta\omega^{t_1}_{r^{t_1}}+\Delta\omega^{t_2}_{r^{t_2}}+\Delta\omega^{t_3}_{r^{t_3}}=$$
$$\omega^{t_0}+\Delta\omega^{t_0}_{r^{t_0}}+\Delta\omega^{t_1}_{r^{t_2}}+\Delta\omega^{t_2}_{r^{t_1}}+\Delta\omega^{t_3}_{r^{t_3}} \tag{12}$$

推导(11)、(12)可得:

$$\omega_{\mathrm{I}}\approx\omega_{\mathrm{II}} \tag{13}$$

即迭代一次交换参数和迭代两次交换参数最终参数训练结果近似相等。将证明进一步推广,可得:

$$\omega_{\mathrm{I}}\approx\omega_{\mathrm{II}}\approx\omega_{\mathrm{III}}\approx\cdots\approx\omega_N \tag{14}$$

即只要满足在N步内交换参数,可在减小通信开销的同时,不影响函数的收敛性。

3.3 基于Flink的DS-SGD算法实现

3.3.1 DS-SGD算法

以第2章带偏置的矩阵分解作为基础算法,结合3.1、3.2节的数据划分策略和点对点参数交换网络,得到如算法2所示的DS-SGD算法。

算法2 DS-SGD

initialize $\boldsymbol{u}_i,\boldsymbol{v}_j$

partition U,R by userId using ConsistenceHashing

exchange V via peer-to-peer network

foreach worker in worker set

  foreach r_{ij} in R_{local} and $iter<N$

    RatingBasedAnomalyDetector(r_{ij})

    FactorBasedForgettingFilter(r_{ij})

    $\boldsymbol{u}_i\leftarrow\boldsymbol{u}_i-2\eta\left(\left(\boldsymbol{u}_i^{T}\boldsymbol{v}_j-R_{ij}\right)\boldsymbol{v}_j+\frac{\lambda}{m}\boldsymbol{u}_i\right)$

    $\boldsymbol{v}_j\leftarrow\boldsymbol{v}_j-2\eta\left(\left(\boldsymbol{u}_i^{T}\boldsymbol{v}_j-R_{ij}\right)\boldsymbol{u}_i+\frac{\lambda}{n}\boldsymbol{v}_j\right)$

    exchange V via peer-to-peer network

  end

end

对于首次在数据流中出现的用户i和物品j,算法需要对用户隐向量$\boldsymbol{u}_i$与物品隐向量$\boldsymbol{v}_j$在(0,1)区间做随机初始化,然后通过一致性哈希算法,将$\boldsymbol{u}_i$与r_{ij}路由到对应的worker节点上;对于$\boldsymbol{v}_j$,通过点对点参数交换网络,以迭代数据流的方式在worker节点间传递,每次到达新的worker节点,触发$1\sim N$次的计算,计算结果首先需要经过异常评分检测函数和遗忘函数(详见第4章)的过滤,然后才能应用到参数更新当中。

3.3.2 推荐系统架构

本文在Flink上对DS-SGD算法进行实现。对比第一代无状态分布式流计算框架Storm等,Flink的算子内置状态(state)管理功能,非常适合复杂机器学习算法的实现[15]。基于Flink和DS-SGD算法的推荐系统架构,如图4所示。

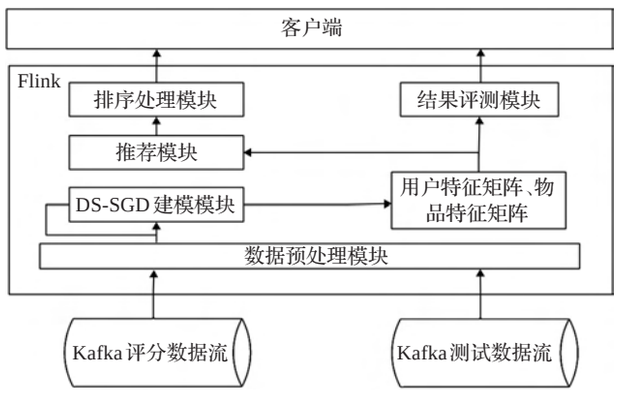

图4 基于Flink和DS-SGD的推荐系统

系统以Kafka作为评分数据源,经过数据预处理模块,将评分变为Flink编程模型中的*Tuple*3<*userId*,*ItemId*,*rating*>元组类型。经过DS-SGD增量建模模块

