推荐系统中的隐私保护矩阵分解算法研究

崔炜荣[1] 徐龙华[1] 杜承烈[2] 李 宝[1]

[1](安康学院电子与信息工程学院 陕西 安康 725000)

[2](西北工业大学计算机学院 陕西 西安 710072)

摘 要 针对协同过滤推荐系统中的用户数据和模型保护问题,提出一种隐私保护矩阵分解算法。该算法基于分布式架构,其梯度下降优化过程由服务器和各个用户相互协作完成。每轮迭代中,服务器仅从客户端得到物品隐藏因子向量梯度更新信息,从而有效保护了用户评分和推荐模型。基于多方安全求和的原理,在梯度更新过程中加入混淆机制,实现了对用户评分"存在性"的保护。开发系统原型并与现有方法进行实验对比,结果表明,该方法在保护用户隐私的同时能够提供更好的推荐准确度。

关键词 推荐系统 协同过滤 隐私保护 矩阵分解

中图分类号 TP391 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2021.05.051

PRIVACY-PRESERVING MATRIX FACTORIZATION ALGORITHM IN RECOMMENDER SYSTEM

Cui Weirong[1] Xu Longhua[1] Du Chenglie[2] Li Bao[1]

[1](*College of Electronic and Information Engineering, Ankang University, Ankang 725000, Shaanxi, China*)

[2](*School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, Shaanxi, China*)

Abstract This paper proposes a privacy-preserving matrix factorization algorithm for user data and model protection in collaborative filtering recommender system. The algorithm was based on a distributed architecture, and the gradient descent optimization process was completed by the server and various users collaboratively. In each iteration, the server only obtained the update information of the gradient of the hidden factor vector for items from the client, thereby effectively protecting the user score and the recommender model. Based on the principle of multi-party security summation, the method added a confusion mechanism in the process of gradient update, which realized the protection of the "existence" of user score. The system prototypes were developed and compared with existing methods. The results show that this method can provide better recommendation accuracy while protecting user privacy.

Keywords Recommender system Collaborative filtering Privacy protection Matrix factorization

0 引 言

当今社会,互联网中的各类信息都在以难以预知的速率爆炸式地增长,人类社会也随之从"信息匮乏"状态步入到"信息过载"状态。越来越多的用户在超过其处理能力的大量信息面前显得无所适从,很容易迷失在信息的海洋中。为了解决"信息过载"这一问题,使得用户可以在海量的数据中快速、准确地得到有用的信息,推荐系统应运而生。推荐系统能够根据用户的历史行为构建模型,预测用户对特定物品的态度并根据其预测向用户进行推荐。在推荐算法不断成熟和推荐系统被广泛应用的同时,其面临的隐私安全问题也日益凸显[1]。本质上,为了获得满意的推荐服务,用户需要向推荐系统提供个人信息。因此,相比其他网络信息服务,推荐系统更加具有天然的隐私不友好性。推荐系统面临的隐私安全风险主要在于:(1) 服务提供商可以直接利用用户所提交的数据获得用户的

收稿日期:2019-07-19。国家自然科学基金项目(61801005);陕西省教育厅科学研究计划项目(18JK0015)。**崔炜荣**,讲师,主研领域:网络与信息安全。**徐龙华**,副教授。**杜承烈**,教授。**李宝**,讲师。

个人信息,从而确认用户的身份、兴趣爱好等敏感信息;(2) 由于推荐系统所收集的数据的稀疏特性,利用推荐系统发布的聚合数据,攻击者可以有效推测出用户的敏感信息[1-2]。在如今人们越来越关心个人隐私安全的大背景下,如何在保证可用性的前提下引入合理的隐私保护机制也成为了有关推荐系统研究的热点问题。

当前,基于矩阵分解的协同过滤(Matrix Factorization Collaborative Filtering, MFCF)[3]是最为主流的推荐算法,该类算法从用户-物品评分矩阵中挖掘出潜在的用户特征向量及物品特征向量,并通过向量相似度预测用户对物品的评价。相比其他方法,MFCF 方法能够提供更高的预测准确度、更快的计算效率,以及更高的可扩展性。

本文针对 MFCF 推荐系统中的隐私保护问题展开研究,提出一种基于分布式计算架构和梯度混淆的隐私矩阵分解算法,称之为 DGCMF(Distributed Gradient Confusion Matrix Factorzation)。与现有的隐私保护 MFCF 方法相比,DGCMF 不仅可以防止用户评分和模型泄露,还可以防止用户"评价行为"是否存在这一信息的泄露,从而为用户隐私提供更强有力的保障。

1　相关工作

在先前的研究中,在推荐系统中增强隐私保护的方式主要分为以下三类:

1) 混淆和扰动。该方式通过在用户数据中增加伪造值以达到隐藏真实用户数据的目的。文献[4-5]中所提出的隐私保护推荐算法均基于该方式构建,使得用户可以对数据在本地进行加扰,从而对服务器隐藏真实的数据。但上述方法主要的问题在于:从安全性角度,上述加扰效果并未得到严格证明;从可用性角度上,上述方法在增强隐私保护的同时较大地牺牲了系统的推荐准确度。

