曲阜師範大學

本科生毕业论文（设计）

题　　目	面向联邦推荐的对抗攻击算法的设计与实现
姓　　名	张中健　　学号　2019414116
院　　系	网络空间安全学院
专　　业	软件工程
指导教师	周子力　　职称　副教授

2023 年 5 月 26 日

曲阜师范大学教务处制

目 录

面向联邦推荐的对抗攻击算法的设计与实现

软件工程专业学生　　张中健

指导教师　　周子力

摘要：推荐系统旨在为用户推荐物品，但集中式的数据存储方式增加了隐私泄露风险。为此，基于联邦学习（FL）的联邦推荐（FR）应运而生。与 FL 不同，FR 具有独特的稀疏聚合机制，即每个物品的嵌入信息只能被部分客户端更新。最近，拜占庭攻击备受关注，因为恶意客户端能够发送任意更新，导致模型安全性问题。然而，现有的拜占庭工作忽视了 FR 的独特稀疏聚合机制，因此首次尝试从稀疏聚合的角度对 FR 上的拜占庭攻击进行研究，重新定义稀疏聚合下的拜占庭鲁棒性，并设计一系列有效的攻击策略，命名为 Spattack。大量实验结果表明，Spattack 可以有效地防止模型收敛，甚至在少数攻击者的情况下破坏防御，这也引起了对保护 FR 系统的警惕。

关键词：联邦学习　推荐系统　拜占庭鲁棒性　对抗攻击

Design and Implementation of Adversarial Attack Algorithms for Federated Recommendation

Student majoring in software engineering　　Zhongjian Zhang

Tutor　　Zili Zhou

Abstract: Recommendation systems are designed to recommend items to users, but the centralized data storage method increases the risk of privacy breaches. Thus, federated recommendation (FR) based on federated learning (FL) emerged. Unlike FL, FR has a unique sparse aggregation mechanism where the embedding information of each item can only be updated by some client devices. Recently, Byzantine attacks have received attention because malicious clients can send arbitrary updates, leading to model security issues. However, existing Byzantine work ignores the unique sparse aggregation in FR. Therefore, this study attempted to investigate Byzantine attacks on FR from the perspective of sparse aggregation for the first time, redefining Byzantine robustness under sparse aggregation and designing a series of effective attack strategies, called Spattack. A large number of experimental results show that Spattack can effectively prevent model convergence and even break down defenses in the case of a small number of attackers, which has also raised concerns about protecting FR systems.

Key words: Federated learning; Recommendation system; Byzantine robustness; Adversarial attack

1 引言

1.1 研究背景与意义

作为缓解信息过载的基本途径，推荐系统广泛应用于电子商务[30, 36, 44]、媒体[35, 37, 48]和社交网络[10, 16]等领域，旨在为用户提供可能感兴趣的物品。传统的推荐系统虽然在准确性方面取得了显著成功，但需要集中存储用户个人数据进行训练，增加了用户隐私泄露的风险。近年来，联邦学习（FL）[24]作为一种保护隐私的范式已成功地应用于推荐领域。在联邦推荐（FR）[2, 7, 18, 20, 32, 45]中，全局物品嵌入被上传到中央服务器进行聚合，同时每个用户的交互数据和隐私特征在本地客户端上保留。基于这种训练范式，本地数据隐私得到有效保护。

与一般的联邦学习系统不同，联邦推荐具有独特的稀疏聚合机制。如图 1-1（a）所示，在一般的联邦学习中，模型参数的每个元素（每个圆）都可以被所有（n个）客户端更新，这种更新方式称为密集聚合。而在联邦推荐中，用户和物品之间的交互通常是稀疏的[23]，导致每个物品的嵌入只能由部分客户端更新，以图 1-1（b）中用户u_n为例，客户端n只能为其交互的物品v_1, v_3产生并发送实质性的梯度$\{\nabla v_1^n, \nabla v_3^n\}$，对于剩余的物品，$u_n$的更新为零向量或为空，这种更新方式称为稀疏聚合。

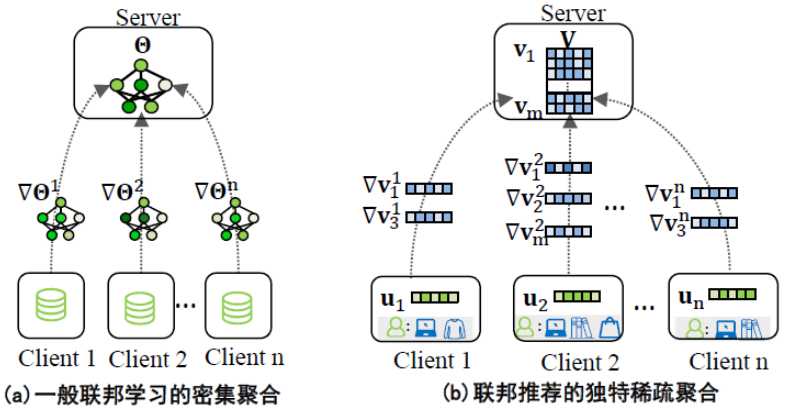


图 1-1 一般联邦学习的密集聚合和联邦推荐的独特稀疏聚合之间的比较

到目前为止，联邦推荐将推荐系统扩展到了对隐私敏感的应用场景，在不收集用户隐私数据的情况下取得了良好的性能。尽管如此，模型安全作为一个基本原则受到了越来越多的关注。然而，现有的拜占庭工作不能直接应用于联邦推荐中，因为它们主要关注一般联邦学习中的密集聚合[5, 25, 28, 42, 43]。此外，虽然已经出现了一些针对联邦推荐的攻击[29, 39, 46, 47]，但它们也都忽略了在稀疏聚合下如何分析联邦推荐的鲁棒性。

1.2 国内外研究现状

1.2.1 联邦推荐系统

联邦学习旨在通过分布式数据协作训练共享模型保护隐私[24]。因此，联邦推荐确保用户的历史数据仅在本地存储，并仅将中间数据上传到服务器进行协作训练。在这个过程中，用户的评分行为（交互物品或评分分数的集合）是私有信息。MuhammadAmmadud-din 等人[2]首次提出了一种联邦协同过滤框架用于隐私保护推荐，用户嵌入被本地存储和更新，而物品嵌入的梯度被上传到服务器进行聚合。此外，为了更好的隐私保护，Chai 等人[7]应用了同态加密；Lin 等人[18]随机采样假评分；Ying 等人[45]通过使用秘密共享而不是同态加密进一步提高了效率。最近，基于图神经网络的联邦推荐也开始出现[19, 20, 38]，进一步将高阶用户-物品交互纳入本地训练数据。总体而言，大多数现有的联邦推荐系统都遵循梯度上传到服务器进行聚合的范式。因此，

它们都是稀疏聚合，这导致每个物品嵌入只能由部分客户端更新，进而导致每个物品的鲁棒性不同。本文首次从稀疏聚合的角度揭示了这种独特的脆弱性。

1.2.2 联邦学习的鲁棒性

近年来，随着联邦学习的流行，其安全性也引起了越来越多的关注。[21,22]提出了大量针对 FL 的攻击策略，其中拜占庭攻击是比较常见的攻击策略[28]。这些研究表明，在经典的Mean(·)聚合器中，单个恶意客户端的任何更新都很容易扭曲整个模型的结果。因此，基于统计的拜占庭防御方法被提出[5,12,25,26,28,40,42,43]，旨在过滤掉拜占庭更新并保证联邦学习的收敛性。尽管这个问题在 FL 中得到了很好的研究，但现有针对 FL 的拜占庭攻击和防御方法都是基于密集聚合机制定义的，因此不能应用于 FR 中的稀疏聚合场景。最近，一些攻击被提出用于联邦推荐，其中[29,47]将目标集中在增加目标物品的曝光机会上，而[39]则通过不当的正/负样本间接操纵模型参数。此外，还有一种上传有毒梯度的攻击被提出，它将所有物品嵌入坍塌到几个聚类中，以混淆不同的物品[46]。然而，这些攻击都忽视了 FR 中独特的稀疏性。

1.3 论文动机与贡献

本文考虑最坏情况下的攻击，即拜占庭攻击[4,6,11,15,17,31]。在这种攻击中，攻击者是无所不知和串通一气的[41]，可以控制多个客户端上传任意恶意模型梯度。需要注意的是，对于使用联邦推荐的领域（例如电子商务），恶意客户端可以通过注册新帐户轻松注入。这自然引出了一个问题：**在独特的稀疏聚合下，联邦推荐模型对拜占庭攻击的鲁棒性如何呢？**

本文通过解决两个挑战来回答这个问题：(1)如何在稀疏聚合下定义拜占庭鲁棒性？现有的拜占庭攻击和防御主要基于一般联邦学习中的密集聚合机制进行定义。在联邦推荐中，由于推荐系统中用户与物品之间的交互通常是稀疏的，对于一个物品，其嵌入通常只由与之交互的用户更新，其余用户上传零值或空。因此，当直接应用现有的聚合器时，由于零值更新是占大多数，聚合的物品嵌入可能会偏向于零值，所以将现有聚合器转化为联邦推荐并重新检验其理论保证和有效性至关重要。(2)如何针对现实中具有不同知识和能力水平的攻击者设计针对联邦推荐的一般拜占庭攻击。具体而言，在联邦推荐中由于参与用户数量众多，攻击者很难获得所有用户的完全知识。此外，由于用户与物品之间的交互通常是稀疏的，拜占庭客户端不应该更新太多的物品。否则，在这种激进修改下，基于用户交互数的防御监控很容易被触发[39]。

本文首次从稀疏聚合的角度探究联邦推荐的拜占庭鲁棒性问题。为了解决第一个挑战，本文将单个物品的聚合作为最小执行单元，将现有的聚合器转移到联邦推荐模型中。也就是说，对于每个物品嵌入，梯度将被分别和并行地收集和聚合。基于这一点进一步指出，联邦推荐的这种稀疏聚合机制将导致一种独特的拜占庭漏洞：每个物品具有不同的度数，从而接收到不同数量的更新，导致其个体鲁棒性不同。在实际情况下，所有物品的度数通常符合长尾分布[1]。换句话说，大多数物品（称为长尾物品）只与少数用户存在交互，因此极其脆弱。

针对第二个挑战，本文基于联邦推荐中稀疏聚合的漏洞设计了一系列攻击策略，命名为 Spattack。然后将它们根据攻击者的知识和能力分为四类。

具体而言，遵循[4,11,41]，本文考虑了无所不知的攻击者（Spattack-I）和不完全知情的攻击者（Spattack-II）。然后进一步根据是否限制每个恶意客户端攻击物品的最大数量来划分 Spattack。

总之，本文的贡献有三个方面：

- 将单个物品的聚合作为最小执行单元，首次从稀疏聚合的角度系统地研究了联邦推荐（FR）的拜占庭鲁棒性问题。理论上分析了其收敛保证，并指出了 FR 的一个特殊漏洞。

- 利用 FR 中稀疏聚合的漏洞，提出了一系列有效的拜占庭攻击策略 Spattack。其中 Spattack 可以根据攻击者的知识和能力分为四种类型。
- 在多个 FR 基准数据集上进行实验。大量实验结果表明，仅通过控制少数恶意客户端，Spattack 就可以阻止纯粹甚至防御下的 FR 模型收敛。

本文剩余部分的组织如下：第 2 节将介绍集中式推荐系统、联邦式推荐系统、拜占庭攻防的相关技术与预备知识；第 3 节将重新定义联邦推荐中稀疏聚合下的拜占庭鲁棒性，在理论上指出其内在的漏洞，并基于这个漏洞设计一系列攻击策略 Spattack；第 4 节将对 Spattack 的有效性、迁移性、超参数影响等方面进行大量实验与分析。最后，在第 5 节总结本文的工作，并指出未来的研究目标。