的处理,生成用于推荐的用户隐向量矩阵 U 和物品隐向量矩阵 V ,以 *MapState<id,vector>* 的结构作为算子的状态保存在 Flink 集群 worker 节点的内存中。通过 Flink 提供的 *QueryableStateClient* 支持,向用户提供推荐接口,也可直接基于 U、V 矩阵进行 RMSE(Root Mean Square Error,均方根误差)等结果指标评测。

4 用户兴趣漂移应对和异常评分检测

4.1 具有遗忘性的矩阵分解

数据流的演变会造成概念漂移:新产品的上市会削弱同类旧产品的流行度;同样的,用户兴趣会随着时间推移而发生改变,使得用户兴趣特征在不同时间段下呈现出一定的差异。

图5中对同一个电影 v_j 而言,在上映之后不同时间 t 的平均分 μ 是不一样的,总体来看会随着时间的推进而逐渐降低;对同一用户 u_i 而言,在 t_1 时刻喜欢的是动作片,但在最近的一个月里,用户 u_i 转而对悬疑片感兴趣,希望能看到更多关于此领域的推荐。推荐系统需要有能力捕获到这些信息,并且及时的调整推荐模型。不幸的是,基于协同过滤的推荐算法不能快速识别用户兴趣的改变,仍会基于用户 u_i 的历史评分记录来推荐更多的动作电影,而不考虑时间因素的影响,这并不符合用户 u_i 的预期。

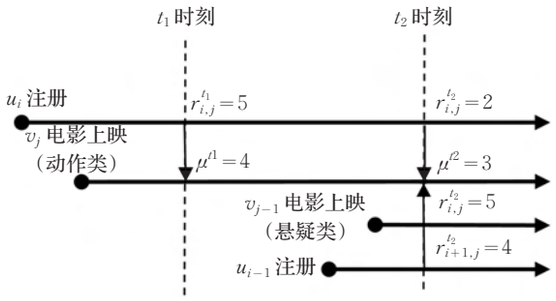

图5 用户兴趣漂移

Zilobaite[16]通过时间序列分析方法将概念漂移类型分为突变型概念漂移、渐变型概念漂移、重复型概念漂移、增量型概念漂移,图5对应的即为突变型概念漂移。为了应对用户兴趣漂移问题,一些工作[3][4]利用评分时间设置一个统一的时间衰减函数来反映用户兴趣的潜在变化,但这个方法只能降低旧数据在模型中的作用,应对增量型用户兴趣漂移,无法解决其他类型的用户兴趣漂移。

此时可以直接从用户隐向量入手,检测评分数据对于用户和物品隐向量的影响:

$$\begin{cases}\boldsymbol{u}_i^{t+1}\leftarrow\alpha^{-\sigma(\Delta u_i)}\cdot\boldsymbol{u}_i^t\\ \boldsymbol{v}_j^{t+1}\leftarrow\alpha^{-\sigma(\Delta v_j)}\cdot\boldsymbol{v}_j^t\end{cases}\tag{15}$$

通过式(15)控制用户和物品隐向量的更新,其中 $\alpha>1$, α 的取值控制着惩罚的力度,函数的取值范围在[0,1]。当标准差较大时,指数函数接近0,对用户和物品隐向量的惩罚力度较大,能够应对突变型、渐变型和重复型用户兴趣漂移;当标准差较小时,指数函数接近1,公式对稳定的用户和物品隐向量的影响可以忽略不计,能够应对增量型用户兴趣漂移。

4.2 异常评分检测与处理

在矩阵分解中引入和控制算法的遗忘性虽然能够有效应对用户兴趣漂移,但无法识别和避免异常评分对模型的影响。异常评分会造成式(15)中的惩罚力度偏大,造成已有用户兴趣模型的丢失。