2) 差分隐私。差分隐私保护的本质也是通过在计算过程中引入特定分布的噪声以达到保护原始数据的效果。但与简单的混淆和扰动不同,差分隐私保护可以提供能被度量和证明的隐私保障。文献[6]首次将差分隐私保护的概念引入了推荐系统,随后多项研究在其基础上展开[7-11]。

3) 同态加密。同态加密技术可以使得用户在不必解密的情况下对密文内容进行计算,被广泛应用于各类隐私保护场景中。文献[11-13]所设计的推荐系统均基于同态加密技术方案实现了隐私保护。使用同态加密最大的障碍在于其相对较高的计算开销。特别是当数据量较大时,效率和能耗成为限制其可用性的关键因素。

上述文献中的方法主要针对用户评分和推荐模型的保护。然而,它们共同缺乏的是对"存在性"的保护。也就是说,即便是无法得知用户的具体评分,服务器也依然可以在与用户的交互计算过程中知道用户对哪些物品进行了评价。利用这些信息,服务器仍然可以推测出用户的偏好、个人特征等隐私信息。与上述方法相比,本文方法除了能够实现对用户数据以及模型的保护外,也能够实现对用户评分"存在性"的保护。

与本文最为相似的工作来自文献[14],作者提出了名为 SDMF 的隐私保护矩阵分解方法。SDMF 通过在梯度更新过程中引入随机响应机制实现对用户评分、模型、存在性的保护。然而,这种方法在梯度下降优化过程中造成梯度更新损失,降低了所生成模型的预测准确性。与之相比,DGCMF 中的梯度混淆是无损的,因此既能够有效保护用户隐私,也能够保证系统的推荐准确性。

2　系统模型与问题描述

2.1　系统模型与假设

本文的系统模型如图 1 所示。令用户集合为 $\mathcal{U}=\{u_1,u_2,\cdots,u_m\}$,物品集合为 $V=\{v_1,v_2,\cdots,v_n\}$。每位用户都对 V 中的部分物品进行了评分。整个推荐系统根据用户们的不完全评分计算推荐模型,并为每一个用户对 V 中全部物品的评分生成预测值。基于此模型,假设:1) 用户评分保存在客户端,且每一个用户只知道自己的评分;2) 用户之间可以通过额外的安全信道进行点对点匿名通信。

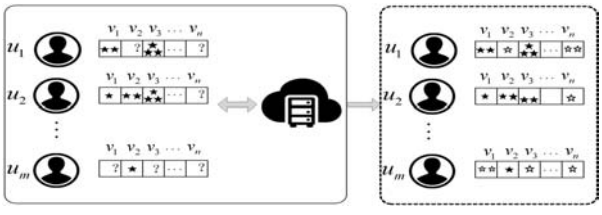


图 1　系统模型

2.2　隐私保护需求

本文算法应满足如下隐私保护需求:

1) 模型保护:令 $\mathcal{M}$ 为推荐系统生成的预测模型,则任意用户和服务器都无法获知完整的 $\mathcal{M}$,也无法通过所持有的部分信息推断出 $\mathcal{M}$ 的完整内容。

2) 评分保护:在模型的计算过程中,服务器无法获知每个用户的历史评分。

3）存在性保护：在模型的计算过程中，服务器无法获知用户是否对某个物品作出评分。

3 算法设计

3.1 总体设计

本文算法实现了分布式架构下的贝叶斯概率矩阵分解。算法架构如图 2 所示，主要符号及其含义见表 1。

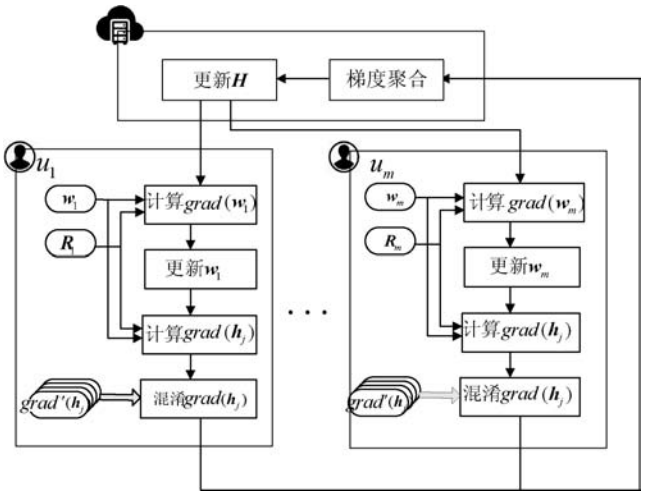


图 2 算法框图

表 1 主要符号及含义

符号	含义
$\mathcal{U},\mathcal{V}$	用户集合与物品集合
$\boldsymbol{W}:\mid\mathcal{U}\mid\times k,\boldsymbol{w}_i$	用户隐藏因子矩阵，$\boldsymbol{w}_i$ 为 $\boldsymbol{W}$ 的第 i 行
$\boldsymbol{H}:\mid\mathcal{V}\mid\times k,\boldsymbol{h}_j$	物品隐藏因子矩阵，$\boldsymbol{h}_j$ 为 $\boldsymbol{H}$ 的第 j 行
$\boldsymbol{R}:\mid\mathcal{U}\mid\times\mid\mathcal{V}\mid$	评分矩阵，r_{ij} 为用户 u_i 对物品 v_j 的评分值

