中国科学: 信息科学 2024 年 第 54 卷 第 7 期: 1569–1587
SCIENTIA SINICA Informationis
理论计算机科学专题 · 论文

面向 LinUCB 算法的数据投毒攻击方法

姜伟龙, 何琨*

华中科技大学计算机科学与技术学院, 武汉 430074
* 通信作者. E-mail: brooklet60@hust.edu.cn

收稿日期: 2023–10–19; 修回日期: 2024–01–18; 接受日期: 2024–03–18; 网络出版日期: 2024–07–09

国家自然基金 (批准号: 62076105, U22B2017) 资助项目

摘要 LinUCB 算法是求解上下文多臂老虎机问题的一种典型算法, 被广泛应用于新闻投放、产品推荐、医疗资源分配等场景中. 目前对该算法的安全性研究略显薄弱, 这就要求研究者进一步加深对该算法的攻击方式的研究, 以作出具有针对性乃至泛用性的防御措施. 本文提出了两种通过添加虚假数据的方式对 LinUCB 算法进行离线数据投毒攻击的攻击方案, 即 TCA 方案 (target context attack) 与 OCA 方案 (optimized context attack). 前者是基于训练数据与目标上下文的相似性来生成投毒数据的; 后者是建模一个优化问题, 通过求解该问题来构造投毒数据, 是前者的优化版本. 实验测试表明, 仅需添加少量投毒数据作为攻击成本即可实现对攻击目标的 100% 攻击成功率.

关键词 上下文多臂老虎机, LinUCB 算法, 数据投毒攻击, 白盒攻击, 优化问题

1 引言

多臂老虎机 (multi-armed bandit, MAB) 模型起源于 20 世纪 50 年代, 最初是 Mosteller 和 Bush 为了研究动物的学习能力而设计的实验, 后来被总结为一个经典的模型 [1]. 在该模型中, 有一台有两条手臂的老虎机和一个赌徒, 赌徒有若干次操作机会, 每次可以选择拉动任意一条手臂. 拉动某条手臂时机器会根据设定好的概率吐出一枚游戏币或不吐出游戏币, 赌徒的目标是在用完所有操作次数后获得尽可能多数量的游戏币. 由于该模型及其变体可以模拟多种现实问题, 如新闻投放 [2]、动态设定价格、广告推送 [3]、医疗决策 [4] 等场景, 自提出以来, 得到了学术界的广泛关注与研究.

上下文多臂老虎机 (contextual multi-armed bandit, CMAB) 模型 [5] 是 MAB 模型的一种经典变体. 在应用 MAB 模型的领域, 以广告推送场景为例, 为了提高广告推送的准确率与用户点击率, 研究者很自然会想到根据一些额外的信息, 如目标用户信息 (用户所在地区、用户的年龄与性别等) 和投放时间 (星期几、一天的早中晚等) 等来辅助决策, 让广告投放更加准确. 这些额外的信息被称为边缘信息, 也就是上下文 (context), 考虑了上下文的 MAB 模型就是 CMAB 模型.

引用格式: 姜伟龙, 何琨. 面向 LinUCB 算法的数据投毒攻击方法. 中国科学: 信息科学, 2024, 54: 1569–1587, doi: 10.1360/SSI-2023-0308
Jiang W L, He K. Data poisoning attacks on the LinUCB algorithm (in Chinese). Sci Sin Inform, 2024, 54: 1569–1587, doi: 10.1360/SSI-2023-0308

线性置信上界 (linear upper confidence bound, LinUCB) 算法是典型的求解 CMAB 问题的算法, 是应用在 MAB 问题上的 UCB 算法 [6] 的扩展. 在每一轮拉动手臂前, 根据奖励生成规则给所有手臂构造一个预期奖励的置信上界, 并基于面向不确定度的乐观探索原则 (optimism-in-face-of-uncertainty, OFU) 选择置信上界最大的手臂进行拉动, 在获取奖励后对手臂参数进行更新, 再进行下一轮.

随着 CMAB 模型应用范围的扩大, 其应对各种恶意攻击的能力也显得更加重要. 近年来有研究者也意识到了此问题, 开始研究对 LinUCB 算法进行攻击的方法以及考虑如何提高相应的防御能力. Ma 等 [7] 提出的一种基于凸优化的奖励投毒攻击方案, 是一种通过离线修改 LinUCB 算法训练数据中的奖励向量来实现攻击目标的方案, 后续其还将研究目标扩展到对抗性多臂老虎机上, 并证明了只需次线性的攻击成本即可实现对受害者模型的控制 [8]; Garcelon 等 [9] 提出的 CACE 攻击则是一种在线奖励投毒攻击, 同时提出了一种利用 LinUCB 算法的 “膨胀不变性” 特性的在线上下文投毒攻击; Liu 等 [10] 进一步将投毒的目标设置为 CMAB 模型的手臂集合. 为了进一步提高 LinUCB 算法应对各种恶意攻击时的安全性, 研究针对 LinUCB 算法的新型数据投毒攻击方式是必要的.

数据投毒攻击作为模型攻击的经典方法, 其具体攻击方式可以是 “修改” 受害者模型原有的训练数据, 也可以是向其中 “添加” 若干条虚假数据或从中 “删除” 若干条训练数据. 现有的研究基本都集中在通过对受害者模型的部分训练数据进行 “修改” 来实现数据投毒攻击目标, 攻击方式缺乏多样性. 针对此现状, 本文创新性地提出了两种通过向 CMAB 模型的训练数据中 “添加” 虚假数据来进行数据投毒攻击的方案, 填补了此类攻击方式在对 LinUCB 算法进行数据投毒攻击领域的空白. 主要研究和设计了投毒数据的构造方法, 分别是直接采用目标上下文作为投毒上下文的 TCA 攻击方案 (target context attack) 与基于优化问题求解并可以视为是 TCA 的优化版本的 OCA 攻击方案 (optimized context attack). 主要贡献包括以下 3 个方面:

(1) 据目前所知, 本文提出的两种针对 LinUCB 算法的数据投毒攻击方案是首次通过向受害者模型的训练数据中 “添加” 虚假数据来进行离线数据投毒攻击的方案.

(2) 首先提出了一种基于目标上下文攻击的 TCA 攻击方案. 之后对攻击过程中 LinUCB 算法每进行一次更新时预期奖励的置信上界的提高 (或降低) 量进行优化, 提出了一种基于优化问题求解的 OCA 攻击方案, 提高了攻击者每进行一次数据投毒后目标手臂对目标上下文的置信上界行进的步长, 攻击效果得到了显著提升.

(3) 分别在合成数据集与真实数据集上进行了实验. 实验结果表明, 提出的两种攻击方案均能保证对 LinUCB 算法有 100% 的攻击成功率, 且 OCA 攻击方案在两种数据集上均表现优良, 凭借较低的攻击成本即可实现攻击目标.

2 相关工作

2.1 上下文多臂老虎机问题

形式上, MAB 模型中存在两个主体: 一台老虎机和一个赌徒. 其中老虎机有 K 条手臂, 手臂集合为 $\mathcal{A}=\{1,2,\ldots,K\}$, 赌徒有 T 轮拉动手臂的机会, 其在第 t 轮拉动手臂 $a_t\in\mathcal{A}$, 老虎机会按照手臂 a_t 对应的固定概率分布 (赌徒并不知情) 给予赌徒一定的奖励 r_t. 赌徒的最终目标是得到最优的手臂序列 $S=\{a_1,a_2,\ldots,a_T\}$, 使其在 T 轮结束后能够获得最高的奖励期望 $\mathbb{E}[\sum_{t=1}^{T}r_t]$.