假设 t 时刻到来的评分数据 r^t 符合用户的偏好,那么用户的隐向量 $\boldsymbol{u}_i$ 的变化应该是很小的。然而,如果观察到用户隐向量在使用 r^t 训练后发生了明显变化,表明 r^t 与该用户的偏好不一致,是异常评分。此种情况在真实场景中尤为常见,例如,用户为他人购买了礼物的时候,产生的评分对于用户本身的偏好来讲很可能是异常评分,应该被丢弃。

$$\Delta\boldsymbol{u}_i=\sum_{j=0}^{k}\left(\boldsymbol{u}_{i,j}^{t+1}-\boldsymbol{u}_{i,j}^{t}\right)^2\tag{16}$$

$$\Delta\boldsymbol{u}_i<E(\Delta\boldsymbol{u}_i)+\lambda\cdot\sigma(\Delta\boldsymbol{u}_i)\tag{17}$$

$$ratio_{\text{noise}}<\alpha\tag{18}$$

由式(16)、(17)可以看出,对于 t 时刻的评分 r^t ,首先计算出 $\Delta\boldsymbol{u}_i$,如果 $\Delta\boldsymbol{u}_i$ 满足 $\Delta\boldsymbol{u}_i$ 的平均值在标准差控制下的一定波动范围内(由 α 控制),那么保留用户隐向量更新值 $\boldsymbol{u}_{i,j}^{t+1}$,否则丢弃掉评分 r^t ,继续使用原来的 $\boldsymbol{u}_{i,j}^{t}$;当异常评分比率超过 α (本文采用0.3)时,判定为发生突变型用户兴趣漂移,此时不对“异常评分”进行处理。

以上两种策略不仅适用于本文提出的DS-SGD算法,可以适用于任何基于矩阵分解的算法。

5 实验及结果

本文在经典的 MovieLens-1m 数据集上进行了实验。MovieLens-1m 包含了6 040位用户对大约3 900部电影的100万份评分,并且保证每个用户至少对20部电影有评分。数据集的评分数据是1~5分的整数形式。本文并没有使用召回率作为评价指标,因为在实际应用中,由于不能准确得知系统没有推荐的商品中哪些是用户喜欢的,所以召回率很难应用于推荐系统的在线评估[17]。

实验1 DS-SGD算法加速比实验。为了比较DS-SGD算法在单机和分布式环境下模型训练的效率,使用加速比 $R_{\text{acc}}=T_s/T_c$ 来衡量,其中, T_s 表示在单个节点上的训练时间, T_c 表示在集群上训练的时间。

由图6加速比曲线可得,加速比并没有随计算节点

数的增加而线性增长(但基本接近线性增长)，这是由于在训练数据集大小固定的情况下，随着计算节点数的增加，通讯时间(包括参数交换时间和任务调度时间)在程序运行总时间中的比率 $R_{\text{comm}}=T_{\text{comm}}/T$ 持续上升。虽然加速比没有能够线性增长，但算法在面对大数据量的时候执行效率对比单机串行算法仍有显著提升，具有良好的扩展性。

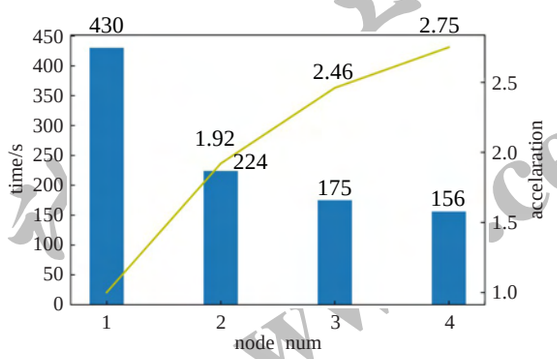

图6　训练时间与加速比

实验2　模型训练通讯开销对比。通过对比使用参数服务器训练的DSGD算法、使用点对点参数交换网络训练的DS-SGD算法和参数交换步长为5的DS-SGD算法，得到模型训练通讯开销统计如表1。