2 相关技术与预备知识

在本节中将正式定义集中式和联邦式推荐系统，并介绍现有拜占庭攻击和防御的背景，表 2-1 为论文符号表。

表 2-1 论文符号表

$\mathcal{D}$	所有的用户-物品交互
$\mathcal{D}_i$	用户u_i的本地用户-物品交互
$\mathcal{U}$	n个良性用户集合
$\tilde{\mathcal{U}}$	n个恶意用户集合
$\mathcal{V}$	m个物品集合
$\mathcal{V}_{u_i}$	与用户u_i交互的物品集合
$\mathcal{U}_{v_j}$	与物品v_j交互的用户集合
U	用户嵌入信息$U=\{u_1,\dots,u_n\}$
V	物品嵌入信息$V=\{v_1,\dots,v_m\}$
$\nabla V^{i,t}$	良性用户u_i在t轮的嵌入梯度$\mathcal{V}$
$\nabla \tilde{V}^{i,t}$	恶意用户$\tilde{u}_i$在t轮的嵌入梯度$\mathcal{V}$
$\nabla v_j^{i,t}$	良性用户u_i在t轮对v_j的嵌入梯度
$\nabla \tilde{v}_j^{i,t}$	恶意用户$\tilde{u}_i$在t轮对v_j的嵌入梯度
Θ	神经网络模型的参数
$[n]$	整数集合$\{1,\cdots,n\}$
η	学习率
ρ	恶意用户（客户端）的比例
α	统计鲁棒性聚合器的破坏点

2.1 集中式推荐系统

一个推荐系统包含了一组用户$\mathcal{U}=\{u_1,u_2,\dots,u_n\}$和一组物品$\mathcal{V}=\{v_1,v_2,\dots,v_m\}$，其中$n$和$m$分别表示用户和物品的数量。每个用户$u_i\in\mathcal{U}$拥有一个由隐式反馈三元组$(u_i,v_j,r_{ij})$组成的本地训练数据集$\mathcal{D}_i$，它们代表用户和物品的交互情况（例如购买、点击或观看等）。其中，$r_{ij}=1$和$r_{ij}=0$分别表示正实例和负实例，即u_i是否与v_j进行交互。对于每个用户u_i，定义$\mathcal{V}_{u_i}=\{v_j\in\mathcal{V}|(u_i,v_j,r_{ij})\in\mathcal{D}_i\}$作为与$u_i$发生过交互的物品集合。记$U=[u_1,u_2,\cdots,u_n]$和$V=[v_1,v_2,\cdots,v_m]$分别表示用户和物品的嵌入向量，那么推荐系统就被训练成可以预测每个用户u_i和物品v_j之间的评分得分$\hat{r}_{ij}=f_\Theta(u_i,v_j)$，其中$\hat{r}_{ij}$表示用户$u_i$对物品$v_j$的偏好程度，$f_\Theta$是得分函数，$U,V,\Theta$是需要学习的模型参数。在传统的集中式训练系统中，每个用户u_i的个人数据集$\mathcal{D}_i$存储在中央服务器上，从而得到模型训练的总数据集$\mathcal{D}$，这将增加服务器的信息过载与用户的隐私风险。

2.2 联邦式推荐系统

在考虑隐私问题的情况下，联邦式推荐系统中用户u_i的隐私数据$\mathcal{D}_i$保留在本地设备上，每个用户共享模型参数V和Θ，通过将本地模型梯度发送到中央服务器进行聚合。根据基础推荐模型的不同，参数Θ是不同的：在矩阵分解的推荐模型中，交互函数Θ是固定的一个空集；在基于深度学习的推荐模型中，Θ是神经网络的权重集合。为了简单起见，本文沿用[29]中采用的经典且广泛使用的矩阵分解（Matrix Factorization，MF）作为基础推荐模型。在 MF 中，f被固定为点乘，即$\hat{r}_{ij} = u_i \odot v_j$。根据[29]采用 Bayesian Personalized Ranking (BPR)[27]作为每个客户端的本地损失函数，这是一种基于对比学习的个性化排名损失函数，具体公式如下：

$$\mathcal{L}_i(u_i, V) = -\sum_{\substack{v_j, v_k \in \mathcal{V}_{u_i} \\ r_{ij}=1 \wedge r_{ik}=0}} \ln\, \sigma(\hat{r}_{ij} - \hat{r}_{ik}) \tag{2.1}$$

其中σ是逻辑 Sigmoid 函数。公式 2.1 假设比起所有的负样本物品，用户更喜欢正样本物品。在联邦学习模式下，中央服务器每次将当前物品嵌入V^t发送给所有客户端，每个客户端用户u_i计算模型损失值$\mathcal{L}_i(u_i^t, V^t)$，然后在本地更新它的私有用户嵌入，具体如下：

$$u_i^{t+1} \leftarrow u_i^t - \eta \cdot \nabla u_i^t, \tag{2.2}$$

其中η是学习率。客户端用户u_i上传其本地物品嵌入的梯度$\nabla V^{i,t}$到中央服务器后，服务器通过以下方式更新V^t：

$$V^{t+1} \leftarrow V^t - \eta \cdot \sum_{i \in [n]} \nabla V^{i,t} \tag{2.3}$$

如图 1-1（b）所示，对于每个用户u_i，私有的交互历史（物品列表）和用户嵌入（绿色的u_i向量）保存在本地客户端上，只有物品嵌入V的梯度被发送至服务器。因此在整个训练阶段，所有用户的隐私都得到了很好的保护。

2.3 拜占庭攻击与防御

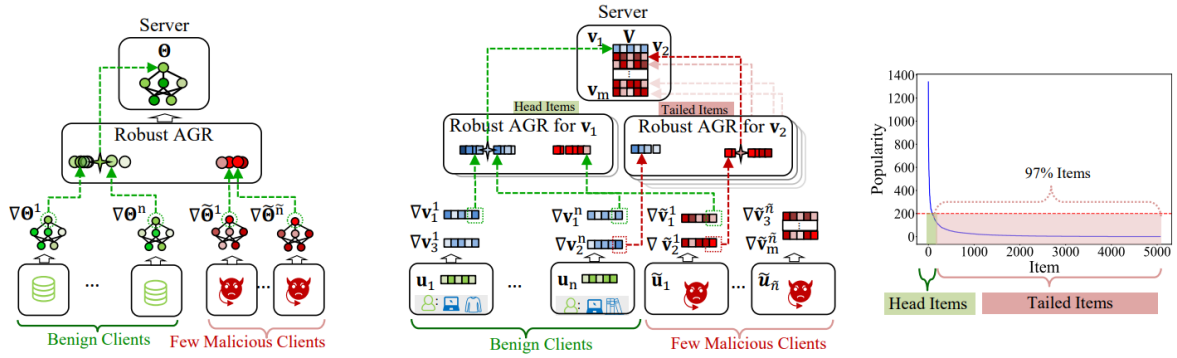
联邦系统容易受到恶意攻击，因此模型安全性是其基本原则之一。拜占庭攻击可以被视为最坏的攻击情况，允许恶意客户端勾结并上传任意模型梯度，以阻止全局模型训练的收敛。

2.3.1 拜占庭攻击

在拜占庭攻击中，攻击者旨在通过控制少数恶意客户端来降低模型的性能甚至阻止模型收敛。如图 2-1（a）所示，恶意用户$\tilde{u}_i$可以发送任意的（红色）梯度$\nabla\tilde{\Theta}^i$。根据[42]的假设，在考虑最坏情况的前提下，假设攻击者完全知道所有良性梯度$\{\nabla\Theta^1, \cdots, \nabla\Theta^n\}$，并且所有恶意客户端默认情况下都是串通的。

2.3.2 拜占庭鲁棒聚合器

防御措施通常在服务器端作为鲁棒聚合器执行。由于服务器无法访问客户端的原始训练数据，通常采用统计上的鲁棒聚合器来过滤拜占庭更新，从而保证 FL 的收敛。以图 2-1（a）为例，假设$\{\nabla\Theta^1, \cdots, \nabla\Theta^n\}$是联邦学习中$n$个良性客户端的梯度向量，服务器使用联邦聚合器收集和聚合每个客户端模型的训练梯度。在无攻击的 FL 设置中，平均值形式的$\text{Mean}(\nabla\Theta^1, \cdots, \nabla\Theta^n) = \frac{1}{n}\sum_{i=1}^{n} \nabla\Theta^i$是一种有效的聚合规则，然而Mean可能会被几个恶意客户端操纵[5]。最近已提出多种 FL 的拜占庭鲁棒聚合器[5, 25, 28, 42, 43]来过滤拜占庭更新并保证 FL 的收敛。例如，中位数聚合器$\text{Median}(\cdot)$计算参数Θ中每个元素Θ_i的中位数[43]，当恶意客户端的比例小于 50%时，中位数聚合器可以在拜占庭攻击下得到正确梯度（绿色星号）来保证模型收敛。



(a) 针对一般联邦学习的拜占庭攻击　　(b) 我们针对联邦推荐的拜占庭攻击

图 2-1 一般联邦学习和联邦推荐的拜占庭鲁棒性分析

3 面向联邦推荐的对抗攻击算法的设计

在本节中，本文重新定义稀疏聚合下的拜占庭鲁棒性，并在理论上指出其内在的漏洞。基于这个漏洞，针对攻击者的不同知识和能力水平设计了一系列攻击策略，命名为Spattack。

3.1 拜占庭攻击问题定义

对于拜占庭攻击，攻击者可以注入一些拜占庭用户$\tilde{\mathcal{U}} = \{\tilde{u}_1, \cdots, \tilde{u}_{\tilde{n}}\}$，使恶意用户的比例小于ρ，即$\tilde{n}/(n+\tilde{n}) < \rho$。一个恶意客户端$\tilde{u}_i \in \tilde{\mathcal{U}}$可以在任何时刻$t$上传梯度空间中的任意恶意方向的更新值$\nabla\tilde{v}_j^{i,t} = *$来扰动物品嵌入。然后服务器收集和聚合所有梯度，包括良性的$\{\nabla V^{1,t}, \cdots, \nabla V^{n,t}\}$和恶意的$\{\nabla\tilde{V}^{1,t}, \cdots, \nabla\tilde{V}^{\tilde{n},t}\}$。AGR(·)是联邦学习的聚合算子，它可以是最常见的Mean(·)，或者是具有统计鲁棒性的Median(·)，拜占庭攻击旨在防止模型收敛，即让推荐损失$\mathcal{L}_i$不降低。具体而言，FR 上的拜占庭攻击目标可以定义为以下的最优化问题：

$$\begin{aligned} &\max_{\{\nabla\tilde{V}_i^t : i\in\tilde{n}\}} \sum_{i=1}^{n} \left(\mathcal{L}_i(u_i^{t+1}, V^{t+1}) - \mathcal{L}_i(u_i^t, V^t)\right), \\ &s.t.\, V^{t+1} = V^t - \eta \cdot \mathrm{AGR}(\{\nabla V^{i,t} : i \in [n]\} \cup \{\nabla\tilde{V}^{i,t} : i \in [\tilde{n}]\}), \\ &u_i^{t+1} = u_i^t - \eta\nabla u_i^t, \text{ for } i \in [n], \\ &\frac{\tilde{n}}{n+\tilde{n}} \le \rho, \end{aligned} \tag{3.1}$$

其中攻击者旨在找到最优的恶意梯度集合$\{\nabla\tilde{V}_i^t : i \in [\tilde{n}]\}$，以提高更新后的损失$\mathcal{L}$。在解决这个最优化问题之前，本文发现 FR 有一个独特的稀疏聚合机制，定义如下：

定义 3.1. 密集/稀疏聚合。设$\theta \in R^d$是共享的模型参数向量，如果存在一个元素$\theta_i (i \in [d])$，只有一部分客户端可以产生有用的更新，则参数θ是稀疏聚合的。如果所有客户端都可以支持更新，则为密集聚合。