令 $\mathcal{M}=(\boldsymbol{W},\boldsymbol{H})$ 为求解的目标预测模型，$\boldsymbol{w}_i\in\boldsymbol{W}$ 为用户 u_i 的隐藏因子向量，$\boldsymbol{h}_j\in\boldsymbol{H}$ 为物品 v_j 的隐藏因子向量。与一般的 MF 算法不同，本文算法的 $\boldsymbol{W}$ 和 $\boldsymbol{H}$ 分别由客户端和服务器端保存，并且每一个 $\boldsymbol{w}_i\in\boldsymbol{W}$ 都由其对应的用户 u_i 保存。也就是说，参与模型运算的服务器和各个用户都只知道模型的一部分，且无法通过该部分推测出整个模型，从而实现了模型保护。

计算过程由初始化和迭代优化两部分组成。客户端和服务器端分别对 $\boldsymbol{W}$ 和 $\boldsymbol{H}$ 完成初始化后便进入迭代优化环节。本文算法采用了 SGLD（Stochastic Gradient Langevin Dynamics）优化算法。SGLD 所增加的高斯噪声为系统引入了差分隐私保障，能够有效防止模型泄露。

在迭代优化环节的每一轮迭代过程中，每一个用户 u_i 首先从服务器端下载更新后的 $\boldsymbol{H}$，之后根据自己的评分数据 $\boldsymbol{r}_i$ 计算和 $\boldsymbol{w}_i$ 每一个 $\boldsymbol{h}_j(r_{ij}\neq\varnothing)$ 的更新梯度 $grad(\boldsymbol{w}_i)$ 和 $grad(\boldsymbol{h}_j)$。接着，u_i 利用 $grad(\boldsymbol{w}_i)$ 对 $\boldsymbol{w}_i$ 进行更新。对 $grad(\boldsymbol{h}_j)$ 进行梯度混淆后将其发送至服务器端。服务器收到每一个用户发来的梯度更新信息后，将其聚合并最终更新 $\boldsymbol{H}$。

由于每轮迭代中，服务器只收到用户发来的物品隐藏因子向量梯度信息，从而防止了用户评分泄露。梯度混淆的加入进一步隐藏了梯度信息中用户和被评分物品的联系，实现了“存在性”保护。

3.2 梯度更新算法

对于所求的模型 $\mathcal{M}=(\boldsymbol{W},\boldsymbol{H})$，若观察到的用户评分数据为 $\boldsymbol{R}$，则矩阵分解问题可以转变为求解参数 $\boldsymbol{W}$、$\boldsymbol{H}$ 的最大后验概率分布问题，即：

$$p(\boldsymbol{W},\boldsymbol{H}\mid\boldsymbol{R},\lambda_r,\boldsymbol{\Lambda}_w,\boldsymbol{\Lambda}_h)\propto p(\boldsymbol{R}\mid\boldsymbol{W},\boldsymbol{H},\lambda_r)p(\boldsymbol{W}\mid\boldsymbol{\Lambda}_w)p(\boldsymbol{H}\mid\boldsymbol{\Lambda}_h)\quad(1)$$

式中：λ_r 为全局正则化参数，$\boldsymbol{\Lambda}_w$ 和 $\boldsymbol{\Lambda}_h$ 是基于 Gamma 分布生成的对角矩阵，用于隐藏因子向量 $\boldsymbol{w}_i$ 和 $\boldsymbol{h}_j$ 的正则化处理。采用正态分布假设，则式（1）的对数形式为：

$$\begin{aligned}F(\boldsymbol{W},\boldsymbol{H})&=\ln p(\boldsymbol{R},\lambda_r,\boldsymbol{\Lambda}_w,\boldsymbol{\Lambda}_h)=\\&\ln p(\boldsymbol{R}\mid\boldsymbol{W},\boldsymbol{H},\lambda_r)+\ln p(\boldsymbol{W}\mid\Lambda_w)+\\&\ln p(\boldsymbol{H}\mid\boldsymbol{\Lambda}_h)+C=\\&\ln N(\boldsymbol{R}\mid\boldsymbol{W}\boldsymbol{H}^T,\lambda_r^{-1})+\ln N(\boldsymbol{W}\mid 0,\boldsymbol{\Lambda}_w^{-1})+\\&\ln N(\boldsymbol{H}\mid 0,\boldsymbol{\Lambda}_h^{-1})+C\end{aligned}\quad(2)$$

式中：C 为常数；N 表示正态分布。为了求得 $\boldsymbol{W}$ 和 $\boldsymbol{H}$ 的最大后验估计，本文采用 SGLD 优化算法进行迭代求解。具体而言，在第 t 次迭代中：

$$\hat{\nabla}_{w_i}F(\boldsymbol{w}_i,\boldsymbol{h}_j)=-\boldsymbol{e}_{ij}\boldsymbol{w}_i+\boldsymbol{h}_j\boldsymbol{\Lambda}_w\quad(3)$$

$$\hat{\nabla}_{h_i}F(\boldsymbol{w}_i,\boldsymbol{h}_j)=-\boldsymbol{e}_{ij}\boldsymbol{h}_j+\boldsymbol{h}_j\boldsymbol{\Lambda}_h\quad(4)$$

$$\eta_t=\frac{\eta_0}{t^\gamma}\quad(5)$$

$$\boldsymbol{w}_i=\boldsymbol{w}_i+\eta_t\hat{\nabla}_{w_i}F(\boldsymbol{w}_i,\boldsymbol{h}_j)+\mathcal{N}(0,\eta_t,\boldsymbol{I})\quad(6)$$

$$\boldsymbol{h}_j=\boldsymbol{h}_j+\eta_t\hat{\nabla}_{h_j}F(\boldsymbol{w}_i,\boldsymbol{h}_j)+\mathcal{N}(0,\eta_t,\boldsymbol{I})\quad(7)$$