而 CMAB 模型则是在 MAB 模型的基础上修改了老虎机给予赌徒奖励的机制, 即赌徒获得的奖励不再是仅与拉动的手臂 a_t 有关, 还与老虎机 “观测” 到的环境 $\boldsymbol{x}_t\in\mathbb{R}^d$ (上下文) 有关. 线性 CMAB

是最常见的 CMAB 模型, 其建立在线性假设下, 即每一轮赌徒预期得到的奖励被视为老虎机在本轮观测到的上下文向量的线性映射. 具体来说, 线性 CMAB 模型中每条手臂 a 与一个参数 $\boldsymbol{\theta}_a \in \mathbb{R}^d$ 相关联, 使得对任意的轮次 t 都有

$$r_t = \boldsymbol{x}_t^{\mathrm{T}} \boldsymbol{\theta}_{a_t} + \eta_t, \tag{1}$$

其中 η_t 为环境产生奖励时附带的 σ- 次高斯噪声.

解决 CMAB 问题的核心在于平衡 "探索" 和 "利用" 两个部分 (exploitation-exploration, E&E): 赌徒需要通过 "探索" 尽可能地寻找能最大化当轮收益的手臂, 然后在 "利用" 阶段按照一定策略选择拉动某些已经 "探索" 过的手臂, 最终使自己能得到的总奖励期望最高.

2.2 LinUCB 算法

自 CMAB 模型提出以来, 国内外研究者们进行了大量算法方面的研究, 包括将原本应用在 MAB 问题上的 ϵ-Greedy 算法 [6]、UCB 算法、汤普森抽样算法 [11] 等扩展到 CMAB 问题上. 例如在线性假设下分别将 UCB 算法和汤普森抽样算法应用到线性 CMAB 问题中得到 LinUCB 算法与线性汤普森抽样算法 [12,13]. 其中 LinUCB 算法凭借其简洁高效的特点被研究者们所青睐, 是最常见的 CMAB 算法, 被广泛应用到各个领域.

在确定每轮要拉动的手臂时, LinUCB 算法会根据历史数据为拉动每条手臂可以获得的预期奖励构建一个置信上界 (upper confidence bound, UCB), 然后每轮选择置信上界最高的手臂.

定义手臂集合为 $\mathcal{A} = \{1, 2, \ldots, K\}$, 对于每一个 $a \in \mathcal{A}$, 在第 t 轮, 其训练数据 H_a 由前 $t-1$ 轮中的上下文、奖励二元组 $(\boldsymbol{x}, r)$ 组成. 令 $m_{a,t}$ 为截止至第 $t-1$ 轮手臂 a 被拉动的次数, 则有 $\sum_{a\in\mathcal{A}} m_{a,t} = t-1$; 令 $\boldsymbol{X}_{a,t} \in \mathbb{R}^{m_{a,t}\times d}$ 为该手臂的历史上下文矩阵, $\boldsymbol{X}_{a,t}$ 的每一个行向量是第 t 轮之前算法选择拉动手臂 a 的轮次中观测到的上下文向量; 令 $\boldsymbol{R}_{a,t} \in \mathbb{R}^{m_{a,t}}$ 为手臂 a 的奖励向量, 该向量中的元素是手臂 a 依次获得的奖励值.

LinUCB 算法使用如下公式对 $\hat{\boldsymbol{\theta}}_a$ 进行更新:

$$\hat{\boldsymbol{\theta}}_{a,t} = \left(\boldsymbol{X}_{a,t}^{\mathrm{T}} \boldsymbol{X}_{a,t} + \boldsymbol{I}\right)^{-1} \boldsymbol{X}_{a,t}^{\mathrm{T}} \boldsymbol{R}_{a,t},\ a \in \mathcal{A}. \tag{2}$$

根据式 (2), 算法便可通过计算 $\boldsymbol{x}_t^{\mathrm{T}} \hat{\boldsymbol{\theta}}_{a_t,t}$ 来得到每条手臂的预期奖励, 但该值与真实奖励值存在一定的误差. 有研究表明 [14], 对于 $\forall \delta > 0$, 有不小于 $1-\delta$ 的概率使式 (3) 成立:

$$|\boldsymbol{x}_t^{\mathrm{T}} \hat{\boldsymbol{\theta}}_{a_t,t} - \boldsymbol{x}_t^{\mathrm{T}} \boldsymbol{\theta}_{a_t,t}| \leqslant \alpha \|\boldsymbol{x}_t\|_{\boldsymbol{A}_{a,t}^{-1}}, \tag{3}$$

其中 $\|\boldsymbol{x}_t\|_{\boldsymbol{A}_{a,t}^{-1}} = \sqrt{\boldsymbol{x}_t^{\mathrm{T}} \boldsymbol{A}_{a,t}^{-1} \boldsymbol{x}_t}$, $\boldsymbol{A}_{a,t} = \left(\boldsymbol{X}_{a,t}^{\mathrm{T}} \boldsymbol{X}_{a,t} + \boldsymbol{I}\right)$. 参数 α 用于控制算法对 "探索" 的偏好, 在 LinUCB 算法中 $\alpha = 1 + \sqrt{\frac{1}{2}\log\frac{2}{\delta}}$, 而在改进后的 OFUL 算法 [15] 中 $\alpha_a = \sigma\sqrt{\log\left(\frac{\det(\boldsymbol{A}_a)^{\frac{1}{2}}\det(\lambda\boldsymbol{I})^{-\frac{1}{2}}}{\delta}\right)} + \lambda^{\frac{1}{2}} S$.

在第 t 轮中, 算法观测上下文 $\boldsymbol{x}_t$ 后计算各手臂对应的 UCB, 选择拉动手臂 a_t, 其中 a_t 满足

$$a_t = \arg\max_{a\in\mathcal{A}} \left\{ \boldsymbol{x}_t^{\mathrm{T}} \hat{\boldsymbol{\theta}}_{a,t} + \alpha \|\boldsymbol{x}_t\|_{\boldsymbol{A}_{a,t}^{-1}} \right\}. \tag{4}$$

LinUCB 算法的伪代码如算法 1 所示.

直觉上, 对于不经常被选择到的手臂 a, 式 (4) 中的第二项, 也就是置信半径往往很大, 从而鼓励 "探索"; 而当有的手臂已经被 "探索" 过多次且其获得的奖励反馈也较高时, 置信半径就会缩小, 但由于前一项的值较大, 因此此类手臂仍有较大可能被拉动, 此时算法就偏向于 "利用".