如图 2-1（a）所示，在一般的联邦学习中，模型参数Θ的每个元素（绿色圆圈）都被假定参与所有客户端的损失函数，即密集聚合。与一般联邦学习不同的是，在 FR 中对于客户端u_i，并不是所有物品嵌入$V = \{v_1, \cdots, v_m\}$都包含在式 2.1 中的本地损失$\mathcal{L}_i$中。例如，如图 2-1（b）所示，客户端u_1只能计算与其交互的物品v_1和v_3的有用梯度$\{\nabla v_1^1, \nabla v_3^1\}$，其余物品的梯度为零或空集，因此在时期$t$上传的物品嵌入梯度$V^{i,t}$是稀疏的。当直接应用聚合器到所有$V_i$上时，物品嵌入将偏向零值。

将密集聚合器调整为稀疏聚合器。通过将单个物品的聚合视为最小的执行单元，将现有的密集聚合器改编为稀疏聚合器。如图 2-1（b）所示，聚合器是针对每个物品单独进行的，以第j个物品的嵌入为例，该嵌入通过以下方式进行更新:

$$v_j^{t+1} = v_j^t - \eta \cdot \mathrm{AGR}\left(\nabla v_j^{i,t} \mid \text{user } i \in \mathcal{U}_{v_j}\right) \tag{3.2}$$

其中$\mathcal{U}_{v_j}$是与物品v_j交互的用户集合，$\nabla v_j^{i,t}$是来自客户端u_i在时刻t发送的关于物品v_j的梯度。当且仅当用户u_i与物品v_j存在交互时，才能将梯度$\nabla v_j^{i,t}$单独和并发地聚合到v_j^{t+1}中。直观地说，不同物品所收到的梯度数量各不相同，导致每个物品都具不同的鲁棒性。因此在稀疏聚合下，需要从理论上重新审视现有的聚合器在拜占庭攻击下的收敛保证。

3.2 拜占庭防御鲁棒性分析

鉴于 FR 中独特的稀疏聚合，本文进一步分析了在纯粹的和统计鲁棒的聚合器下 FR 的鲁棒性，并指出与通常的 FL 相比稀疏聚合将引入一种新的漏洞。

3.2.1 没有防御机制的 FR 的鲁棒性

与一般的 FL 算法一样，没有防御机制的联邦推荐系统通常使Mean(·)聚合器计算输入向量的平均值，这对拜占庭攻击非常敏感，就像命题 3.1 所述，即使只有一个恶意客户端也可以摧毁Mean(·)聚合器。

命题 3.1. 对于每个物品v_j，令$\{\nabla v_j^{i,t} \mid \text{user } i \in \mathcal{U}_{v_j}\}$为时刻$t$中良性梯度向量的集合。考虑对每个元素进行平均更新的Mean(·)聚合器，设$\nabla \tilde{v}_j$是一个维度为R^d的具有任意值的恶意更新。则$\mathrm{Mean}(\{\nabla v_j^{i,t} \mid \text{user } i \in \mathcal{U}_{v_j}\} \cup \nabla \tilde{v}_j) = \frac{1}{|\mathcal{U}_{v_j}|+1}\left(\sum_{i \in \mathcal{U}_{v_j}} \nabla v_j^{i,t} + \nabla \tilde{v}_j\right)$的输出可以被一个单一的恶意$\nabla \tilde{v}_j$控制为零向量。当所有的物品都受到攻击时，一个单一的恶意攻击者就可以阻止收敛。

证明 如果攻击者添加了一个恶意客户端，并且每个物品v_j上传的嵌入梯度为$\nabla \tilde{v}_j = -\sum_{i \in \mathcal{U}_j} \nabla v_j^{i,t}$的情况下，Mean(·)聚合器的输出是零向量，这将防止模型收敛。 □

3.2.2 具有防御机制的 FR 的鲁棒性

最常见的防御方法是使用统计上对离群值更加鲁棒的聚合器，在这些防御方法中，FL 模型具有一致的高破坏点，例如当$\rho < 50\%$时，Median(·)聚合器在理论上可以保证 FL 的收敛性，这已经在文献[43]中得到证明。然而本文发现在 FR 中不同的物品具有不同的破坏点，这取决于物品的度。具体来说，每个物品嵌入只能由与该物品交互的特定客户端进行更新，因此具有大量更新的流行物品将更加鲁棒。不幸的是，在推荐系统中只有少数物品经常进行交互（头部物品），而其余物品的交互频率较低（尾部物品）。以 Steam 推荐数据集[8]中的物品流行度（即物品的度）为例，在图 2-1（b）中绘制出来。如图所示，97%的尾部物品（红色长尾区域）的交互次数少于 200 次，只有 3%的头部物品（绿色区域）频繁地进行超过 200 次的交互。现有的 FL 防御方法主要基于统计学，因此无法保证大多数物品的收敛性。

设x为物品的度，$p(x)$为其概率分布函数，假设概率分布可以定义为幂律分布的典型形式$p(x) = Cx^{-\beta}$，其中C是归一化常数，β是幂律分布的缩放参数，防御失败可以形式化地描述如下：

命题 3.2. 设α为联邦鲁棒聚合器的破坏点，良性客户端和恶意客户端的数量分别为n和$\tilde{n}$，物品度的幂律分布的缩放参数为β，常数为C。那么至少有$1 - \frac{C}{\beta-1}\left(\frac{1-\alpha}{\alpha}\tilde{n}\right)^{(1-\beta)}$的物品嵌入可以被破坏。

证明 考虑具有破坏点α的鲁棒聚合器，当恶意客户端的数量满足$\frac{\tilde{n}}{n+\tilde{n}} < \alpha$时，模型收敛可以得到保证。而在 FR 中，对于一个度为d_v的物品v，聚合器只会在稀疏聚合中收集d_v个良性梯度。在拜占庭攻击中，所有的恶意客户端可以轻易地勾结起来向某个物品v发送一致的恶意梯度。一旦$\tilde{n}/(d_v + \tilde{n}) > \alpha$，即$v$的度数满足$d_v < (\tilde{n} - \alpha\tilde{n})/\alpha$，那么恶意更新将最终占据主导地位，因此统计上鲁棒的聚合器将被欺骗选择恶意更新。设$p(x) =$

$Cx^{-\beta}$是物品度的幂律分布,其累积分布函数$P(x)$定义为物品度大于x的概率:

$$P(x) = \mathrm{P}\,\mathrm{r}(X > x) = C\int_{x}^{+\infty} p(X)dX = C\int_{x}^{+\infty} X^{-\beta}dX = \frac{Cx^{(1-\beta)}}{\beta - 1} \tag{3.3}$$

因此，物品的度小于$(\tilde{n} - \alpha\tilde{n})/\alpha$的概率可以计算如下：

$$1 - \mathrm{P}\left(\mathrm{X} > \frac{1-\alpha}{\alpha}\tilde{n}\right) = 1 - \frac{C}{\beta - 1}\left(\frac{1-\alpha}{\alpha}\tilde{n}\right)^{(1-\beta)} \tag{3.4}$$

□

在拜占庭攻击下，鲁棒聚合器可以过滤掉一般联邦学习中的离群值，但无法保护长尾物品嵌入向量在联邦推荐系统中的安全。以 Steam 数据集为例，物品的度分布可以建模为图 2-1（b）所示的典型的幂律分布形式。例如，如果攻击者可以控制$\rho = 5\%$的客户端,则每个物品最多只能接收 197 个恶意梯度。显然对于那些交互次数少于 200 次的 97%的尾部物品，只需要极少量的恶意（红色）更新就可以成为主导并占据聚合。在这种情况下，统计上鲁棒的Median(·)聚合器将选择大多数（红色圆点），产生恶意输出（红色星号）。综上所述，一般情况下，统计上鲁棒的 FL 聚合器容易受到 FR 中稀疏聚合机制的漏洞攻击。

3.3 拜占庭攻击策略设计

接下来本文将介绍利用上述漏洞攻击 FR 的攻击策略，然后针对攻击者的不同知识和能力水平提出 Spattack。

3.3.1 攻击直觉

为了阻止推荐系统的收敛，公式 3.1 中的攻击者旨在阻止模型推荐损失降低。考虑稀疏聚合的漏洞，得出以下直觉：(1)恶意客户端上传的恶意梯度与真实梯度之间的距离越远，攻击的影响就越大。(2)在训练过程中，破坏的物品数量越多，攻击就越强大。因此，可以通过上传尽可能远离真实梯度的梯度，并尽可能多的破坏物品嵌入来最大化攻击目标。

表 3-1 Spattack 攻击分类

Spattack	I-D	I-S	II-D	II-S
Knowledge	√	√		
Capability	√		√	

3.3.2 攻击分类

表 3-1 总结了不同知识和能力水平下攻击者可以使用的 Spattack 方案,其中 Knowledge 指是否知道良性梯度信息，Capability 指是否攻击所有物品。在实际场景中，Spattack 通常在不同的情况下进行，具体取决于攻击者对每个时期的良性梯度的了解以及恶意客户端中最多攻击的物品数量，因此，可能存在以下四种情况：

Spattack-I-D攻击。首先考虑最坏的情况，即攻击者既是全知的又是全能的：如果攻击者可以在每个时期获取良性梯度，并且在每个恶意客户端中攻击的物品数量没有限制，则攻击者可以发起 Spattack-I-D攻击。在这种情况下，根据第一个直觉，更远离真实梯度的恶意梯度会导致更大的攻击效果，因此 Spattack-I-D攻击可以上传与良性梯度方向相反的梯度。具体而言，对于物品v_j，从与v_j交互的良性用户$u_i\left(u_i \in \mathcal{U}_{v_j},\ u_i \in \mathcal{U}\right)$收集良性梯度$\nabla v_j^{i,t}$，然后将收集到的梯度进行平均，以获取预期的良性梯度$\nabla \bar{v}_j^t$，即$\nabla \bar{v}_j^t = \sum_{u_i \in \mathcal{U}_{v_j} \cap \mathcal{U}} \nabla v_j^{i,t}$。恶意客户端$\tilde{u}_i \in \tilde{\mathcal{U}}$将上传梯度$\nabla \tilde{v}_j^{i,t} = -\gamma \cdot \nabla \bar{v}_j^t$，其中比例系数$\gamma = \frac{1}{\tilde{n}}$。因此聚合器最终输出的平均梯度为零，从而防止物品嵌入的收敛。对于第二个直觉，贪心地破坏更多物品，通过上传所有物品的攻击梯度而最大化攻击目标。在这种攻击中，鲁棒的聚合器努力选择统计上的多数时，对于尾部物品将得到相反的良性梯度。根据命题 3.2，即使在只有少量恶意客户端的情况下，Spattack-I-D攻击者也保证会破坏多数物品嵌入。

Spattack-II-D攻击。FR 的稀疏聚合漏洞对于非全知攻击者也存在显著风险，攻击者只需为所有物品上传随机噪声即可，称为 Spattack-II-D。具体而言，在没有良性梯度知识的情况下，可以通过随机采样高斯噪声向量来构造恶意梯度，并在所有恶意客户端中保持相同的噪声。在Mean(·)AGR 下，聚合的梯度可能会因此偏斜。更糟糕的是，统计上鲁棒的 AGR，如Median(·)聚合器，可能会选取上传的随机噪声作为尾部物品的输出，因此非全知攻击者也可以破坏鲁棒的 AGR 并防止模型收敛。

Spattack-I-S和 Spattack-II-S攻击。假设$\tilde{m}_{max}$每个恶意客户端中待攻击物品的最大数量，在拜占庭梯度中，如果$\tilde{m}_{max}$越大，则攻击越强。假设$\tilde{m}_{max}$没有限制，在这种情况下，攻击很容易发挥作用，但是恶意客户端的过大$\tilde{m}_{max}$将导致攻击被检测到。为了限制恶意用户的行为类似于良性用户，本文将$\tilde{m}_{max}$限制为良性用户交互物品的最大数量。具体而言，为了使恶意客户端的注入尽可能难以察觉且有效，本文使用采样操作来确定恶意客户端的用户-物品分布，同时根据物品流行度的分布为每个恶意客户端采样待攻击物品列表，因此攻击者可以自动将更多的恶意梯度分配给与良性客户端有更多交互的物品，然后分别基于相反的良性梯度（Spattack-I-S）或随机噪声（Spattack-II-S）生成恶意梯度。