式中：η_0 为初始学习速率；t 为迭代次数；γ 为衰减因子；$\boldsymbol{I}$ 为单位矩阵；$\boldsymbol{e}_{ij}=\boldsymbol{r}_{ij}-\boldsymbol{w}_i\boldsymbol{h}_j^T$。令 $grad_{u_i}(\boldsymbol{h}_j)$ 表示由用户 u_i 根据 $\boldsymbol{w}_i$ 所计算出的 $\boldsymbol{h}_j$ 的加扰梯度更新信息，则：

$$grad_{u_i}(\boldsymbol{h}_j)=\eta_t\hat{\nabla}_{w_i}F(\boldsymbol{w}_i,\boldsymbol{h}_j)+\mathcal{N}(0,\eta_t,\boldsymbol{I})\quad(8)$$

用户 u_i 发送 $grad_{u_i}(\boldsymbol{h}_j)$ 至服务器。由于服务器仅

得知梯度信息,因此防止了用户隐藏因子向量的泄露。

3.3　梯度混淆算法

采用上述架构,虽然在迭代优化过程中服务器仅能从客户端得到 $\boldsymbol{H}$ 的加扰梯度更新信息且无法从中推断出用户的隐藏因子向量,然而服务器依然可以根据这些信息获知用户对哪些物品进行了评价。例如当服务器收到来自用户 u_i 的梯度更新信息 $grad_{u_i}(\boldsymbol{h}_j)$ 时,根据 $\boldsymbol{h}_j$ 和物品 $\boldsymbol{v}_j$ 的对应关系,可知用户 u_i 一定对 v_j 进行了评价。为了避免这种"存在性"用户信息泄露,本文基于安全求和思想设计了梯度混淆过程。

在梯度混淆的过程中,对于每一个计算出的梯度更新信息 $grad_u(\boldsymbol{h}_j)$,用户 u 以一定概率选择将其值发送给随机选取的另一用户,并随后将其置零。收到 $grad_u(\boldsymbol{h}_j)$ 的用户 u' 将其与自身的 $grad_{u'}(\boldsymbol{h}_j)$ 相加。举例而言,假设用户 u 计算出的梯度值分别为 $grad_u(\boldsymbol{h}_{j_1}),grad_u(\boldsymbol{h}_{j_2}),\cdots,grad_u(\boldsymbol{h}_{j_q})$,$SG$ 为空集,则 u 依据算法 1 实施混淆。

算法 1　梯度混淆

for $t = j_1$ to j_q
　据概率 f 选取 $grad_u(\boldsymbol{h}_t)$
　if $grad_u(\boldsymbol{h}_t)$ 被选中 then
　　$SG \leftarrow SG \cup \{grad_u(\boldsymbol{h}_t)\}$
　　$grad_u(\boldsymbol{h}_t) \leftarrow 0$
　end if
end for

经过上述的梯度混淆处理后,u 将筛选出来的梯度更新值集合 SG 发送给随机选取的用户 u'。收到 SG 后,对于每一个 $grad_u(\boldsymbol{h}_j) \in SG$,$u'$ 将其与自身的对应梯度更新值进行合并,即计算混淆梯度:

$$\widetilde{grad}_{u'}(\boldsymbol{h}_j) = grad_{u'}(\boldsymbol{h}_j) + grad_u(\boldsymbol{h}_j) \tag{9}$$

图 3 描述了这一个过程。其中行向量表示用户在第 t 次迭代过程中所计算并分解的梯度更新信息。值为 0 表示用户并未对 $\boldsymbol{h}_j$ 进行过单类评价。曲线箭头表示以一定的概率选取并发送。以 u_2 为例,在该轮迭代中,其根据概率 f_{u_2} 选取了梯度更新值 $grad_{u_2}(\boldsymbol{h}_1)$,并将其发送至随机选取的用户 u_1。同时,u_1 根据概率 f_{u_1} 选取了梯度更新值 $grad_{u_1}(\boldsymbol{h}_3)$,将其发送给 u_2。u_3 根据概率 f_{u_3} 选取了梯度更新值 $grad_{u_3}(\boldsymbol{h}_2)$,也将其发送给了 u_2。最终,u_2 生成混淆梯度 $(0,grad_{u_3}(\boldsymbol{h}_2),grad_{u_2}(\boldsymbol{h}_2)+grad_{u_1}(\boldsymbol{h}_3),0)$。每轮迭代完成后,$u_2$ 将混淆后的梯度更新值发送至服务器。