算法 1 LinUCB

输入: $\delta \in \mathbb{R}_+$;

for $t \Leftarrow 1, 2, 3, \ldots, T$ **do**
  观测到上下文向量 $\boldsymbol{x}_t \in \mathbb{R}^d$;
  for $a \in \mathcal{A}$ **do**
    if a 的参数未初始化 **then**
      $\boldsymbol{A}_{a,t} \Leftarrow \boldsymbol{I}_d$ (d 维单位矩阵);
      $\boldsymbol{b}_{a,t} \Leftarrow \boldsymbol{0}_d$ (d 维零向量);
    end if
    $\hat{\boldsymbol{\theta}}_{a,t} \Leftarrow \boldsymbol{A}_{a,t}^{-1}\boldsymbol{b}_{a,t}$;
    $p_{a,t} \Leftarrow \hat{\boldsymbol{\theta}}_{a,t}^{\mathrm{T}}\boldsymbol{x}_t + \alpha\sqrt{\boldsymbol{x}_t^{\mathrm{T}}\boldsymbol{A}_{a,t}^{-1}\boldsymbol{x}_t}$;
  end for
  选择 $a_t \Leftarrow \arg\max_{a\in\mathcal{A}} p_{a,t}$ 进行拉动, 环境给予奖励 r_t;
  $\boldsymbol{A}_{a_t,t+1} \Leftarrow \boldsymbol{A}_{a_t,t} + \boldsymbol{x}_t\boldsymbol{x}_t^{\mathrm{T}}$;
  $\boldsymbol{b}_{a_t,t+1} \Leftarrow \boldsymbol{b}_{a_t,t} + r_t\boldsymbol{x}_t$;
end for

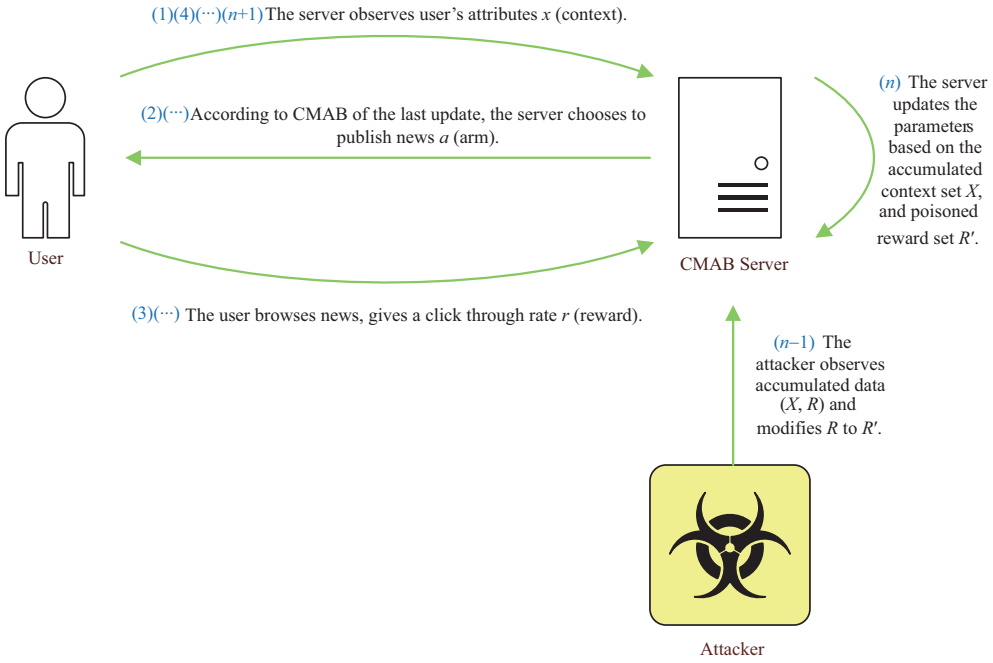

图 1 (网络版彩图) 对 CMAB 模型进行离线奖励投毒攻击

Figure 1 (Color online) Offline data poisoning attacks on rewards of CMAB

2.3 面向 CMAB 模型的数据投毒攻击方案

对 CMAB 模型进行数据投毒攻击的研究直到 2018 年才出现. Ma 等 [7] 提出了一个基于凸优化的离线攻击框架, 并表明只要稍微修改训练数据中的奖励部分, 攻击者就可以强迫 LinUCB 算法在面对指定的目标上下文向量时拉动特定的目标手臂. 图 1 展示了攻击者在对 "周期更新" 假设 [7] 下的 CMAB 模型进行离线奖励投毒攻击的攻击流程.

2020 年, 在 Ma 等提出的离线攻击框架下, Garcelon 等提出了对 CMAB 模型进行在线投毒攻击

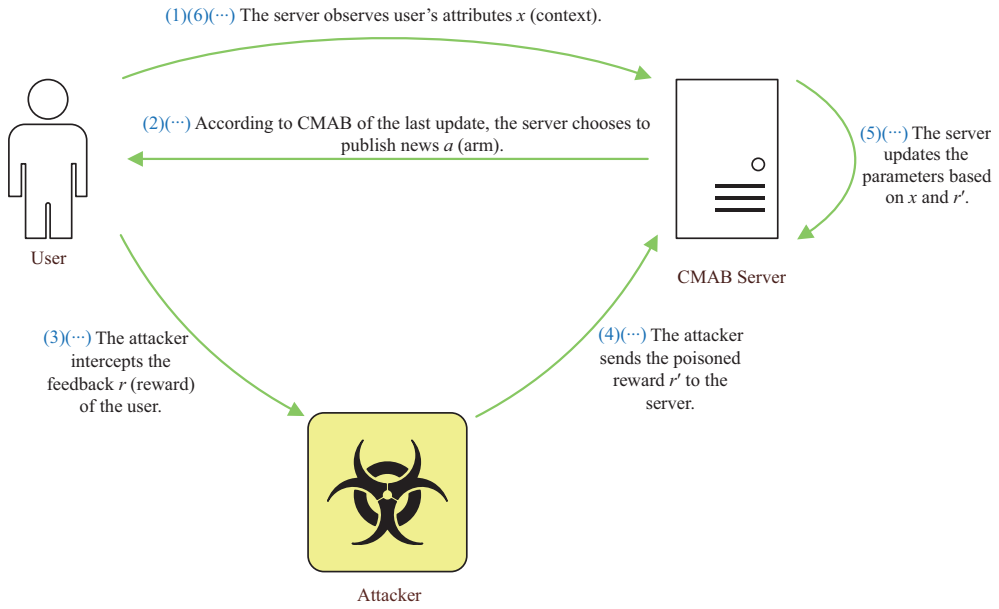

图 2 (网络版彩图) 对 CMAB 模型进行在线奖励投毒攻击

Figure 2 (Color online) Online data poisoning attacks on rewards of CMAB

的方式, 分别针对奖励有界与奖励无界两种假设提出了有效的攻击方案, 实现了在算法运行 T 轮的情况下, 可以通过花费 $o(T)$ 的成本来使算法拉动目标手臂的次数达到 $T-o(T)$ 的数量级. 同时, 该研究还将 Liu 等 [16] 提出的针对 SMAB (stochastic MAB) 问题的 ACE (adaptive attack by constant estimation) 攻击方案扩展到 CMAB 模型上, 提出了 CACE (contextual ACE) 攻击方案. 对 CMAB 模型进行在线奖励投毒攻击的流程如图 2 所示.

对 CMAB 模型中的上下文部分进行投毒攻击的研究相对较少. Garcelon 等提出了一个针对 CMAB 模型的上下文进行在线攻击的方案, 通过每一轮将不能使算法拉动目标手臂的上下文按照一定比例进行 "膨胀" (可以视为一种投毒操作), 利用 LinUCB 算法的 "膨胀不变性", 使算法参数随着更新逐渐向攻击者所希望的方向靠拢, 最终实现攻击目标, 攻击流程如图 3 所示.

Garcelon 等的研究中还提到了一种针对上下文进行离线攻击的方法, 但该方法要求只能修改一个用户对应的上下文, 且无法确保攻击能否成功, 因此只分析了方案的可行性条件. 攻击流程可以类比图 1, 只是此处攻击者修改的是累积上下文矩阵 $\boldsymbol{X}$ 而非奖励向量 $\boldsymbol{R}$.