4 对抗攻击算法实现与实验结果分析

在本节中，本文进行了大量实验旨在回答以下研究问题（RQ）：

- RQ1：与现有的拜占庭攻击方法相比，Spattack 表现如何？
- RQ2：Spattack 能否破坏部署在 FR 上的防御措施？
- RQ3：Spattack 能否转移到不同的 FR 系统中？
- RQ4：超参数对 Spattack 的有什么影响？

4.1 实验环境

所有实验都在一台带有一个 GPU（NVIDIA GeForce RTX 3090）和 CPU（Intel Xeon Gold 6348）的 Linux 服务器上进行，并且其操作系统为 Ubuntu 18.04.5。本文使用深度学习库 PyTorch 实现了所提出的攻击。Python 和 PyTorch 的版本分别为 3.9.11 和 1.10.1+cu111。

4.2 实验设置

数据集。本文提出的 Spattack 在三个广泛使用的数据集上进行评估，包括电影推荐数据集 MovieLens-1M（ML1M）[13]，小型版本的 MovieLens-100K（ML100K）和游戏推荐数据集 Steam-200K（Steam）[8]，数据集的特征总结在表 4-1 中。遵循先前工作[14,29]中的数据集配置，通过将交互历史视为隐式反馈三元组并删除重复交互来统一互动。测试集采用留一法划分，其中用户的最新交互被保留为测试集，其余交互用作训练集。

表 4-1 数据集的统计

Dataset	#Users	#Items	#Edges	Sparsity
ML100K	943	1,682	100,000	93.70%
ML1M	6,040	3,706	1,000,209	95.53%
Steam	3,753	5,134	114,713	99.40%

评估方法。为了评估对测试物品推荐的排名质量，采用了两种常见的评估方法，即命中率（HR）和排名为K时的标准化折现累积增益（nDCG@K），这里分别设置K为 5 和 10。对于每个用户，由于在所有物品中对测试物品进行排名很耗时，因此遵循广泛使用的策略[9, 14]，从未与用户交互的物品中随机抽样 100 个物品，然后将测试物品排名在这 100 个物品中。HR@K表示测试物品是否在前K位中排名，nNDCG@K考虑位置重要性。较高的 HR@K和 nNDCG@K表示更好的推荐性能，上述评估指标仅在良性客户端上计算。

联邦推荐系统。本文在评估中使用 FedMF[29]作为目标模型的架构，概述可复现性

的详细信息如下：根据[29]使用 BPR 损失函数，FedMF 本地更新用户嵌入并在服务器上优化物品嵌入，嵌入单元大小设置为 32，使用均值为 0 标准差为 0.01 的正态分布初始化用户和物品嵌入。为了评估 Spattack 的泛化能力，本文还对最先进的 FedGNN[38]进行攻击，该方法可以在保护隐私的同时协同训练具有高阶用户-物品交互信息的 GNN 模型。本文使用 BPR 损失函数训练一个 2 层 FedGNN 模型，其中物品嵌入和 GNN 参数均在服务器进行全局优化，用户和物品嵌入维度设置为 32，隐藏单元大小设置为 64，使用 Xavier Glorot 的初始化方法并将增益值设为 1 来初始化 GNN 权重矩阵。随机梯度下降（SGD）被选为默认的优化算法，其学习率设置为 0.01，默认情况下训练总轮数设置为 200，恶意客户端从第一轮开始攻击。

基准拜占庭攻击策略。将提出的 Spattack 攻击与两类方法进行比较，首先是数据投毒攻击，其中攻击者通过修改训练数据生成恶意梯度：

- LabelFlip[34]不需要了解训练数据分布，仅通过翻转恶意客户端的训练标签来投毒数据。每个恶意客户端使用本地的正样本作为负样本，使用负样本作为正样本。
- FedAttack[39]通过使用不当的正/负样本来执行数据投毒攻击。每个恶意客户选择与用户兴趣最相似的物品作为负样本，与用户兴趣最不相似的物品为正样本。

第二种是模型投毒攻击，其攻击者直接修改本地模型梯度：

- Gaussian[11]估计良性梯度的高斯分布，然后上传分布中的样本。
- LIE[3]向良性梯度的平均值添加小量噪声。将 0.1 分配为影响模型参数标准差的缩放因子。
- Fang[11]在正常梯度的相反方向添加噪声。从[3,4]的随机均匀分布中选择攻击缩放因子。
- Cluster[46]上传旨在使物品嵌入坍塌成几个密集聚类的恶意梯度。将初始聚类数设置为 1，聚类数和阈值的范围设置为[1,10]。

Spattack-I-S/D。对于全知攻击者，默认以规模参数$\gamma = \frac{1}{\tilde{n}}$生成与良性更新相反方向的恶意梯度，平均聚合器将输出零梯度，防止物品嵌入的收敛。

Spattack-II-S/D。对于非全知攻击者，从均值为 0，标准差为 1 的高斯分布中采样当前梯度，所有恶意客户端将共享同一个梯度。

Spattack-I/II-S。当每个客户端中毒物品的最大数量$\tilde{m}_{max}$受到限制时，根据物品度数的分布为每个恶意客户端采样待攻击物品列表，即更新次数更多的物品将接收更多的恶意更新，恶意客户端的采样操作是不可重复采样的。

基准拜占庭防御策略。为了验证 Spattack 是否可以破解 FL 中的鲁棒聚合器，本文在以下防御策略下评估 Spattack 性能：

- Mean 是 FL 中的基准聚合器，计算每个维度的梯度平均值。
- Median[43]是具有 0.5 破坏点的统计鲁棒聚合器，计算元素的中位数值。
- Trimmed-mean[43]对于每个维度修剪几个极端值，然后平均其剩余值。
- Krum[5]选择与上传的其他梯度最相似的梯度。
- Norm[33]使用阈值截取梯度的范数。

表 4-2 四种 Spattack 攻击类型（II-S，II-D，I-S，I-D）和 6 种基准拜占庭攻击在 3%恶意比例下的比较

Dataset	Metric	LabelFlip	FedAttack	Gaussian	LIE	Cluster	Fang	Type-II-S	Type-II-D	Type-I-S	Type-I-D
ML100K	HR@5	0.2517	0.2550	0.2550	0.2539	0.2461	0.1957	0.1018	0.0721	0.0594	**0.0530**
		(-3%)	(-2%)	(-2%)	(-2%)	(-5%)	(-25%)	(-59%)	(-71%)	(-76%)	**(-79%)**
	nDCG@5	0.1706	0.1721	0.1729	0.1724	0.1678	0.1229	0.0620	0.0380	0.0362	**0.0339**
		(-3%)	(-2%)	(-1%)	(-2%	(-4%)	(-30%)	(-62%)	(-77%)	(-78%)	**(-79%)**
	HR@10	0.4083	0.4094	0.4116	0.4116	0.3982	0.2919	0.2163	0.1601	0.0997	**0.0944**
		(-2%)	(-2%)	(-2%)	(-2%)	(-5%)	(-30%)	(-47%)	(-60%)	(-75%)	**(-77%)**
	nDCG@10	0.2206	0.2213	0.2230	0.2229	0.2166	0.1541	0.0980	0.0658	0.0492	**0.0470**
		(-2%)	(-2%)	(-1%)	(-1%)	(-4%)	(-32%)	(-54%)	(-69%)	(-77%)	**(-78%)**
ML1M	HR@5	0.3051	0.3056	0.3053	0.3054	0.3033	0.2827	0.1007	0.0921	0.0925	**0.0907**
		(-1%)	(-1%)	(-1%)	(-1%)	(-2%)	(-9%)	(-68%)	(-71%)	(-70%)	**(-71%)**
	nDCG@5	0.2013	0.2021	0.2017	0.2018	0.2004	0.1858	0.0581	**0.0521**	0.0553	0.0549
		(-2%)	(-1%)	(-1%)	(-1%)	(-2%)	(-9%)	(-72%)	**(-75%)**	(-73%)	(-73%)
	HR@10	0.4632	0.4634	0.4634	0.4634	0.4592	0.3977	0.2141	0.1935	0.1753	**0.1679**
		(-1%)	(-1%)	(-1%)	(-1%)	(-2%)	(-15%)	(-54%)	(-58%)	(-62%)	**(-64%)**
	nDCG@10	0.2522	0.2528	0.2526	0.2526	0.2506	0.2231	0.0939	0.0846	0.0817	**0.0793**
		(-1%)	(-1%)	(-1%)	(-1%)	(-2%)	(-12%)	(-63%)	(-67%)	(-68%)	**(-69%)**
Steam	HR@5	0.4792	0.4798	0.4879	0.4862	0.4263	0.0278	0.0426	**0.0139**	0.0671	0.0685
		(-15%)	(-15%)	(-14%)	(-14%)	(-25%)	(-95%)	(-93%)	**(-98%)**	(-88%)	(-88%)
	nDCG@5	0.3157	0.3172	0.3216	0.3209	0.2750	0.0160	0.0261	**0.0080**	0.0390	0.0408
		(-17%)	(-16%)	(-15%)	(-15%)	(-27%)	(-96%)	(-93%)	**(-98%)**	(-90%)	(-89%)
	HR@10	0.6429	0.6431	0.6474	0.6471	0.6220	0.0619	0.0834	**0.0322**	0.1308	0.1287
		(-7%)	(-7%)	(-6%)	(-6%)	(-10%)	(-91%)	(-88%)	**(-95%)**	(-81%)	(-81%)
	nDCG@10	0.3685	0.3700	0.3732	0.3730	0.3386	0.0269	0.0391	**0.0138**	0.0593	0.0601
		(-12%)	(-12%)	(-11%)	(-11%)	(-19%)	(-94%)	(-91%)	**(-97%)**	(-86%)	(-86%)

4.3 攻击性能评估（RQ1）

本文将四种 Spattack 与现有的基准攻击策略进行比较，并在表 4-2 中展示了实验结果，其中得分越低表示攻击效果越好，同时也报告了与没有攻击时的性能降低率。在 3% 的恶意客户端比例下，实验结果的主要发现总结如下：

- 在没有防御的情况下，Spattack 可以通过控制少量恶意客户端来防止联邦推荐模型的收敛，这表明 FR 极易受到 Spattack 的攻击。例如，在 3%恶意客户端下，Spattack 可以实现 47%到 98%的性能降低。这种显著的降低表明联邦推荐系统与标准联邦学习系统类似，极易受到拜占庭攻击的影响，只需要一个恶意客户端即可破坏整个 FR 模型（如命题 3.1 所示），因此实践中需要具有鲁棒性的聚合器。
- Spattack 显著优于所有基准攻击策略，这表明 FR 更易受到本文设计的拜占庭攻击的影响。其原因在于 Spattack 充分利用了稀疏聚合的脆弱性，通过贪心地破坏更多的物品来攻击。具体而言，LabelFlip 和 FedAttack 只能通过修改数据间接操纵梯度，而在 LIE，Cluster 和 Fang 中，可以沿着恶意方向直接操纵梯度，从而实现更高的攻击效果。虽然 Fang 可以在良性梯度的相反方向扰动，但由于没有考虑 FR 的稀疏聚合，良性梯度的平均值会偏向零向量，导致攻击效果不佳。
- 在大多数情况下 Steam 数据集的结果下降超过 ML100K 和 ML1M，这表明 Steam 的鲁棒性较低，原因是 Steam 平均涉及的交互较少（表 4-1 中表明其具有更大的稀疏性），因此具有更多易受攻击的尾部物品。