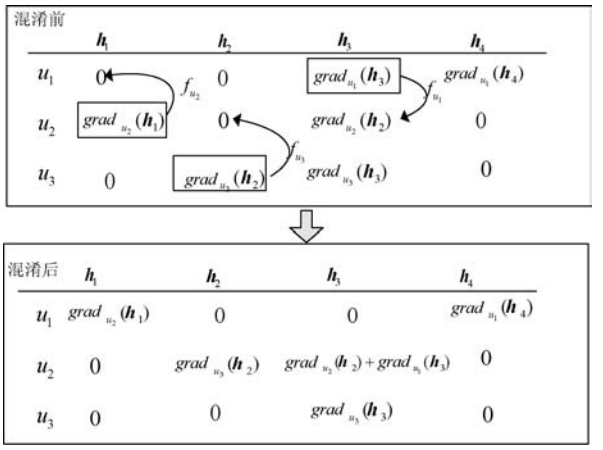


图 3　梯度混淆过程

在每轮迭代末,服务器对所收到的来自各个用户端的混淆梯度更新值进行求和平均,并更新对应的 $\boldsymbol{h}_j$。以图 3 为例,假设针对物品隐藏因子向量 $\boldsymbol{h}_3$,服务器收到分别来自 u_1、u_2、u_3 的混淆梯度 $\widetilde{grad}_{u_1}(\boldsymbol{h}_3)$、$\widetilde{grad}_{u_2}(\boldsymbol{h}_3)$、$\widetilde{grad}_{u_3}(\boldsymbol{h}_3)$,则服务器经过以下计算进行梯度聚合和参数更新:

$$\widetilde{grad}(\boldsymbol{h}_3) = \frac{1}{3}\sum_{i=1}^{3}\widetilde{grad}_{u_i}(\boldsymbol{h}_3) \tag{10}$$

$$\boldsymbol{h}_3 = \boldsymbol{h}_3 + grad(\boldsymbol{h}_3) \tag{11}$$

由于 $\frac{1}{3}\sum_{i=1}^{3}\widetilde{grad}_{u_i}(\boldsymbol{h}_3) = \frac{1}{3}\sum_{i=1}^{3}grad_{u_i}(\boldsymbol{h}_3)$,所以用户端的梯度混淆过程汇总没有增加额外的噪声,也没有减少应有的信息量,所以称之为无损的。

3.4　算法流程

基于分布式架构,本文算法由客户端算法 client(u,t) 和服务器端算法 server 两部分组成,流程分别如算法 2 和算法 3 所示。

算法 2　client(u,t)

输入:$\boldsymbol{R}_i,\eta_0,\boldsymbol{r},\boldsymbol{\Lambda}_w,\boldsymbol{\Lambda}_h$。
输出:混淆后的梯度更新值 $\overline{GRAD}$。

1. if $t = 0$ then 初始化 w_i
2. if $t > 0$ then
3. 　从服务器下载 $\boldsymbol{H}$
4. 　$\overline{\boldsymbol{w}} \leftarrow 0$
5. 　$count_{\nabla u} \leftarrow 0$ //$count_{\nabla u}$ 用来对值不为 0 的 R_{ij} 进行计数
6. 　$\overline{GRAD} \leftarrow 0, \overline{SG} \leftarrow 0$
7. 　$\eta_t = \frac{\eta_0}{t^r}$
8. 　for $j = 1$ to $|\mathcal{V}|$:
9. 　　if $R \neq 0$ then
10. 　　$\overline{\boldsymbol{w}}_i = \overline{\boldsymbol{w}}_i + \eta_t \hat{\nabla}_{\boldsymbol{w}_i} F(\boldsymbol{w}_i,\boldsymbol{h}_j) + \mathcal{N}(0,\eta_t,\boldsymbol{I})$
11. 　　$count_{\nabla u} \leftarrow count_{\nabla u} + 1$
12. 　$\overline{GRAD}[j] = \overline{GRAD}[j] + \eta_t \hat{\nabla}_{\boldsymbol{h}_j} F(\boldsymbol{w}_i,\boldsymbol{h}_j) + \mathcal{N}(0,\eta_t,\boldsymbol{I})$

13. end if
14. end for
15. $\boldsymbol{w}_i = \boldsymbol{w}_i + \frac{\bar{\boldsymbol{w}}}{count_{\nabla u}}$
16. 生成概率参数 $f \in [0,1]$
17. for $j = 1$ to $|\mathcal{V}|$:
18. if $\overline{GRAD}[j] \neq 0$ then
19. 根据概率 f 选取 $\overline{GRAD}[j]$
20. if $\overline{GRAD}[j]$ 被选中 then
21. $\overline{SG}[j] \leftarrow \overline{GRAD}[j], \overline{GRAD}[j] \leftarrow 0$
22. end if
23. end if
24. end for
25. 随机选取一个用户,向其发送 $(u, \overline{SG})$
26. for 每一个收到的来自其他用户的 SG:
27. for $j = 1$ to $|\mathcal{V}|$:
28. $\overline{GRAD}[j] \leftarrow \overline{GRAD}[j] + \overline{SG}[j]$
29. end for
30. end for
31. 发送 $(u, \overline{GRAD})$ 至服务器
32. end if

算法 2 中,用户首先更新了学习速率(第 7 行);接着根据评分向量 R_{ij}、$\boldsymbol{w}_i$ 和 $\boldsymbol{H}$ 计算出 $\boldsymbol{w}_i$ 的更新梯度和被评分物品对应的 $\boldsymbol{h}_j$ 的更新梯度(第 8 至 14 行);之后更新 $\boldsymbol{w}_i$(第 15 行)并进行梯度混淆(第 16 至 29 行);最后将经过混淆的梯度值发送给服务器(第 31 行)。