Liu 等将投毒攻击的对象再次扩展, 将目标定为了 CMAB 算法每轮选择拉动的手臂. 直觉上, 对手臂投毒攻击显然存在一定的限制, 因为在 CMAB 模型中手臂集合一般是有限集, 对手臂进行投毒可能会导致毒化后的手臂不是一条合法手臂, 即不在原本的手臂集合中. 但 Liu 等设计的攻击方案恰巧规避了这一点, 限制攻击目标中的目标手臂只能设置为非最劣臂, 其中最劣臂指的是满足以下条件的手臂 a: 在上下文集合中至少存在一条上下文 $\boldsymbol{x}$ 使得算法在观测到 $\boldsymbol{x}$ 时, 手臂 a 的 UCB 值在所有手臂中最小. 当攻击者发现某个用户上下文向量无法使算法拉动目标手臂时, 攻击者会按照一定概率将算法拉动的手臂强制修改为最劣手臂, 自然模型在该轮获得的奖励也被迫改变, 从而达到毒化训练数据的目的. 算法使用中毒的训练数据更新手臂参数时, 拉动该手臂所能获得的预期奖励期望会因为有概率被攻击而产生变化, 又因为可能被强制拉动的手臂是最劣臂, 所以拉动非目标臂时的奖励期望低于其未被攻击时的奖励期望, 在 T 轮攻击过程中算法就会慢慢偏向于拉动目标手臂, 进而实现攻击目

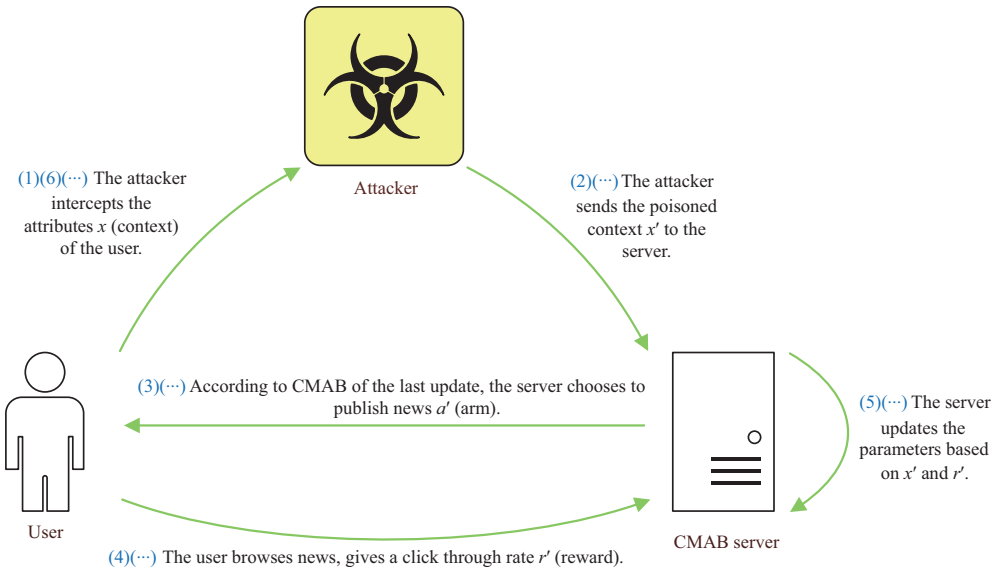

图 3　(网络版彩图) 对 CMAB 模型进行在线上下文投毒攻击

Figure 3　(Color online) Online data poisoning attacks on contexts of CMAB

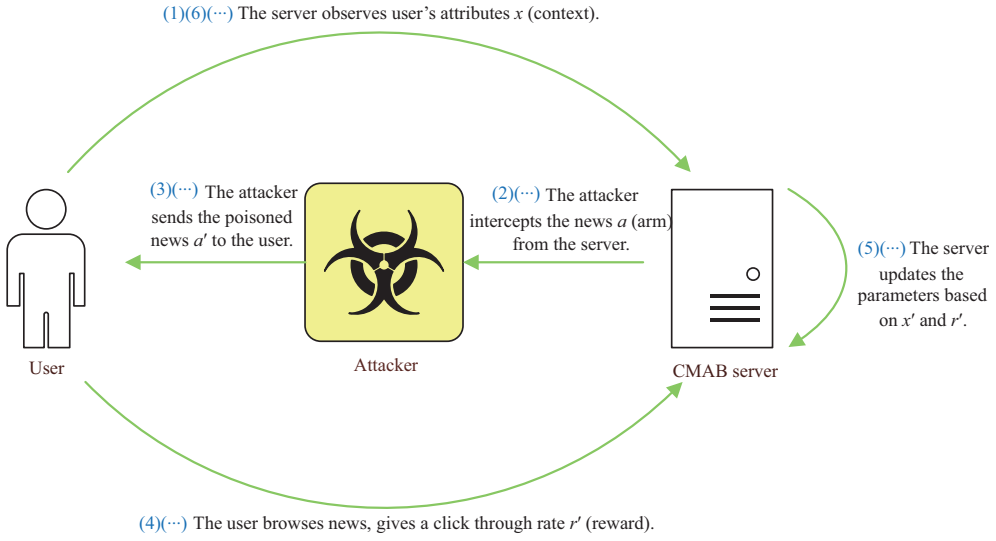

图 4　(网络版彩图) 对 CMAB 模型进行在线手臂投毒攻击

Figure 4　(Color online) Online data poisoning attacks on arms of CMAB

标. 攻击流程如图 4 所示.

3　基于添加虚假数据的 LinUCB 算法数据投毒攻击方案

本节首先说明了被攻击的 LinUCB 算法需要满足的假设条件与本文设计的攻击方案需要实现的攻击目标, 然后设计了两种不同的方法来构造投毒所需的虚假数据, 并对设计中的关键步骤给出了详细证明.

3.1 攻击假设与流程设计

3.1.1 LinUCB 算法前置假设

本文中作为数据投毒攻击目标的 LinUCB 算法均满足以下假设:

(1) 存在上下文集合 $\mathcal{X} \subset \mathbb{R}^d$, 使得对于所有的算法运行轮次 t, 有 $\boldsymbol{x}_t \in \mathcal{X}$; 且 $\forall \boldsymbol{x} \in \mathcal{X}$, 有 $\|\boldsymbol{x}\|_2 = 1$.

(2) 存在手臂集合 $\mathcal{A} = \{1, 2, \ldots, K\}$, 对于 $\forall a \in \mathcal{A}$, 有 $\boldsymbol{\theta}_a \in \mathbb{R}^d$, 且 $\|\boldsymbol{\theta}_a\|_2 = 1$, 则根据柯西 – 施瓦茨 (Cauchy-Schwartz) 不等式有 $\forall \boldsymbol{x} \in \mathcal{X}$, $\forall a \in \mathcal{A}$, 有 $|\langle \boldsymbol{x}, \boldsymbol{\theta}_a \rangle| \leqslant \|\boldsymbol{x}\|_2 \cdot \|\boldsymbol{\theta}\|_2 \leqslant 1$.

(3) 环境给予奖励反馈 $r = \langle \boldsymbol{x}, \boldsymbol{\theta} \rangle + \eta$ 时自动进行 clip 操作, 其中 $\mathrm{clip}(r) = \min\{\max\{-1, r\}, 1\}$, 以保证奖励 $|r| \leqslant 1$.

(4) 算法观测到的上下文 $\boldsymbol{x}$ 在 $\mathcal{X}$ 上是均匀分布的.

令上下文向量 $\boldsymbol{x}$ 与手臂参数 $\boldsymbol{\theta}$ 的 ℓ_2 范数均为 1 是合理的, 因为 LinUCB 算法对于 $\boldsymbol{x}$ 与 $\boldsymbol{\theta}$ 具有 “膨胀不变性”: 假设算法观测到上下文 $\boldsymbol{x}$ 时选择拉动手臂 a, 则当算法观测到另一个上下文向量 $n\boldsymbol{x}$, 其中 $n > 0$ 时, 也将同样拉动手臂 a.