4.4 在防御下的攻击效果（RQ2）

现有的 FL 系统被证明极易受到拜占庭攻击的影响，因此近年来涌现了许多防御方法，它们可以在恶意客户端比例小于破坏点时保证模型收敛。Spattack 是否能够绕过这些防御机制，阻止 FR 模型的收敛？在本部分中评估了 Spattack 在不断增加的恶意客户端比例ρ下对防御 FR 的有效性。具体而言，将ρ设置为$\{0\%, 1\%, 3\%, 5\%\}$以评估全知 Spattack-I 的效果，并将其设置为更高的$\{0\%, 5\%, 10\%, 15\%\}$以评估更困难的非全知 Spattack-II。通过 Mean、Median 和 Norm 聚合器对所有攻击进行了 FR 实验，但仅在 Spattack-I-D 和 Spattack-II-D 下使用 Trimmed Mean（TrimM）和 Krum。因为 TrimM 和 Krum 防御假定每个元素的恶意更新数量是一致且固定的，而 Spattack-I-S 和 Spattack-II-S 对每个采样的物品嵌入向量进行了不同数量的更新，因此这些防御方法无法应用。结果显示在图 4-1 中，有以下发现：

- 只有 5%的恶意客户端就能使 Spattack 严重降低推荐性能，甚至在大多数情况下阻止模型收敛。尽管这些防御在 FL 中理论上可以保证在高恶意率下模型收敛（例如对于 Median，$\rho < 50\%$），但它们在 FR 中比预期要脆弱得多，这揭示了 FR 独特的易受攻击性，原因在于不同的物品具有不同数量的更新，因此大多数物品（尾部物品）的防御更容易被攻破。
- 当攻击者的知识和能力受限时，在恶意比率增加时，防御 FR 的性能持续下降，甚至无法收敛。例如，当$\rho = 10\%$时，Spattack-II-*S*可以在 Median 聚合器下阻止模型收敛，此时的ρ远低于 Median 的破坏点 50%。结果也提供了一个有价值的启示：隐藏良性客户端的梯度不能很好地保护联邦推荐系统，因为攻击者可以使用随机噪声来替换梯度，并且仍然能够破坏防御。
- 此外，可以观察到 Norm 聚合器比统计鲁棒性的聚合器提供更好的防御效果。因为在相同方差的高斯噪声下，虽然较大的方差将有利于在统计鲁棒性聚合器下扭曲数据，但是可以被范数约束防御轻松截断。

更详细的实验结果显示在附录 A 中的表 A-1、表 A-2、表 A-3 和表 A-4 中。

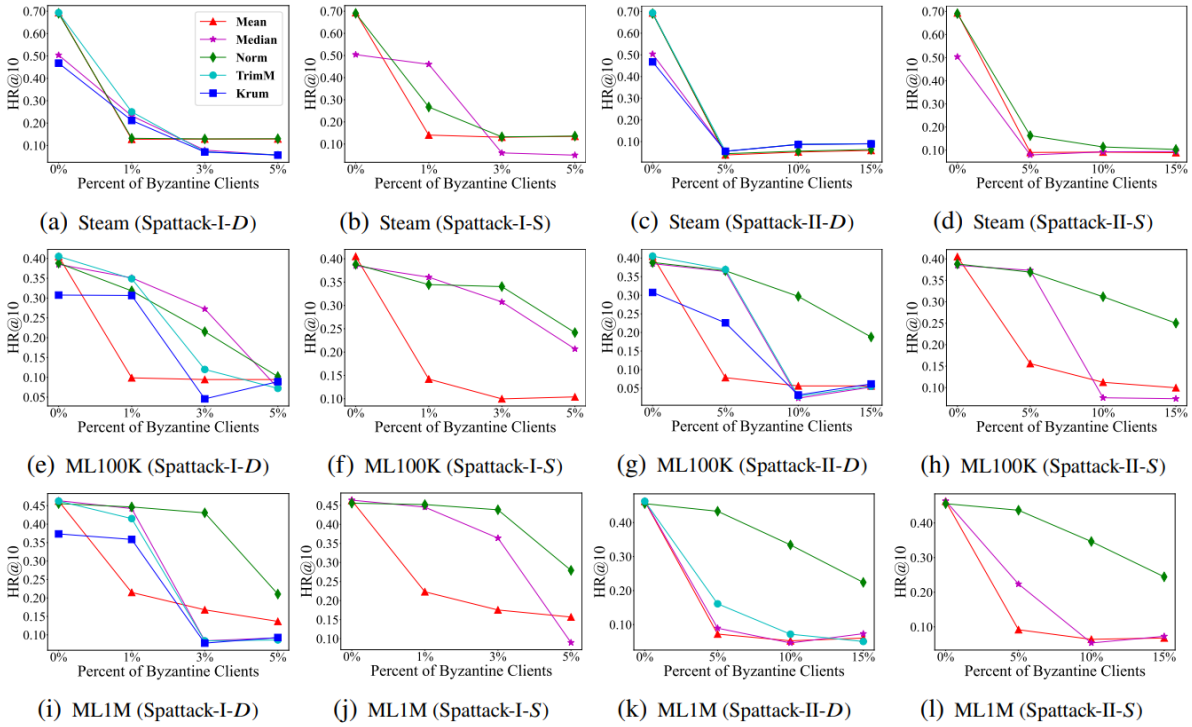


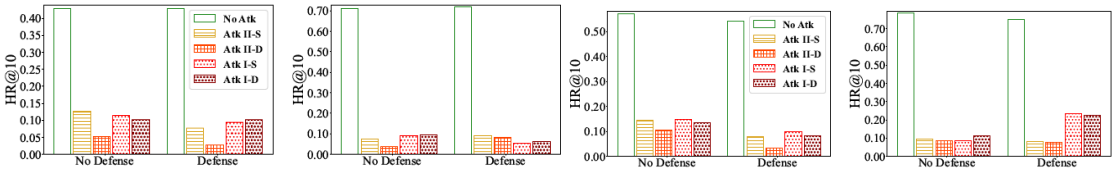
图 4-1 Spattack 在不同比例下的拜占庭客户端中对多种防御策略的性能表现

4.5 攻击的可迁移性（RQ3）

除了 FedMF 外，Spattack 攻击能否转移到其他联邦推荐系统呢？在实际攻击场景中，攻击者需要破坏不同类型的 FR 模型。在探索了 Spattack 如何影响使用 SGD 训练的 FedMF 模型后，本文将了解 Spattack 攻击是否也适用于其他 FR 模型和优化器。为了证明 Spattack 对其他联邦学习具有通用性，本文通过上传物品嵌入和额外 GNN 模型参数的恶意梯度，对最先进的 FedGNN[38]执行了四种攻击策略。没有防御和防御方案对应于 Mean 聚合器和 Median 聚合器。结果显示在图 4-2 中，并且完整的指标和数据集结果显示在附录 A 中的图 A-1 中。

首先观察到，所有指标的性能在 Spattack 攻击下急剧下降，证明了 FedMF 和 FedGNN 的共同漏洞。即使是 GNN 的参数密集聚合，攻击者也可以通过投毒物品嵌入来阻止模型训练。此外，Spattack 在防御情况下可以实现更有效的攻击。原因是对于大多数物品（尾部物品），恶意梯度很容易成为其聚合中的大多数，因此中位数聚合器倾向于选择将恶意梯度作为输出，而平均聚合器中的毒化则会因为平均恶意和良性梯度而减轻。类

似的现象也可以在图 4-4 中看到。本文还展示了 Spattack 在使用 Adam 优化器时的有效性，结果显示在图 4-3 中，更详细的结果在附录 A 图 A-2 中呈现。



(a) ML100K (b) Steam

图 4-2 针对 FedGNN 模型的攻击性能

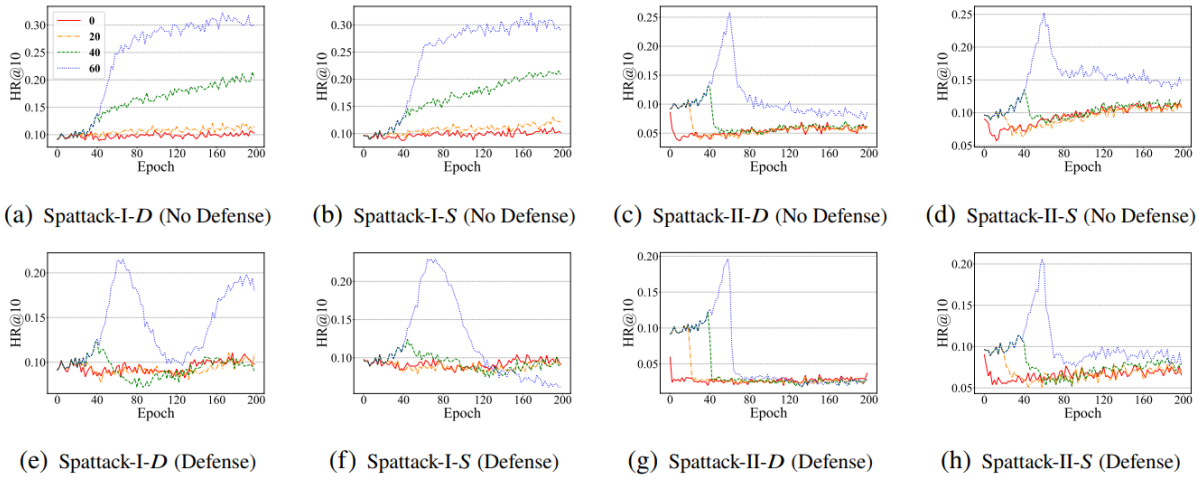
(a) ML100K (b) Steam

图 4-3 在 Adam 优化器下的攻击性能

4.6 攻击的超参数分析（RQ4）

最后，本文研究了攻击参数对 Spattack 的影响。在图 4-4 中，展示了在训练过程中，在防御和没有防御条件下从 0、20、40 和 60 轮开始攻击的情况下，FR 的收敛情况。结果在 ML100K 数据集上用 200 个 epoch 的 HR@10 得分进行了可视化，得出如下有价值的见解：

首先，从较小的起始 Epoch 开始的 Spattack 倾向于具有更好的攻击性能，因为模型已在相对较大的 Epoch 下收敛。此外，可以观察到在没有防御的 Spattack-I-*D*和 Spattack-I-*S*中，从第 60 个 epoch 开始攻击的 HR@10（紫色线）在第 60 个 epoch 时停止上升并保持略微震荡，相比之下其他攻击将在第 60 个 epoch 后迅速降至 10。原因是在这里上传了恶意梯度，其平均值等于平均良性梯度的负数，导致聚合后物品嵌入的梯度为零。在防御版本的 FR 中，Median(·)聚合器将直接选择恶意梯度作为输出，并沿着相反方向迅速降低模型性能，这也验证了稀疏聚合机制的脆弱性。



(a) Spattack-I-*D* (No Defense) (b) Spattack-I-*S* (No Defense) (c) Spattack-II-*D* (No Defense) (d) Spattack-II-*S* (No Defense)

(e) Spattack-I-*D* (Defense) (f) Spattack-I-*S* (Defense) (g) Spattack-II-*D* (Defense) (h) Spattack-II-*S* (Defense)

图 4-4 在 ML100K 数据集上，Spattack 在不同的 Epoch 开始执行攻击的 FR 收敛情况

5 总结与展望

本文首次从 FR 独特的稀疏聚合角度系统地研究其拜占庭鲁棒性，其中 FR 中的物品嵌入只能由部分客户端进行更新（通常在 FL 中是密集聚合）。然后基于 FR 稀疏聚合漏洞，设计了一系列攻击策略，称为 Spattack，这些攻击是根据攻击者的知识和能力水平实施的。大量实验结果表明，FR 对 Spattack 非常脆弱，即使是很少的恶意客户端也可以破坏纯粹甚至防御下的 FR 模型的训练。

FR 的安全性是一个重要问题，本文提供了深入研究所需的重要见解。未来目标是针对 Spattack 定制一个鲁棒的聚合器，以保证模型的收敛性。

致谢

在此次毕设中，我得到了许多人的支持和帮助。在这里，我想对所有关心、支持我

的人表示最真挚的感激，他们给我提供了宝贵的指导和帮助，如同明灯照亮了前方的道路。

首先，我要感谢我的指导老师周子力副教授。在我本科期间，他不仅是我在学术上的引路人，更是我生活中的良师益友。在科研项目中，他为我提供了切实可行的建议和指导，并时刻鼓励我勇往直前、不断进取。同时，在毕设过程中，我也得到了北京邮电大学张梦玫博士的的帮助。她不仅为我提供了宝贵的研究思路和方法，还耐心而细致地指导我方案设计、实验分析等各个环节，她对科研的严谨，执着追求完美的工作态度也深深影响了我。

此外，我还要感谢曲阜师范大学数字大脑团队与北京邮电大学 GammaLab 实验室的所有老师和同学，是他们给了我良好的学习、研究和交流平台，让我受益匪浅。他们不仅在科研上给予了我帮助，还为我提供了温馨舒适的学习环境。

最后，向所有给予我帮助和关注的老师、同学、朋友和家人们表示由衷的感激！祝愿您们身体健康、工作顺利、前程似锦！

参考文献

[1] H. Abdollahpouri, R. Burke, and B. Mobasher. Managing popularity bias in recommender systems with personalized re-ranking. In The Florida AI Research Society, 2019.