算法 3 server

输入:用户集合 $\mathcal{U}$ 及物品集合 $\mathcal{V}$。

输出:物品隐藏因子矩阵 $\boldsymbol{H}$。

1. 初始化 $\boldsymbol{H}$
2. $t \leftarrow 0$
3. for $i = 1$: $|\mathcal{U}|$
4. 远程调用 client(u_i, t)
5. end for
6. $t \leftarrow 1$
7. while 没有收敛 do:
8. $\bar{\boldsymbol{H}} \leftarrow 0$
9. $\bar{\boldsymbol{C}} \leftarrow 0$ // $\bar{\boldsymbol{C}}$ 用于对收到的用户梯度更新值进行计数
10. for $i = 1$: $|\mathcal{U}|$
11. 远程调用 client(u_i, t)
12. end for
13. while Ture do:
14. 等待直到接收到 $(u, \overline{GRAD})$
15. for $j = 1$ to $|\mathcal{V}|$:
16. $\bar{\boldsymbol{H}}[j] = \bar{\boldsymbol{H}}[j] + \overline{GRAD}[j]$
17. $\bar{\boldsymbol{C}}[j] \leftarrow \bar{\boldsymbol{C}}[j] + 1$
18. end for
19. if 收到所有用户发来的 $(u, \overline{GRAD})$ then
20. break
21. end if
21. end while
22. for $j = 1$ to $|\mathcal{V}|$
23. $\bar{\boldsymbol{H}}[j] \leftarrow \bar{\boldsymbol{H}}[j] / \bar{\boldsymbol{C}}[j]$
24. end for
25. $\boldsymbol{H} \leftarrow \boldsymbol{H} + \bar{\boldsymbol{H}}$
26. $t \leftarrow t + 1$
27. end while

算法 3 中,服务器在初始化 $\boldsymbol{H}$ 之后,远程启动每个客户端的初始化过程(第 1 至 5 行)。之后进入迭代优化环节。在每轮迭代中,首先下发 $\boldsymbol{H}$ 并调用客户端的梯度更新方法(第 10 至 12 行)。待全部用户完成更新计算并回传物品隐藏因子向量梯度更新值后,对其进行聚合并完成 $\boldsymbol{H}$ 的更新(第 13 至 25 行)。

4 安全性分析

根据 2.2 节所述,本文算法应当满足模型、评分和存在性三方面的隐私保护需求。

4.1 模型保护

本文算法采用了分布式架构,$\boldsymbol{W}$ 中的每一个用户隐藏因子向量都保存在对应的客户端。在模型的训练过程中,服务器在每轮迭代中仅能从客户端接收到 $\widetilde{grad}_u(\boldsymbol{h})$。此外,根据文献[15]的证明,SGLD 优化算法中的加扰机制使其能够提供差分隐私保障。因此,服务器无法从 $\widetilde{grad}_u(\boldsymbol{h})$ 中推导出其对应的用户隐藏因子向量 $\boldsymbol{w}$。对于每个用户而言,虽然其能够获得 $\boldsymbol{H}$ 的完整内容,但由于其无法获知保存在其他客户端的 $\boldsymbol{w}$,因此无法推导出 $\boldsymbol{W}$ 的完整内容。根据以上分析,令 $\mathcal{M}$ 为推荐系统生成的预测模型,则任意用户和服务器都无法获知完整的 $\mathcal{M}$,也无法通过所持有的部分信息推断出 $\mathcal{M}$ 的完整内容。

4.2 评分保护

与采用集中式架构的 MF 算法不同,在本文中,作为最为重要的隐私数据的用户历史评分并没有被服务器所收集,而是分别被保存在各个客户端。在迭代计算过程中,用户 u_i 的历史评分 $\boldsymbol{r}_i$ 只参与了预测误差 $\boldsymbol{e}_{ij}$ 的计算,且服务器无法从接收到的混淆梯度更新信息中推导出 $\boldsymbol{r}_i$。根据以上分析,在模型的计算过程中,服

务器无法获知每个用户的历史评分。

4.3　存在性保护

在本文中,梯度混淆是实现用户评分存在性保护的关键。假设在第 t 轮迭代中,用户 u_i 计算得出 $\boldsymbol{h}_j$ 的原始梯度更新信息为 $grad_{u_i}(\boldsymbol{h}_j)$,经过混淆后的梯度更新信息为 $\widetilde{grad}_{u_i}(\boldsymbol{h}_j)$,则 $\widetilde{grad}_{u_i}(\boldsymbol{h}_j) = grad_{u_i}(\boldsymbol{h}_j) + K$。$K$ 是一个随机变量,其值既有可能为 0,也有可能为来自其他若干个用户的梯度更新值之和。显然,服务器无法根据 $\widetilde{grad}_{u_i}(\boldsymbol{h}_j)$ 的值判定 $grad_{u_i}(\boldsymbol{h}_j)$ 是否为 0。换句话说,服务器无法根据 $\widetilde{grad}_{u_i}(\boldsymbol{h}_j)$ 判定用户 u_i 是否对 $\boldsymbol{h}_j$ 进行过评价。以图3 中的 u_2 为例,$grad_{u_2}(\boldsymbol{h}_1) \neq 0$,$grad_{u_2}(\boldsymbol{h}_2) = 0$,经过混淆后,$\widetilde{grad}_{u_2}(\boldsymbol{h}_1) = 0$,$\widetilde{grad}_{u_2}(\boldsymbol{h}_2) = grad_{u_3}(\boldsymbol{h}_3)$。因此,服务器既不能根据 $\widetilde{grad}_{u_2}(\boldsymbol{h}_1) = 0$ 判定 u_2 没有对 $\boldsymbol{h}_1$ 进行评价,也不能根据 $\widetilde{grad}_{u_2}(\boldsymbol{h}_2) = grad_{u_3}(\boldsymbol{h}_3)$ 判定 u_2 对 $\boldsymbol{h}_2$ 作出过评价。