3.1.2 攻击目标

面向 CMAB 模型的数据投毒攻击分为在线攻击与离线攻击, 二者的攻击目标尽管都希望算法在观测到某个攻击者指定的目标上下文向量 (记为 $\boldsymbol{x}^*$) 后拉动攻击者指定的目标手臂 (记为 a^*), 但细节有所不同:

(1) 在线攻击目标. 攻击者在算法运行的每一轮都对数据进行投毒, 使得算法在攻击者参与的 T 轮中, 满足 “观测到 $\boldsymbol{x}^*$ 后拉动 a^*” 的轮数的数量级为 $O(T)$.

(2) 离线攻击目标. 离线攻击一般都建立在 “周期更新” 的假设下. 攻击者在算法的两次更新期间完成数据投毒攻击, 随后算法使用被毒化的训练数据进行参数更新, 更新后的算法在下一次更新参数前观测到 $\boldsymbol{x}^*$ 时, 必须拉动 a^*.

本文研究的是对 LinUCB 算法进行离线数据投毒攻击的方法.

3.1.3 攻击成本

本文设计的攻击方案均使用 “添加” 虚假数据的方式进行, 即向受害者模型不同手臂 a 的训练数据中插入若干条精心设计的虚假数据, 直至实现攻击目标. 自然而然地考虑到可以使用 “使攻击成功的最少投毒数据条数” 作为攻击成本来衡量攻击性能, 但通过实验观察到投毒数据的条数会受到原始训练数据规模的影响, 因此使用投毒数据占原始训练数据的比例作为攻击成本更便于观察实验现象: $c = \sqrt{\frac{\|\boldsymbol{X}_\Delta\|_F^2 + \|\boldsymbol{R}_\Delta\|_F^2}{\|\boldsymbol{X}\|_F^2 + \|\boldsymbol{R}\|_F^2}}$. 其中, $\boldsymbol{X}_\Delta$ 与 $\boldsymbol{R}_\Delta$ 分别是上下文投毒矩阵与奖励投毒向量, $\boldsymbol{X}$ 与 $\boldsymbol{R}$ 分别是投毒前的原始上下文矩阵与奖励向量, $\|\cdot\|_F$ 为矩阵的 F 范数.

3.1.4 攻击方案流程设计

本文的离线数据投毒攻击流程如图 5 所示, 其中设计攻击方案的核心在于构造投毒数据 Δ, 形式上不同手臂 a 的 Δ_a 由上下文矩阵、奖励向量二元组 $(\Delta\boldsymbol{X}_a, \Delta\boldsymbol{R}_a)$ 组成. 接下来第 3.2 与 3.3 小节分别提出基于目标上下文与基于优化问题求解的投毒数据构造方案, 分别对应 TCA 攻击方案与 OCA 攻击方案.

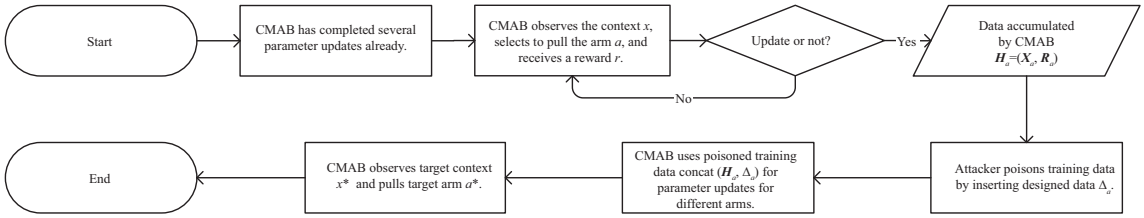

图 5　攻击方案流程设计

Figure 5　Process design of attack schemes

3.2　TCA 攻击方案

文献 [7] 中提出了奖励投毒攻击方案 (reward attack, RA): 攻击者对手臂 a 的训练数据中的奖励向量 $\boldsymbol{R}_a$ 进行投毒 (修改), 通过求解凸优化问题来构造投毒数据, 记为 $\Delta_{\boldsymbol{R}_a}$. 投毒后的奖励向量变为 $\boldsymbol{R}_a+\Delta_{\boldsymbol{R}_a}$, 也就是对于原本手臂 a 的第 i 条训练数据 $(\boldsymbol{X}_a^{(i)},\boldsymbol{R}_a^{(i)})$, 投毒后变为 $(\boldsymbol{X}_a^{(i)},\boldsymbol{R}_a^{(i)}+\Delta_{\boldsymbol{R}_a}^{(i)})$.

记攻击前在观测到 $\boldsymbol{x}^*$ 时 UCB 高于 a^* 的手臂集合为 A^+, $\langle\cdot,\cdot\rangle$ 为两个向量的内积, 文献 [7] 的实验部分给出了以下两条结论:

(1) 训练数据中奖励值的修改导致 a^* 的 UCB 值得到提高, 而 a^+ 的 UCB 值在受到攻击后降低, 最终满足受害者模型在被攻击后观测到 $\boldsymbol{x}^*$ 时 a^* 的 UCB 值最大.

(2) 当攻击成功时, a^* 对应的投毒数据中, $\Delta_{\boldsymbol{R}_a}^{(i)}$ 与 $\langle\boldsymbol{X}_a^{(i)},\boldsymbol{x}^*\rangle$ 呈正相关; $a^+\in A^+$ 对应的投毒数据中, $\Delta_{\boldsymbol{R}_a}^{(i)}$ 与 $\langle\boldsymbol{X}_a^{(i)},\boldsymbol{x}^*\rangle$ 呈负相关; 而攻击前 UCB 低于 a^* 的手臂训练数据无需进行投毒.

根据上述结论, 自然而然地想到: 可以通过向 $a^*(/a^+)$ 的训练数据中添加能最大化 (/最小化) $\langle\boldsymbol{x},\boldsymbol{x}^*\rangle$ 的 $\boldsymbol{x}$ 与 $|r|_{\max}$ 组成的虚假数据来提高 (/降低) 对应手臂的 UCB 值, 进而实现攻击目标. 而根据 3.1.1 小节所做假设, 有 $+(/-)\boldsymbol{x}^*=\arg\max(/\arg\min)_{\boldsymbol{x}}\langle\boldsymbol{x},\boldsymbol{x}^*\rangle$, $|r|_{\max}=1$. 基于上述思路, 本节提出了基于目标上下文的投毒数据构造方案: 对于手臂 a^*, 直接使用 $(\boldsymbol{x}^*,1)$ 作为投毒数据; 对于手臂 a^+, 直接使用 $(-\boldsymbol{x}^*,1)$ 作为投毒数据. 并据此, 提出了 TCA 攻击方案, 其具体运行流程如下:

(1) 攻击者基于受害者模型的参数与原本的训练数据模拟参数更新.

(2) 计算 a^* 的 UCB 值, 然后遍历其他手臂, 若出现 UCB 值高于 a^* 的手臂 a^+, 则向受害者模型的手臂 a^+ 的训练数据中添加一条由 $(-\boldsymbol{x}^*,1)$ 组成的投毒数据, 攻击者则继续模拟受害者模型使用投毒数据更新参数.

(3) 遍历所有手臂后, 判断 a^* 的 UCB 值是否最大. 若是, 则攻击成功; 否则向受害者模型的手臂 a^* 的训练数据中添加一条由 $(\boldsymbol{x}^*,1)$ 组成的投毒数据, 攻击者则继续模拟受害者模型使用投毒数据更新后的参数.

(4) 重复步骤 (2) 和 (3), 直至实现攻击目标.