[2] M. Ammaduddin, E. Ivannikova, S. A. Khan, W. Oyomno, Q. Fu, K. E. Tan, and A. Flana- gan. Federated collaborative filtering for privacy-preserving personalized recommendation system. arXiv: Information Retrieval, 2019.

[3] G. Baruch, M. Baruch, and Y. Goldberg. A little is enough: Circumventing defenses for distributed learning. Neural Information Processing Systems, 2019.

[4] M. Baruch, G. Baruch, and Y. Goldberg. A little is enough: Circumventing defenses for distributed learning. In Neural Information Processing Systems, 2019.

[5] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer. Machine learning with adversaries: byzantine tolerant gradient descent. Neural Information Processing Systems, 2017.

[6] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In NIPS, 2017.

[7] D. Chai, L. Wang, K. Chen, and Q. Yang. Secure federated matrix factorization. IEEE Intelligent Systems, 36:11–20, 2019.

[8] G. Cheuque, J. Guzmán, and D. Parra. Recommender systems for online video game plat- forms: the case of steam. Companion Proceedings of The 2019 World Wide Web Confer- ence, 2019.

[9] A. M. Elkahky, Y. Song, and X. He. A multi-view deep learning approach for cross do- main user modeling in recommendation systems. Proceedings of the 24th International Conference on World Wide Web, 2015.

[10] W. Fan, Y. Ma, D. Yin, J. Wang, J. Tang, and Q. Li. Deep social collaborative filtering.Proceedings of the 13th ACM Conference on Recommender Systems, 2019.

[11] M. Fang, X. Cao, J. Jia, and N. Z. Gong. Local model poisoning attacks to byzantine-robust federated learning. In USENIX Security Symposium, 2019.

[12] S. Fu, C. Xie, B. Li, and Q. Chen. Attack-resistant federated learning with residual-based reweighting. ArXiv, abs/1912.11464, 2019.

[13] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. ACM Trans. Interact. Intell. Syst., 5:19:1–19:19, 2016.

[14] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. Proceedings of the 26th International Conference on World Wide Web, 2017.

[15] S. Hu, J. Lu, W. Wan, and L. Y. Zhang. Challenges and approaches for mitigating byzantine attacks in

federated learning. ArXiv, abs/2112.14468, 2021.
[16] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for rec- ommendation in social networks. In ACM Conference on Recommender Systems, 2010.
[17] L. Lamport, R. E. Shostak, and M. C. Pease. The byzantine generals problem. ACM Trans. Program. Lang. Syst., 4:382–401, 1982.
[18] Z. Lin, W. Pan, and Z. Ming. Fr-fmss: Federated recommendation via fake marks and secret sharing. Proceedings of the 15th ACM Conference on Recommender Systems, 2021.
[19] Z. Liu, L. Yang, Z. Fan, H. Peng, and P. S. Yu. Federated social recommendation with graph neural network. ACM Transactions on Intelligent Systems and Technology (TIST), 13:1 – 24, 2021.
[20] S. Luo, Y. Xiao, and L. Song. Personalized federated recommendation via joint repre- sentation learning, user clustering, and model adaptation. Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022.
[21] L. Lyu, H. Yu, X. Ma, C. Chen, L. Sun, J. Zhao, Q. Yang, and S. Y. Philip. Privacy and robustness in federated learning: Attacks and defenses. IEEE transactions on neural networks and learning systems, 2022.
[22] L. Lyu, H. Yu, J. Zhao, and Q. Yang. Threats to federated learning. Federated Learning: Privacy and Incentive, pages 3–16, 2020.
[23] H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using proba- bilistic matrix factorization. In International Conference on Information and Knowledge Management, 2008.
[24] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication- efficient learning of deep networks from decentralized data. In International Conference on Artificial Intelligence and Statistics, 2016.
[25] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault. The hidden vulnerability of distributed learning in byzantium. In International Conference on Machine Learning, 2018.
[26] K. Pillutla, S. M. Kakade, and Z. Harchaoui. Robust aggregation for federated learning. IEEE Transactions on Signal Processing, 70:1142–1154, 2019.
[27] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian person- alized ranking from implicit feedback. ArXiv, abs/1205.2618, 2009.
[28] N. Rodr'iguez-Barroso, D. J. L'opez, M. V. Luz'on, F. Herrera, and E. Martínez-Cámara. Survey on federated learning threats: concepts, taxonomy on attacks and defences, exper- imental study and challenges. ArXiv, abs/2201.08135, 2022.
[29] D. Rong, S. Ye, R. Zhao, H. N. Yuen, J. Chen, and Q. He. Fedrecattack: Model poisoning attack to federated recommendation. 2022 IEEE 38th International Conference on Data Engineering (ICDE), pages 2643–2655, 2022.
[30] J. B. Schafer, J. A. Konstan, and J. Riedl. E-commerce recommendation applications. Data Mining and Knowledge Discovery, 5:115–153, 2004.
[31] V. Shejwalkar and A. Houmansadr. Manipulating the byzantine: Optimizing model poi- soning attacks and defenses for federated learning. In Network and Distributed System Security Symposium, 2021.
[32] Z. Sun, Y. Xu, Y. Liu, W. He, Y. L. Jiang, F. Wu, and L. zhen Cui. A survey on federated recommendation systems. ArXiv, abs/2301.00767, 2022.
[33] A. T. Suresh, B. McMahan, P. Kairouz, and Z. Sun. Can you really backdoor federated learning. arXiv: Learning, 2019.
[34] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu. Data poisoning attacks against federated learning systems. Cornell University - arXiv, 2020.
[35] H. Wang, F. Zhang, X. Xie, and M. Guo. Dkn: Deep knowledge-aware network for news

recommendation. Proceedings of the 2018 World Wide Web Conference, 2018.
[36] J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, and Lee. Billion-scale commodity embed- ding for e-commerce recommendation in alibaba. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018.
[37] C. Wu, F. Wu, M. An, J. Huang, Y. Huang, and X. Xie. Npa: Neural news recommen- dation with personalized attention. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019.
[38] C. Wu, F. Wu, L. Lyu, T. Qi, Y. Huang, and X. Xie. A federated graph neural network framework for privacy-preserving personalization. Nature Communications, 13, 2021.
[39] C. Wu, F. Wu, T. Qi, Y. Huang, and X. Xie. Fedattack: Effective and covert poisoning attack on federated recommendation via hard sampling. Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022.
[40] Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis. Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks. IEEE Transactions on Signal Pro- cessing, 68:4583–4596, 2019.
[41] C. Xie, O. Koyejo, and I. Gupta. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. ArXiv, abs/1903.03936, 2019.
[42] J. Xu, S.-L. Huang, L. Song, and T. Lan. Byzantine-robust federated learning through collaborative malicious gradient filtering. 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS), pages 1223–1235, 2021.
[43] D. Yin, Y. Chen, K. Ramchandran, and P. L. Bartlett. Byzantine-robust distributed learn- ing: Towards optimal statistical rates. arXiv: Learning, 2018.
[44] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018.
[45] S. Ying. Shared mf: A privacy-preserving recommendation system. ArXiv, abs/2008.07759, 2020.
[46] Y. Yu, Q. Liu, L. Wu, R. Yu, S. L. Yu, and Z. Zhang. Untargeted attack against feder- ated recommendation systems via poisonous item embeddings and the defense. ArXiv, abs/2212.05399, 2022.
[47] S. Zhang, H. Yin, T. Chen, Z.-L. Huang, Q. V. H. Nguyen, and L. zhen Cui. Pipattack: Poisoning federated recommender systems for manipulating item promotion. Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, 2021.
[48] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. J. Li. Drn: A deep reinforcement learning framework for news recommendation. Proceedings of the 2018 World Wide Web Conference, 2018.

附录 A

为了进一步说明 Spattack 的有效性，针对其四类攻击方法在不同的鲁棒聚合器下，补充更详细的实验结果与分析。

Spattack-I-D 攻击的性能评估。本文在表 A-1 中评估了 Spattack-I-D 攻击的有效性，假设攻击者知道良性梯度且没有更新物品数量的最大限制。现有的防御理论上保证了 FL 中高恶意率（例如$\rho < 50\%$）下的模型收敛，但在 FR 中比预期更脆弱，这揭示了现有防御措施在 FR 上存在新漏洞。具体来说，尽管这些防御措施可以在$\rho = 1\%$时保护模型免受急剧下降，但可以观察到，在 Spattack 攻击下防御性能随着恶意率ρ的增加持续下降，甚至无法收敛。当$\rho = 3\%$时，防御的退化已经相当大，即平均表现下降率约为 71%。当$\rho = 5\%$时，这种退化将进一步增加至 82%。原因是传统 FL 中所有客户端都会更新共享模型参数的每个元素，但用户通常与少量物品进行交互，并且只维护它们的嵌入，这

些尾部物品在防御措施中具有较低的破坏点，可以很容易地被攻击破坏。

Spattack-I-S 攻击的性能评估。此外，在考虑用户-物品交互的稀疏性的情况下，本文还在表 A-2 中评估了 Spattack-I-S 攻击。尽管 Spattack-I-D 攻击没有$\tilde{m}_{max}$的限制将上传涉及所有物品的梯度，但是这可能很容易触发基于用户度数的异常检测。在 Spattack-I-S 中，将$\tilde{m}_{max}$限制为良性客户端上传的最大物品数量。即使限制了恶意客户端的交互次数，Spattack-I-S 依然可以在大多数情况下显著降低推荐性能，并防止 FR 模型在 5%比例下收敛。例如，当恶意客户端比例$\rho = 5\%$时，Spattack-I-S 至少在 ML100K、ML1M 和 Steam 数据集上实现 35%、31%和 80%的性能下降率。直观地说，上传的中毒梯度中非零行越少，梯度的威力就越小，但是在某些情况下，Spattack-I-S 的性能下降与 Spattack-I-D 相比只有微小的差异。总的来说，Spattack 攻击策略可以绕过基于用户度数的异常检测，并且即使只有少数恶意客户端也能成功攻击。