此外,由于每一个用户在每一轮迭代末都依据实时随机生成的选择概率 f 进行了混淆(算法 2 第 16 行)。因此服务器无法从所收集的用户梯度更新值序列中推测出用户是否对某一物品进行过评价,从而实现了存在性保护。

5　实　验

在算法 2 和算法 3 的基础上,使用 Python 开发了系统原型。为了评估系统的可用性,将其与文献[15]中提出的 SDMF 方法进行了对比。

5.1　原型构造方法

协议原型基于 Python3.6 + TensorFlow2.0 开发,主要由 Server 类和 User 类组成,Sever 类是对服务器的模拟,其中主要包含的属性有物品隐藏因子矩阵 $\boldsymbol{H}$,主要包含的方法有聚合梯度更新(针对 $\boldsymbol{H}$)。User 类是对用户的模拟,其中主要包含的属性有用户 ID、评分向量 $\boldsymbol{r}$ 和隐藏因子向量 $\boldsymbol{w}$,主要包含的方法有梯度更新和梯度混淆方法。梯度更新方法中调用了 TensorFlow 框架 Stochastic Gradient Langevin Dynamics 模块实现 SGLD 优化过程。

模拟运行的每一轮迭代中,服务器实例依照算法 3 依次调用每一个用户实例的梯度更新方法。每一个用户实例依照算法 2 进行 $\boldsymbol{w}$ 梯度更新,并将经过混淆后的 $\boldsymbol{h}$ 梯度传回给服务器实例,由服务器实例进行聚合并调用梯度更新方法更新 $\boldsymbol{H}$。

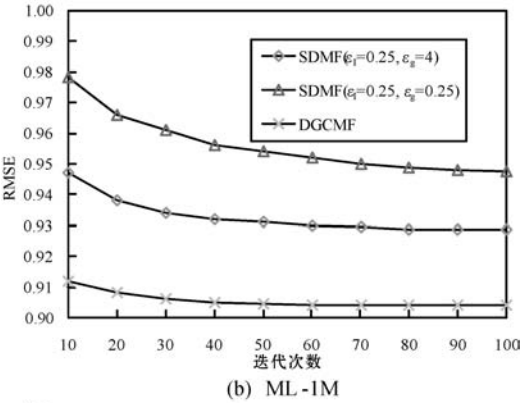
5.2　数据集及实验方法

使用了 3 个公共评分数据集进行实验,具体包括:两个 MovieLens 数据集(ML-100K 和 ML-1M),一个 Netflix 数据子集(包含 10 000 个用户和 5 000 个物品)[16]针对上述每一个数据集,将其随机划分为占比 80% 的训练集和占比 20% 的测试集,训练集中的 20% 作为验证集。具体的超参数设置为:$k = 50$,$r = 0.6$,$(\boldsymbol{\Lambda}_w, \boldsymbol{\Lambda}_h) \sim \text{Gamma}(1, 100)$。初始学习速率 η_0 分别为 5×10^{-6}(ML-100K)、5×10^{-7}(ML-1M)、5×10^{-7}(Netflix)。对于 SDMF,选取了($\varepsilon_1 = 0.25, \varepsilon_g = 4$)和($\varepsilon_1 = 0.25, \varepsilon_g = 1$)作为两组差分隐私配置参数。

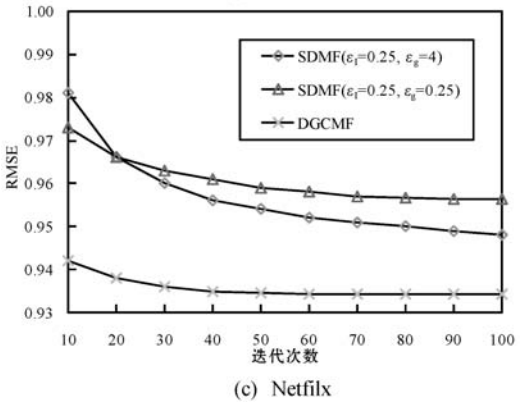
5.3　结果及分析

在 ML-100K、ML-1M 和 Netfilx 数据集上的学习曲线如图 4 所示。

(a) ML-100K



(b) ML-1M



(c) Netfilx

图 4　实验结果

可以清楚地看到,对于 SDMF 而言,参数 ε 值越

小,则系统可以提供越强的差分隐私保障,但其预测准确度就越低。此外,尽管 SDMF 和 DGCMF 都可以实现模型保护、用户评分保护和存在性保护,但可以看出,DGCMF 所生成的模型预测准确性明显高于 SDMF 所生成的模型。这是由于本文所设计的DGCMF和SDMF 在存在性保护方面采取了不同的混淆机制。SDMF 采用基于随机响应的方式模糊用户的评价行为。在每轮迭代中,用户根据一定的概率设定,选择是否计算某一已打分物品的隐藏因子向量梯度。此外,用户还有可能伪造未打分物品的隐藏因子向量梯度。这种混淆方式本质上是有损的,因此会对系统的准确性造成负影响。而DGCMF所采用的方式能够在隐藏"存在性"的同时确保其混淆行为是无损的,从而在实现隐私保护的同时最大限度地保障推荐准确度。