TCA 算法的伪代码如算法 2 所示.

3.3　OCA 攻击方案

3.3.1　设计思路

从 LinUCB 算法的运行流程上看, 尽管式 (2) 中的训练数据用矩阵表示, 但具体实现 (算法 1) 上采用的是每轮更新 (每次更新一条数据) 的方式. 因此向 LinUCB 算法的训练数据中添加虚假数据来进行的投毒攻击本质上可以视为凭空让算法额外再运行若干轮, 而攻击者就可以在这关键的轮次中实施攻击来实现目标.

算法 2 TCA

输入: LinUCB 算法参数 $\hat{\boldsymbol{\theta}}_a$, $\boldsymbol{A}_a$, $\boldsymbol{b}_a$, 手臂数量 K, 训练数据 $\boldsymbol{X}_a$, $\boldsymbol{R}_a$, $a \in \mathcal{A}$, 目标上下文 $\boldsymbol{x}^*$, 目标手臂 a^*;

for $a \Leftarrow 1, 2, 3, \ldots, K$ **do**
  基于原参数 $\hat{\boldsymbol{\theta}}_a$, $\boldsymbol{A}_a$, $\boldsymbol{b}_a$ 模拟 LinUCB 算法使用 $\boldsymbol{X}_a$, $\boldsymbol{R}_a$ 进行参数更新, 记录更新后的参数 $\hat{\boldsymbol{\theta}}'_a$, $\boldsymbol{A}'_a$, $\boldsymbol{b}'_a$;
end for
while True **do**
  基于 $\hat{\boldsymbol{\theta}}'_{a^*}$, $\boldsymbol{A}'_{a^*}$, $\boldsymbol{b}'_{a^*}$ 计算 UCB_{a^*};
  for $a \Leftarrow 1, 2, 3, \ldots, K$ **do**
    基于 $\hat{\boldsymbol{\theta}}'_a$, $\boldsymbol{A}'_a$, $\boldsymbol{b}'_a$ 计算 UCB_a;
    if $a \neq a^*$ and $\mathrm{UCB}_a > \mathrm{UCB}_{a^*}$ **then**
      将 $(-\boldsymbol{x}^*, 1)$ 添加到受害者模型手臂 a 的训练数据中;
      攻击者模拟 LinUCB 算法使用 $(-\boldsymbol{x}^*, 1)$ 更新手臂 a 的参数, 并更新 $\hat{\boldsymbol{\theta}}'_a$, $\boldsymbol{A}'_a$, $\boldsymbol{b}'_a$;
    end if
  end for
  if $a^* = \arg\max_a \mathrm{UCB}_a$ **then**
    break;
  else
    将 $(\boldsymbol{x}^*, 1)$ 添加到受害者模型手臂 a^* 的训练数据中;
    攻击者模拟 LinUCB 算法使用 $(\boldsymbol{x}^*, 1)$ 更新手臂 a^* 的参数, 并更新 $\hat{\boldsymbol{\theta}}'_{a^*}$, $\boldsymbol{A}'_{a^*}$, $\boldsymbol{b}'_{a^*}$;
  end if
end while

基于 TCA 的核心思路 (提高 a^* 的 UCB 值的同时降低 a^+ 的 UCB 值, 直到 a^* 的 UCB 值为所有手臂中的最大值) 与贪婪的思想, 将投毒数据的构造问题建模为如下的优化问题: 在 LinUCB 算法使用正确的训练数据更新参数后, 对于 a^* 求解能最大化下一轮 UCB 值的训练数据, 对于 a^+ 求解能最小化下一轮 UCB 值的训练数据, 其中约束条件为 $\|\boldsymbol{x}\|_2 = 1$. 上述优化问题可以使用如下公式形式化地给出:

$$\begin{aligned} \max_{a=a^*} \quad & f(\boldsymbol{x}) = \hat{\boldsymbol{\theta}}_{a,t+1}^{\mathrm{T}} \boldsymbol{x}^* + \alpha \sqrt{\boldsymbol{x}^{*\mathrm{T}} \boldsymbol{A}_{a,t+1}^{-1} \boldsymbol{x}^*} \\ \text{其中} \quad & \|\boldsymbol{x}\|_2 = 1, \\ & \hat{\boldsymbol{\theta}}_{a,t+1} = \left(\boldsymbol{A}_{a,t} + \boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\right)^{-1} \left(\boldsymbol{b}_{a,t} + \boldsymbol{x}\right), \end{aligned} \tag{5}$$

$$\begin{aligned} \min_{a\neq a^*} \quad & f(\boldsymbol{x}) = \hat{\boldsymbol{\theta}}_{a,t+1}^{\mathrm{T}} \boldsymbol{x}^* + \alpha \sqrt{\boldsymbol{x}^{*\mathrm{T}} \boldsymbol{A}_{a,t+1}^{-1} \boldsymbol{x}^*} \\ \text{其中} \quad & \|\boldsymbol{x}\|_2 = 1, \\ & \hat{\boldsymbol{\theta}}_{a,t+1} = \left(\boldsymbol{A}_{a,t} + \boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\right)^{-1} \left(\boldsymbol{b}_{a,t} + \boldsymbol{x}\right). \end{aligned} \tag{6}$$

基于上述思路, 本小节提出了基于求解优化问题的投毒数据构造方案: 对于手臂 a^+, 通过求解问题 (5) 的最大值点来获取投毒上下文, 再结合 $|r|_{\max}$ 来组成投毒数据, 而对于手臂 a^+, 投毒上下文则是通过求解问题 (6) 的最小值点来获取的. 具体的算法运行流程与 TCA 类似, 只不过其中的投毒上下文获取方式不同.

3.3.2 求解 OCA 攻击中的优化问题

OCA 中待求解的两个优化问题 (5) 和 (6) 结合起来可视为求解 $f(x)$ 的最值点. 但目标函数中存在大量矩阵与向量形式的参数, 难以直接求解, 因此本小节将求解原优化问题转化为求其近似解并最

终简化为求解一元四次方程的问题. 由于上述两个优化问题在结构上是对称的, 因此只需讨论其中之一, 下文的推导过程以优化问题 (5) 为例.

目标函数 $f(x)$ 中较难处理的是矩阵和求逆的部分, 即 $\hat{\boldsymbol{\theta}}_{a,t+1}$ 的第一个因式 $\left(\boldsymbol{A}_{a,t}+\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\right)^{-1}$. 有研究 [17] 给出了该式的展开方法:

$$\left(\boldsymbol{A}_{a,t}+\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\right)^{-1}=\boldsymbol{A}_{a,t}^{-1}-\frac{\boldsymbol{A}_{a,t}^{-1}\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_{a,t}^{-1}}{1+\mathrm{tr}\left(\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_{a,t}^{-1}\right)}. \tag{7}$$

式 (7) 中的分数部分含有矩阵的迹的形式, 为了方便推导, 将其转化为二次项表示, 转化过程依据以下定理.

定理1 $\forall t\in 1,2,\ldots,T$, $\forall a\in\mathcal{A}$, $\boldsymbol{A}_{a,t}^{-1}$ 为实对称矩阵.

证明 由算法 1 可得对于 $\forall t\in 1,2,\ldots,T$, $\forall a\in\mathcal{A}$, $\boldsymbol{A}_{a,t}$ 可以写成如下形式:

$$\boldsymbol{A}_{a,t}=\boldsymbol{I}_d+\sum_{i=1}^{t-1}\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}. \tag{8}$$

易知 $\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}$ 是实对称矩阵, 由实对称矩阵的定义可知实对称矩阵之和与实对称矩阵之逆均为实对称矩阵, 因此 $\boldsymbol{A}_{a,t}$ 与 $\boldsymbol{A}_{a,t}^{-1}$ 均为实对称矩阵.