Spattack-II-D 攻击的性能评估。考虑不是全知攻击者，即良性梯度不可用。通过随机生成高斯噪声向量作为所有客户端的梯度来启动 Spattack-II-D 攻击，其中高斯分布的均值和标准差分别设置为 0 和 1。对于非全知 Spattack-II，将恶意客户端比例ρ设置为{5%，10%，15%}。Spattack-II 需要更多的恶意客户端才能获得可比较的结果，原因是评估上传随机噪声作为梯度的最坏情况，即不使用任何梯度信息作为指导。如表 A-3 所示，尽管配备了具有静态鲁棒性的聚合器，Spattack 攻击仍然可以在 10%的比例下防止收敛并降低 81%到 97%，这种性能下降是由于稀疏聚合的脆弱性造成的。这些结果提供了一个宝贵的暗示，即隐藏良性客户端的梯度不能很好地保护联邦推荐系统，因为攻击者可以使用随机噪声作为替代品来破坏模型。此外，人们还可以观察到，在防御无关的攻击场景中，Norm 聚合器可以提供比静态鲁棒聚合器更好的防御效果。这是因为在高斯噪声相同标准差的情况下，大标准差有利于在静态鲁棒聚合器下扭曲数据，但可以轻松被规范范围内的防御裁剪。

Spattack-II-S 攻击的性能评估。即使在知识和能力有限的情况下，攻击 Spattack-II-D 仍显着降低了 FR 模型的推荐性能，如表 A-4 所示，这表明联邦推荐系统在实践中容易受到 Spattack 攻击影响，这可能会阻碍它在各个领域的应用，平均推荐性能在 5%、10%和 15%恶意比例下分别下降约 56%、73%和 78%。

表 A-1 在 Spattack-I-*D*攻击下的推荐性能

Dataset	Defense	1%				3%				5%			
		HR@5	nDCG@5	HR@10	nDCG@10	HR@5	nDCG@5	HR@10	nDCG@10	HR@5	nDCG@5	HR@10	nDCG@10
ML100K	Mean	0.0551 (-78%)	0.0354 (-78%)	0.0986 (-76%)	0.0491 (-77%)	0.0530 (-79%)	0.0339 (-79%)	0.0944 (-77%)	0.0470 (-78%)	0.0573 (-77%)	0.0352 (-79%)	0.0944 (-77%)	0.0470 (-78%)
	Median	0.2312 (-9%)	0.1545 (-10%)	0.3510 (-9%)	0.1924 (-10%)	0.1485 (-42%)	0.0943 (-45%)	0.2725 (-29%)	0.1339 (-37%)	0.0371 (-85%)	0.0233 (-87%)	0.0732 (-81%)	0.0346 (-84%)
	Norm	0.1972 (-17%)	0.1305 (-18%)	0.3181 (-18%)	0.1691 (-18%)	0.1410 (-41%)	0.0981 (-38%)	0.2153 (-45%)	0.1216 (-41%)	0.0530 (-78%)	0.0340 (-79%)	0.1018 (-74%)	0.0496 (-76%)
	TrimM	0.2269 (-10%)	0.1488 (-9%)	0.3489 (-14%)	0.1876 (-12%)	0.0647 (-74%)	0.0407 (-75%)	0.1198 (-70%)	0.0582 (-73%)	0.0361 (-86%)	0.0213 (-87%)	0.0721 (-82%)	0.0328 (-85%)
	Krum	0.1941 (+1%)	0.1235 (+3%)	0.3065 (0%)	0.1596 (+1%)	0.0255 (-87%)	0.0134 (-89%)	0.0456 (-85%)	0.0199 (-87%)	0.0509 (-73%)	0.0295 (-75%)	0.0891 (-71%)	0.0418 (-73%)
ML1M	Mean	0.1151 (-63%)	0.0702 (-66%)	0.2149 (-54%)	0.1022 (-60%)	0.0907 (-71%)	0.0549 (-73%)	0.1679 (-64%)	0.0793 (-69%)	0.0730 (-77%)	0.0437 (-79%)	0.1366 (-70%)	0.0640 (-75%)
	Median	0.2955 (-5%)	0.1975 (-4%)	0.4422 (-5%)	0.2446 (-4%)	0.0394 (-87%)	0.0228 (-89%)	0.0839 (-82%)	0.0370 (-85%)	0.0457 (-85%)	0.0270 (-87%)	0.0919 (-80%)	0.0418 (-84%)
	Norm	0.3000 (-2%)	0.1981 (-2%)	0.4465 (-2%)	0.2453 (-2%)	0.2901 (-5%)	0.1893 (-7%)	0.4306 (-5%)	0.2347 (-6%)	0.1442 (-53%)	0.0989 (-51%)	0.2104 (-54%)	0.1202 (-52%)
	TrimM	0.2593 (-17%)	0.1765 (-14%)	0.4151 (-10%)	0.2262 (-11%)	0.0391 (-87%)	0.0222 (-89%)	0.0838 (-82%)	0.0364 (-86%)	0.0445 (-86%)	0.0255 (-88%)	0.0863 (-81%)	0.0390 (-85%)
	Krum	0.2361 (0%)	0.1504 (0%)	0.3586 (-4%)	0.1899 (-3%)	0.0368 (-84%)	0.0216 (-86%)	0.0776 (-79%)	0.0346 (-82%)	0.0462 (-80%)	0.0268 (-82%)	0.0929 (-75%)	0.0418 (-79%)
Steam	Mean	0.0677 (-88%)	0.0403 (-89%)	0.1276 (-82%)	0.0596 (-86%)	0.0685 (-88%)	0.0408 (-89%)	0.1287 (-81%)	0.0601 (-86%)	0.069 (-88%)	0.0411 (-89%)	0.129 (-81%)	0.0603 (-86%)
	Median	0.1719 (-38%)	0.1205 (-38%)	0.2323 (-54%)	0.1400 (-47%)	0.0442 (-84%)	0.0265 (-86%)	0.0791 (-84%)	0.0376 (-86%)	0.0290 (-90%)	0.0175 (-91%)	0.0568 (-89%)	0.0262 (-90%)
	Norm	0.0717 (-87%)	0.0428 (-88%)	0.1322 (-81%)	0.0622 (-84%)	0.0690 (-87%)	0.0409 (-88%)	0.1292 (-81%)	0.0602 (-85%)	0.0682 (-87%)	0.0408 (-88%)	0.1300 (-81%)	0.0605 (-85%)
	TrimM	0.2001 (-65%)	0.1622 (-57%)	0.2502 (-64%)	0.1783 (-58%)	0.0378 (-93%)	0.0228 (-94%)	0.0714 (-90%)	0.0335 (-92%)	0.0288 (-95%)	0.0175 (-95%)	0.0584 (-92%)	0.0269 (-94%)
	Krum	0.1607 (-37%)	0.1253 (-29%)	0.2118 (-55%)	0.1416 (-42%)	0.0381 (-85%)	0.0237 (-87%)	0.0709 (-85%)	0.0341 (-86%)	0.0290 (-89%)	0.0175 (-90%)	0.0570 (-88%)	0.0264 (-89%)

表 A-2 在 Spattack-I-*S*攻击下的推荐性能

Dataset	Defense	1%				3%				5%			
		HR@5	nDCG@5	HR@10	nDCG@10	HR@5	nDCG@5	HR@10	nDCG@10	HR@5	nDCG@5	HR@10	nDCG@10
ML100K	Mean	0.0647 (-74%)	0.0373 (-77%)	0.1421 (-65%)	0.0618 (-71%)	0.0594 (-76%)	0.0362 (-78%)	0.0997 (-75%)	0.0492 (-77%)	0.0541 (-78%)	0.0318 (-81%)	0.1039 (-74%)	0.0479 (-78%)
	Median	0.2365 (-7%)	0.1591 (-8%)	0.3606 (-6%)	0.1989 (-7%)	0.1994 (-22%)	0.1312 (-24%)	0.3075 (-20%)	0.1660 (-22%)	0.1198 (-53%)	0.0720 (-58%)	0.2068 (-46%)	0.0996 (-54%)
	Norm	0.2216 (-7%)	0.1417 (-11%)	0.3446 (-11%)	0.1811 (-13%)	0.1994 (-16%)	0.1249 (-22%)	0.3404 (-12%)	0.1698 (-18%)	0.1538 (-35%)	0.1032 (-35%)	0.2418 (-38%)	0.1314 (-37%)
ML1M	Mean	0.1204 (-61%)	0.0738 (-64%)	0.2230 (-52%)	0.1065 (-58%)	0.0925 (-70%)	0.0553 (-73%)	0.1753 (-62%)	0.0817 (-68%)	0.0805 (-74%)	0.0493 (-76%)	0.1568 (-66%)	0.0736 (-71%)
	Median	0.2995 (-4%)	0.2001 (-2%)	0.4452 (-4%)	0.2468 (-3%)	0.2439 (-22%)	0.1570 (-23%)	0.3641 (-21%)	0.1957 (-23%)	0.0447 (-86%)	0.0264 (-87%)	0.0897 (-81%)	0.0407 (-84%)
	Norm	0.3028 (-1%)	0.1986 (-2%)	0.4520 (-1%)	0.2468 (-2%)	0.2902 (-5%)	0.1898 (-6%)	0.4382 (-4%)	0.2375 (-5%)	0.2023 (-34%)	0.1402 (-31%)	0.2793 (-39%)	0.1648 (-34%)
Steam	Mean	0.0701 (-88%)	0.0410 (-89%)	0.1404 (-80%)	0.0635 (-85%)	0.0671 (-88%)	0.0390 (-90%)	0.1308 (-81%)	0.0593 (-86%)	0.0695 (-88%)	0.0410 (-89%)	0.1348 (-81%)	0.0617 (-85%)
	Median	0.3333 (+20%)	0.2448 (+27%)	0.4602 (-9%)	0.2855 (+8%)	0.0266 (-90%)	0.0152 (-92%)	0.0600 (-88%)	0.0258 (-90%)	0.0218 (-92%)	0.0115 (-94%)	0.0498 (-90%)	0.0204 (-92%)
	Norm	0.1761 (-67%)	0.1226 (-64%)	0.2673 (-61%)	0.1520 (-61%)	0.0685 (-87%)	0.0398 (-88%)	0.1324 (-81%)	0.0602 (-85%)	0.0703 (-87%)	0.0414 (-88%)	0.1359 (-80%)	0.0623 (-84%)