6 结 语

针对协同过滤推荐系统中的隐私保护问题,本文设计 DGCMF 算法。在 DGCMF 中,矩阵分解过程由各个客户端和服务器共同配合完成。由于服务器只能收到来自客户端的物品隐藏因子向量梯度,从而避免了用户评分和模型的泄露。此外,客户端对所要上传给服务器的梯度更新信息进行了混淆,实现了对用户评分"存在性"的保护。与现有方法相比,DGCMF 的梯度混淆过程不会造成梯度更新损失,在实现隐私保护的同时能够提供更好的推荐准确度。今后,将针对如何进一步提高模型的准确度和生成效率展开研究,以期在安全性和效率两方面达到更优的权衡。

参 考 文 献

[1] Knijnenburg B P, Berkovsky S. Privacy for recommender systems: Tutorial abstract[C]//Eleventh ACM Conference on Recommender Systems, 2017: 394 - 395.

[2] Ricci F, Rokach L, Shapira B. Recommender systems: Introduction and challenges[M]//Recommender Systems Handbook. Springer, 2015: 1 - 34.

[3] Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems[J]. Computer, 2009(8): 30 - 37.

[4] Polat H, Du W. SVD-based collaborative filtering with privacy[C]//2005 ACM Symposium on Applied Computing, 2005: 791 - 795.

[5] Zhang S, Ford J, Makedon F. Deriving private information from randomly perturbed ratings[C]//2006 SIAM International Conference on Data Mining, 2006: 59 - 69.

[6] McSherry F, Mironov I. Differentially private recommender systems: Building privacy into the netflix prize contenders[C]//15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009: 627 - 636.

[7] Hua J, Xia C, Zhong S. Differentially private matrix factorization[C]//24th International Conference on Artificial Intelligence, 2015: 1763 - 1770.

[8] Berlioz A, Friedman A, Kaafar M A, et al. Applying differential privacy to matrix factorization[C]//9th ACM Conference on Recommender Systems, 2015: 107 - 114.

[9] Machanavajjhala A, Korolova A, Sarma A D. Personalized social recommendations: Accurate or private[J]. Proceedings of the VLDB Endowment, 2011, 4(7): 440 - 450.

[10] Xin Y, Jaakkola T. Controlling privacy in recommender systems[C]//Advances in Neural Information Processing Systems, 2014: 2618 - 2626.

[11] Erkin Z, Beye M, Veugen T, et al. Privacy-preserving content-based recommendations through homomorphic encryption[C]//33rd WIC Symposium on Information Theory in the Benelux, 2012.

[12] Badsha S, Yi X, Khalil I. A practical privacy-preserving recommender system[J]. Data Science and Engineering, 2016, 1(3): 161 - 177.

[13] Kikuchi H, Kizawa H, Tada M. Privacy-preserving collaborative filtering schemes[C]//2009 International Conference on Availability, Reliability and Security, 2009: 911 - 916.

[14] Jiang J Y, Li C T, Lin S D. Towards a more reliable privacy-preserving recommender system[EB]. arXiv: 1711. 07638, 2017.

[15] Liu Z, Wang Y X, Smola A. Fast differentially private matrix factorization[C]//9th ACM Conference on Recommender Systems, 2015: 171 - 178.

[16] Bennett J, Lanning S. Netflix: The Netflix prize[C]//KDD Cup and Workshop in Conjunction with KDD, 2007.

(上接第 255 页)

[7] 魏政磊, 赵辉, 李牧东, 等. 控制参数值非线性调整策略的灰狼优化算法[J]. 空军工程大学学报(自然科学版), 2016, 17(3): 68 - 72.

[8] 胡小平, 曹敬. 改进灰狼优化算法在 WSN 节点部署中的应用[J]. 传感技术学报, 2018, 31(5): 753 - 758.

[9] 段亚青, 王华倩, 乔学工. 基于测距和灰狼优化的无线传感器网络定位算法[J]. 传感技术学报, 2018, 31(12): 1894 - 1899.

[10] Gai W, Qu C, Liu J, et al. An improved grey wolf algorithm for global optimization[C]//2018 Chinese Control and Decision Conference(CCDC), 2018: 2494 - 2498.

[11] 滕志军, 吕金玲, 郭力文, 等. 一种基于 Tent 映射的混合灰狼优化的改进算法[J]. 哈尔滨工业大学学报, 2018, 50(11): 40 - 49.

[12] 宁爱平, 张雪英. 人工蜂群算法的收敛性分析[J]. 控制与决策, 2013, 28(10): 1554 - 1558.