定理2 对于任意实对称矩阵 $\boldsymbol{A}\in\mathbb{R}^{d\times d}$ 与 $\boldsymbol{x}\in\mathbb{R}^d$, 有 $\mathrm{tr}(\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A})=\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}$.

根据式 (7) 与定理 1 和 2 可得

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{a,t+1}^{\mathrm{T}}\boldsymbol{x}^{*}&=\boldsymbol{x}^{*\mathrm{T}}\left(\boldsymbol{A}_{a,t}^{-1}-\frac{\boldsymbol{A}_{a,t}^{-1}\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_{a,t}^{-1}}{1+\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_{a,t}^{-1}\boldsymbol{x}}\right)(\boldsymbol{b}_{a,t}+\boldsymbol{x})\\&=\boldsymbol{x}^{*\mathrm{T}}\boldsymbol{A}_{a,t}^{-1}\boldsymbol{b}_{a,t}+\boldsymbol{x}^{*\mathrm{T}}\boldsymbol{A}_{a,t}^{-1}\boldsymbol{x}-\boldsymbol{x}^{*\mathrm{T}}\frac{\boldsymbol{A}_{a,t}^{-1}\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_{a,t}^{-1}}{1+\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_{a,t}^{-1}\boldsymbol{x}}\boldsymbol{b}_{a,t}-\boldsymbol{x}^{*\mathrm{T}}\frac{\boldsymbol{A}_{a,t}^{-1}\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_{a,t}^{-1}}{1+\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_{a,t}^{-1}\boldsymbol{x}}\boldsymbol{x}.\end{aligned} \tag{9}$$

定理3 $\forall t\in 1,2,\ldots,T$, $\forall a\in\mathcal{A}$, $\boldsymbol{A}_{a,t}^{-1}$ 为正定矩阵.

证明 $\forall\boldsymbol{x}\in\mathbb{R}^d$ 且 $\boldsymbol{x}\neq\boldsymbol{0}$, 有

$$\begin{aligned}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_{a,t}\boldsymbol{x}&=\boldsymbol{x}^{\mathrm{T}}\left(\boldsymbol{I}_d+\sum_{i=1}^{t-1}\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}\right)\boldsymbol{x}\\&=\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}+\sum_{i=1}^{t-1}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}\\&=\|\boldsymbol{x}\|_2^2+\sum_{i=1}^{t-1}\left(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}_i\right)^2\\&>0\end{aligned} \tag{10}$$

恒成立.

由定理 1 可知 $\boldsymbol{A}_{a,t}$ 为对称矩阵, 故 $\boldsymbol{A}_{a,t}$ 是正定矩阵, 又由于正定矩阵的逆矩阵也是正定矩阵, 可得 $\boldsymbol{A}_{a,t}^{-1}$ 为正定矩阵.

定理4 (Rayleigh-Ritz 定理 [18]) 设 $\boldsymbol{A}$ 为厄米特矩阵 (Hermitian matrix), $\lambda_{\min}$ 代表 $\boldsymbol{A}$ 的最小特征值, $\lambda_{\max}$ 代表 $\boldsymbol{A}$ 的最大特征值. 则对于任意非零向量 $\boldsymbol{x}$, 有以下不等式成立:

$$\lambda_{\min}\leqslant\frac{\boldsymbol{x}^{\mathrm{H}}\boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^{\mathrm{H}}\boldsymbol{x}}\leqslant\lambda_{\max}. \tag{11}$$

根据定理 4 可直接得到以下推论.

推论1 二次型 $f=\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}$ 在 $\|\boldsymbol{x}\|_2=1$ 时的最大值与最小值分别为 $\boldsymbol{A}$ 的最大特征值与最小特征值, 其中 $\boldsymbol{A}$ 为正定对称矩阵.

算法 1 中 $\boldsymbol{A}_a$ 的初始值为 $\boldsymbol{I}_d$, 特征值为 1, 其更新方式可以视为一种增量更新, 每轮的增量为 $\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}$. 由于本文做了 $\boldsymbol{x}$ 的 "均匀分布" 假设, 则当手臂 a 经历 $t\gg 1$ 轮参数更新后, $\boldsymbol{A}_{a,t}$ 的特征值将远大于 1. 又由于 $\boldsymbol{A}_{a,t}^{-1}$ 的特征值等于 $\boldsymbol{A}_{a,t}$ 的特征值的倒数, 则 $\boldsymbol{A}_{a,t}^{-1}$ 的特征值将远小于 1, 此时结合定理 3 与推论 1 有

$$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_{a,t}^{-1}\boldsymbol{x}\ll 1. \tag{12}$$

据此, 式 (9) 中第 3 和 4 项的分母均可近似为 1, $f(\boldsymbol{x})$ 的最后一项置信项也可以直接舍去, 同时由于目标是求解 $f(\boldsymbol{x})$ 的最值点, 其第一项常数项也可不予考虑, 最终原优化问题可被简化为如下形式:

$$\begin{aligned}&\max_{a=a^*} \quad g(\boldsymbol{x})=\boldsymbol{x}^{*\mathrm{T}}\boldsymbol{A}_{a,t}^{-1}\boldsymbol{x}(1-\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_{a,t}^{-1}\boldsymbol{b}_{a,t})\\ &\text{其中} \quad \|\boldsymbol{x}\|_2=1.\end{aligned} \tag{13}$$

令 $\boldsymbol{\mu}^{\mathrm{T}}=\boldsymbol{x}^{*\mathrm{T}}\boldsymbol{A}_{a,t}^{-1}$, $\boldsymbol{\nu}=\boldsymbol{A}_{a,t}^{-1}\boldsymbol{b}_{a,t}$, 代入 $g(\boldsymbol{x})$ 可得 $g(\boldsymbol{x})=\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{x}\left(1-\boldsymbol{\nu}^{\mathrm{T}}\boldsymbol{x}\right)$. 联合约束条件 $\varphi(\boldsymbol{x})=\|\boldsymbol{x}\|_2-1=0$, 使用拉格朗日 (Lagrange) 乘数法可构造如下函数:

$$\mathcal{L}(\boldsymbol{x},\lambda)=\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{x}\left(1-\boldsymbol{\nu}^{\mathrm{T}}\boldsymbol{x}\right)-\frac{1}{2}\lambda\left(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}-1\right), \tag{14}$$

其中 $-\frac{1}{2}\lambda$ 为待定系数.

$\mathcal{L}(\boldsymbol{x},\lambda)$ 分别对 $\boldsymbol{x}$ 和 λ 求一阶偏导, 则 $g(\boldsymbol{x})$ 所有的极值点就在方程组 (15) 的解 $(\boldsymbol{x}_0,\lambda_0)$ 对应的 $\boldsymbol{x}_0$ 上.

$$\begin{cases}\mathcal{L}_{\boldsymbol{x}}(\boldsymbol{x},\lambda)=\boldsymbol{\mu}-\left(\boldsymbol{\mu}\boldsymbol{\nu}^{\mathrm{T}}+\boldsymbol{\nu}\boldsymbol{\mu}^{\mathrm{T}}\right)\boldsymbol{x}-\lambda\boldsymbol{x}=0,\\ \mathcal{L}_{\lambda}(\boldsymbol{x},\lambda)=\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}-1=0.\end{cases} \tag{15}$$

考虑方程组 (15) 的一式, 若方程有解, 则 $\boldsymbol{x}$ 一定可以表示为如下形式:

$$\boldsymbol{x}=k_1\boldsymbol{\mu}+k_2\boldsymbol{\nu}. \tag{16}$$

因为方程组 (15) 的一式中的 $\boldsymbol{\mu}-\left(\boldsymbol{\mu}\boldsymbol{\nu}^{\mathrm{T}}+\boldsymbol{\nu}\boldsymbol{\mu}^{\mathrm{T}}\right)$ 可以写成 $\left(1-\boldsymbol{\nu}^{\mathrm{T}}\boldsymbol{x}\right)\boldsymbol{\mu}-\left(\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{x}\right)\boldsymbol{\nu}$ 的形式, 而该向量在 $\boldsymbol{\mu}$ 和 $\boldsymbol{\nu}$ 张成的空间中, 因此向量 $\lambda\boldsymbol{x}$ 也应该在此空间中.