表 A-3 在 Spattack-II-*D*攻击下的推荐性能

Dataset	Defense	5%				10%				15%			
		HR@5	nDCG@5	HR@10	nDCG@10	HR@5	nDCG@5	HR@10	nDCG@10	HR@5	nDCG@5	HR@10	nDCG@10
ML100K	Mean	0.0318 (-87%)	0.0160 (-90%)	0.0785 (-81%)	0.0309 (-85%)	0.0233 (-91%)	0.0121 (-93%)	0.0562 (-86%)	0.0225 (-89%)	0.0265 (-89%)	0.0139 (-92%)	0.0562 (-86%)	0.0231 (-89%)
	Median	0.2163 (-15%)	0.1324 (-23%)	0.3637 (-6%)	0.1797 (-16%)	0.0095 (-96%)	0.0056 (-97%)	0.0233 (-94%)	0.0099 (-95%)	0.0233 (-91%)	0.0139 (-92%)	0.0530 (-86%)	0.0234 (-89%)
	Norm	0.2418 (+2%)	0.1480 (-7%)	0.3659 (-6%)	0.1876 (-10%)	0.1516 (-36%)	0.0910 (-43%)	0.2969 (-23%)	0.138 (-33%)	0.106 (-55%)	0.0601 (-62%)	0.1877 (-52%)	0.086 (-59%)
	TrimM	0.2269 (-10%)	0.1511 (-8%)	0.3690 (-9%)	0.1968 (-8%)	0.0074 (-97%)	0.0041 (-97%)	0.0286 (-93%)	0.0110 (-95%)	0.0276 (-89%)	0.0162 (-90%)	0.0551 (-86%)	0.0249 (-88%)
	Krum	0.1474 (-23%)	0.0879 (-27%)	0.2259 (-27%)	0.1132 (-28%)	0.0159 (-92%)	0.0101 (-92%)	0.0318 (-90%)	0.0152 (-90%)	0.0286 (-85%)	0.0159 (-87%)	0.0615 (-80%)	0.0264 (-83%)
ML1M	Mean	0.0272 (-91%)	0.0153 (-93%)	0.0720 (-84%)	0.0295 (-88%)	0.0237 (-92%)	0.0128 (-94%)	0.0523 (-89%)	0.0219 (-91%)	0.0260 (-92%)	0.0146 (-93%)	0.0603 (-87%)	0.0254 (-90%)
	Median	0.0121 (-96%)	0.0055 (-97%)	0.0894 (-81%)	0.0296 (-88%)	0.024 (-92%)	0.0144 (-93%)	0.046 (-90%)	0.0215 (-92%)	0.0387 (-88%)	0.0220 (-89%)	0.0732 (-84%)	0.0330 (-87%)
	Norm	0.2805 (-8%)	0.1822 (-10%)	0.4333 (-5%)	0.2313 (-8%)	0.1962 (-36%)	0.1185 (-42%)	0.3343 (-27%)	0.1628 (-35%)	0.1119 (-63%)	0.0664 (-67%)	0.2240 (-51%)	0.1023 (-59%)
	TrimM	0.0232 (-93%)	0.0103 (-95%)	0.1611 (-65%)	0.0537 (-79%)	0.0359 (-88%)	0.0207 (-90%)	0.0717 (-85%)	0.0321 (-87%)	0.0255 (-92%)	0.0152 (-93%)	0.0503 (-89%)	0.0231 (-91%)
Steam	Mean	0.0205 (-96%)	0.0114 (-97%)	0.0384 (-94%)	0.0170 (-96%)	0.0226 (-96%)	0.0129 (-97%)	0.0520 (-93%)	0.0221 (-95%)	0.0274 (-95%)	0.0155 (-96%)	0.0592 (-91%)	0.0256 (-94%)
	Median	0.0285 (-90%)	0.0160 (-92%)	0.0549 (-89%)	0.0245 (-91%)	0.0434 (-84%)	0.0261 (-87%)	0.0874 (-83%)	0.0400 (-85%)	0.0493 (-82%)	0.0293 (-85%)	0.0906 (-82%)	0.0426 (-84%)
	Norm	0.0171 (-97%)	0.0098 (-97%)	0.0442 (-94%)	0.0184 (-95%)	0.0226 (-96%)	0.0121 (-96%)	0.0560 (-92%)	0.0227 (-94%)	0.0250 (-95%)	0.0134 (-96%)	0.0634 (-91%)	0.0256 (-94%)
	TrimM	0.0296 (-95%)	0.0166 (-96%)	0.0552 (-92%)	0.0248 (-94%)	0.0440 (-92%)	0.0263 (-93%)	0.0879 (-87%)	0.0402 (-90%)	0.0480 (-92%)	0.0285 (-93%)	0.0903 (-87%)	0.0421 (-90%)
	Krum	0.0288 (-89%)	0.0168 (-90%)	0.0554 (-88%)	0.0253 (-90%)	0.0434 (-83%)	0.0265 (-85%)	0.0866 (-81%)	0.0402 (-84%)	0.0472 (-81%)	0.0281 (-84%)	0.0895 (-81%)	0.0416 (-83%)

表 A-4 在 Spattack-II-*S*攻击下的推荐性能

Dataset	Defense	5%				10%				15%			
		HR@5	nDCG@5	HR@10	nDCG@10	HR@5	nDCG@5	HR@10	nDCG@10	HR@5	nDCG@5	HR@10	nDCG@10
ML100K	Mean	0.0742 (-70%)	0.0425 (-74%)	0.1559 (-62%)	0.0683 (-68%)	0.0456 (-82%)	0.0249 (-85%)	0.1124 (-72%)	0.0462 (-78%)	0.0477 (-81%)	0.0264 (-84%)	0.0997 (-75%)	0.0428 (-80%)
	Median	0.2238 (-12%)	0.1433 (-17%)	0.3733 (-3%)	0.1913 (-11%)	0.0308 (-88%)	0.0179 (-90%)	0.0764 (-80%)	0.0321 (-85%)	0.0339 (-87%)	0.0206 (-88%)	0.0742 (-81%)	0.0336 (-84%)
	Norm	0.2397 (+1%)	0.1525 (-4%)	0.3690 (-5%)	0.1935 (-7%)	0.2068 (-13%)	0.1243 (-22%)	0.3118 (-20%)	0.1579 (-24%)	0.1315 (-45%)	0.0804 (-50%)	0.2503 (-36%)	0.1187 (-43%)
ML1M	Mean	0.0359 (-88%)	0.0207 (-90%)	0.0917 (-80%)	0.0385 (-85%)	0.0268 (-91%)	0.0156 (-92%)	0.0641 (-86%)	0.0274 (-89%)	0.0288 (-91%)	0.0166 (-92%)	0.0679 (-85%)	0.0290 (-89%)
	Median	0.0825 (-74%)	0.039 (-81%)	0.2237 (-52%)	0.0840 (-67%)	0.0285 (-91%)	0.0169 (-92%)	0.0536 (-88%)	0.0250 (-90%)	0.0387 (-88%)	0.0229 (-89%)	0.0728 (-84%)	0.0338 (-87%)
	Norm	0.2815 (-8%)	0.1822 (-10%)	0.4368 (-4%)	0.2321 (-7%)	0.2063 (-33%)	0.1246 (-38%)	0.3464 (-24%)	0.1696 (-32%)	0.1281 (-58%)	0.0746 (-63%)	0.2445 (-46%)	0.1118 (-55%)
Steam	Mean	0.0453 (-92%)	0.0259 (-93%)	0.0901 (-87%)	0.0401 (-90%)	0.0440 (-92%)	0.0258 (-93%)	0.0914 (-87%)	0.0408 (-90%)	0.0410 (-93%)	0.0239 (-94%)	0.0890 (-87%)	0.0393 (-91%)
	Median	0.0378 (-86%)	0.0221 (-89%)	0.0791 (-84%)	0.0351 (-87%)	0.0488 (-82%)	0.0285 (-85%)	0.0927 (-82%)	0.0426 (-84%)	0.0450 (-84%)	0.0264 (-86%)	0.0938 (-81%)	0.0420 (-84%)
	Norm	0.0679 (-87%)	0.0389 (-89%)	0.1628 (-76%)	0.0692 (-82%)	0.0554 (-90%)	0.0322 (-91%)	0.1138 (-84%)	0.0508 (-87%)	0.0469 (-91%)	0.0265 (-92%)	0.1031 (-85%)	0.0444 (-89%)

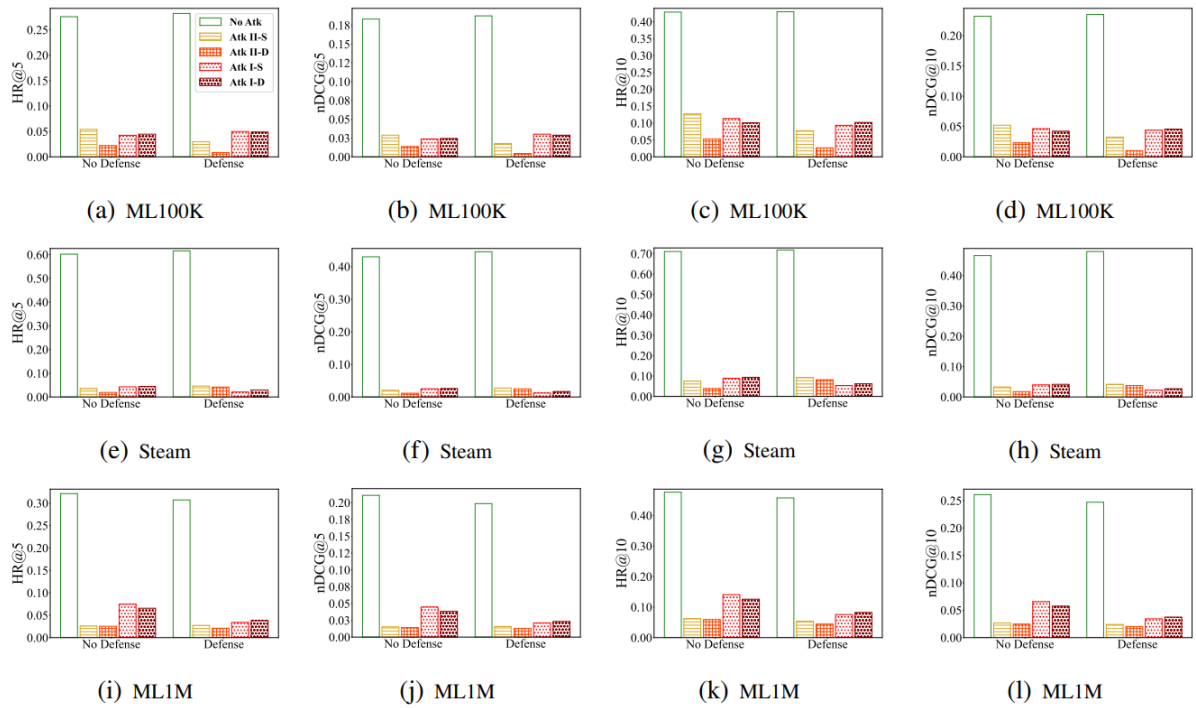


图 A-1 FedGNN 模型的实验结果

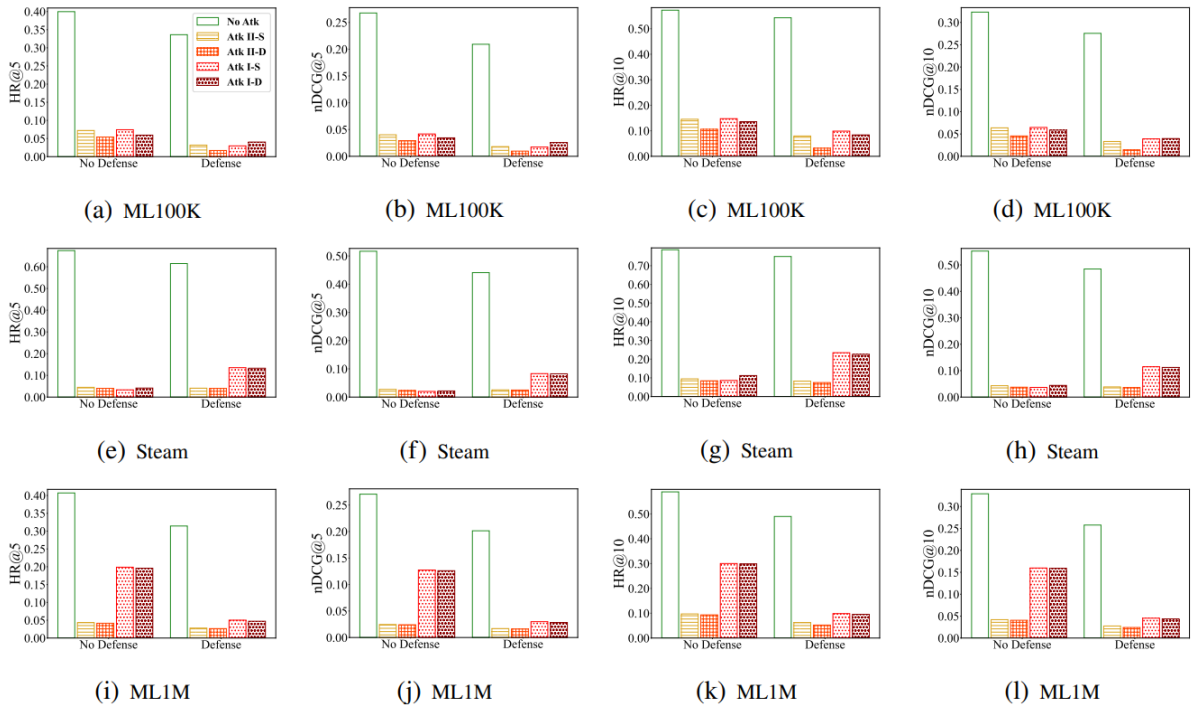


图 A-2 使用 Adam 优化器的 FR 模型实验结果