表1　通讯开销对比

类型	参数交换次数/万次	训练时间/s	*RMSE*
参数服务器	2 000	700	0.83
点对点参数交换	1 000	130	0.94
点对点参数交换(N=5)	431	95	0.95

通过表1可以看出，点对点参数交换方式能够将参数交换开销降低一半。在提升参数交换步长到5步时，能够进一步降低参数交换开销(取决于worker本地的训练集情况)。

实验3　模型拟合度和模型训练效率对比实验。通过固定集群节点数目和训练数据集，对比DS-SGD和DSGD算法的训练效果。两者采用相同的集群节点数目和训练数据集，隐向量因子数量为10，惩罚因子为0.1，默认步长0.05，默认最大迭代次数为10。

推荐的准确度是评价推荐系统的常用指标之一。本文采用的是推荐系统中常见的均方根误差 *RMSE* 来衡量，定义为：

$$RMSE=\frac{\sqrt{\sum_{u,i\in T}\left(r_{ui}-\hat{r}_{ui}\right)^2}}{|T|} \tag{19}$$

通过图7可以看出，在相同的训练时间下，DS-SGD的 *RMSE* 要远低于DGSD，证明本文提出的基于迭代数据流和点对点参数交换的训练方式，能够显著提升训练效率；当启用了遗忘性和异常评分检测与处理策略后(DS-SGD v2)，能够额外降低预测评分的误差，证明本文提出的方法能够在一定程度上应对用户兴趣漂移及降低异常评分对推荐结果的不利影响。当然，评估一个推荐系统是否理想远不止从以上方面来进行，本文仅针对分布式系统与单机系统，参数服务器与点对点参数交换，流式挖掘与批量挖掘的最显著差异，选用以上指标对算法进行评估。

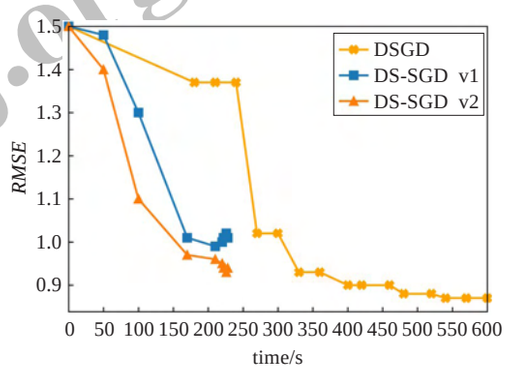

图7　*RMSE* 对比图

6　结语

针对分布式数据流环境中引入参数服务器带来的滞后梯度与掉队者问题，使用一致性哈希进行评分矩阵和用户隐向量矩阵划分，并通过点对点参数交换网络完成参数交换，无需再引入参数服务器；通过提升参数交换步长，进一步降低通讯开销。针对数据流环境下的用户兴趣漂移问题，本文在矩阵分解算法中引入遗忘策略和异常评分检测策略，通过检测用户和物品隐向量的标准差变动，应对突变型、渐变型、重复型、增量型用户兴趣漂移的影响；通过检测评分数据引起的用户隐向量更新波动，降低异常评分对推荐模型的不利影响。实验证明，对比传统的矩阵分解算法，本文提出的DS-SGD算法能够有效提升分布式数据流环境下推荐系统的时效性。

参考文献：

[1] Lee D D，Seung H S.Algorithms for non-negative matrix factorization[C]//Proceeedings of International Conference on Neural Information Processing Systems，2000：535-541.

[2] Chang S，Zhang Y，Chang Y，et al.Streaming recommender systems[C]//Proceeedings of International Conference on World Wide Web.International World Wide Web Conferences Steering Committee，2017：381-389.

[3] 孙光辉.基于时间效应和用户兴趣变化的改进推荐算法研究[D].北京：北京邮电大学，2014.

[4] 周利.基于遗忘函数和领域最近邻的网络营销推荐系统研究[D].长沙：湖南大学，2011.

(下转第148页)