令 $u=\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\mu}$, $v=\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\nu}$, $w=\boldsymbol{\nu}^{\mathrm{T}}\boldsymbol{\nu}$, 将式 (16) 代入可得

$$\begin{cases}\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{x}=k_1u+k_2v,\\ \boldsymbol{\nu}^{\mathrm{T}}\boldsymbol{x}=k_1v+k_2w.\end{cases} \tag{17}$$

将式 (16) 和 (17) 代回方程组 (15), 再结合线性规则可得

$$\begin{cases}1-k_1v-k_2w=\lambda k_1,\\ -k_1u-k_2v=\lambda k_2,\\ k_1^2u+2k_1k_2v+k_2^2w=1.\end{cases} \tag{18}$$

联立方程组 (18) 的前两式, 令 $s=\lambda+v$, 可得

$$\begin{cases} k_1=\dfrac{s}{s^2-uw}, \\ k_2=\dfrac{-u}{s^2-uw}. \end{cases} \tag{19}$$

将其代入方程组 (18) 的三式并化简, 可得关于 s 的一元四次方程:

$$s^4-(2uw+u)\,s^2+2uvs+u^2w^2-u^2w=0. \tag{20}$$

根据天珩公式, 将其表示成 $as^4+bs^3+cs^2+ds+e=0$ 的形式, 其中 a, b, c, d, $e\in\mathbb{R}$, 且 $a\neq 0$, 则有 $a=1$, $b=0$, $c=-(2uw+u)$, $d=2uv$, $e=u^2w^2-u^2w$, 给出下列重根判别式参数:

$$\begin{cases} D=3b^2-8ac=8u\,(2w+1)\,, \\ E=-b^3+4abc-8a^2b=-16uv, \\ F=3b^4+16a^2c^2-16ab^2c+16a^2bd-64a^3e=16u^2\,(8w+1)\,, \\ A=D^2-3F=16u^2\,(4w-1)^2\,, \\ B=DF-9E^2=128u^2\left(16uw^2+10uw+u-18v^2\right), \\ C=F^2-3DE^2=256u^4\,(8w+1)^2-6144u^3v^2\,(2w+1)\,, \\ \Delta=B^2-4AC. \end{cases} \tag{21}$$

一元四次方程无实根需要满足以下任一条件:

(1) $E=F=0$ 且 $D<0$;

(2) $\Delta<0$ 且 D 与 F 不全为正数.

根据方程 (20) 中系数的特点 $-u\geqslant 0$ 且 $w\geqslant 0$, 可知 $D\geqslant 0$ 与 $F\geqslant 0$ 恒成立, 则条件 (1) 必不满足; 对于条件 (2), 当且仅当 $u=0$ 时有 D, F 不全为正数, 此时二者同时为 0, 但此时 E, A, B, C 也均为 0, 即 Δ 也为 0, 亦不满足条件. 因此方程 (20) 必有实数解.

方程 (20) 的求解可以使用多种方法, 如牛顿 – 拉夫森迭代法 (Newton-Raphson method)、费拉里法 (Ferrari method) [19]、天珩公式 [20] 等. 之后将所得 s 代回式 (19) 得到 k_1 与 k_2, 再将 k_1 与 k_2 代回式 (16) 即可得到 $g(\boldsymbol{x})$ 的极值点 $\boldsymbol{x}_0$.

由于 $g(\boldsymbol{x})$ 在 $\|\boldsymbol{x}\|_2=1$ 上连续, 因此其最值点就是其极值点, 将所有极值点代入 $g(\boldsymbol{x})$, 排序后即可得到 $g(\boldsymbol{x})$ 的最大值与最小值, 对应的 $\boldsymbol{x}$ 也就是最大值点与最小值点, 也即 OCA 所要求的投毒上下文向量. 其中对于 a^* 应当采用最大值点进行投毒, 对于 a^+ 应当采用最小值点进行投毒, 伪代码如算法 3 所示.

3.3.3 特殊情况

若被攻击的手臂 a 的训练数据为空, 则在求解优化问题构造投毒数据的过程中 $\boldsymbol{A}_a$ 与 $\boldsymbol{b}_a$ 均为初始值, 有

$$\begin{cases} \boldsymbol{A}_{a,t}=\boldsymbol{I}_d, \\ \boldsymbol{b}_{a,t}=\boldsymbol{0}, \end{cases} \rightarrow \begin{cases} \boldsymbol{\mu}=\boldsymbol{A}_{a,t}^{-1}\boldsymbol{x}^*=\boldsymbol{x}^*, \\ \boldsymbol{\nu}=\boldsymbol{A}_{a,t}^{-1}\boldsymbol{b}_{a,t}=\boldsymbol{0}, \end{cases} \rightarrow \begin{cases} u=\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\mu}=1, \\ v=\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\nu}=0, \\ w=\boldsymbol{\nu}^{\mathrm{T}}\boldsymbol{\nu}=0. \end{cases} \tag{22}$$

算法 3 OCA

输入: LinUCB 算法参数 $\hat{\boldsymbol{\theta}}_a$, $\boldsymbol{A}_a$, $\boldsymbol{b}_a$, 手臂数量 K, 训练数据 $\boldsymbol{X}_a$, $\boldsymbol{R}_a$, $a \in \mathcal{A}$, 目标上下文 $\boldsymbol{x}^*$, 目标手臂 a^*;

for $a \Leftarrow 1, 2, 3, \ldots, K$ **do**
  基于原参数 $\hat{\boldsymbol{\theta}}_a$, $\boldsymbol{A}_a$, $\boldsymbol{b}_a$ 模拟 LinUCB 算法使用 $\boldsymbol{X}_a$, $\boldsymbol{R}_a$ 进行参数更新, 记录更新后的参数 $\hat{\boldsymbol{\theta}}'_a$, $\boldsymbol{A}'_a$, $\boldsymbol{b}'_a$;
end for
while True **do**
  基于 $\hat{\boldsymbol{\theta}}'_{a^*}$, $\boldsymbol{A}'_{a^*}$, $\boldsymbol{b}'_{a^*}$ 计算 UCB_{a^*};
  for $a \Leftarrow 1, 2, 3, \ldots, K$ **do**
    基于 $\hat{\boldsymbol{\theta}}'_a$, $\boldsymbol{A}'_a$, $\boldsymbol{b}'_a$ 计算 UCB_a;
    基于 $\boldsymbol{A}'_a$, $\boldsymbol{b}'_a$ 求解方程 (20) 得到解集 S;
    计算 $g(\boldsymbol{x}_i)$, 其中 $\boldsymbol{x}_i$ 为 $s_i \in S$ 对应上下文, $i \in \{1, 2, \ldots, |S|\}$;
    if $a \neq a^*$ and $\mathrm{UCB}_a > \mathrm{UCB}_{a^*}$ **then**
      $\boldsymbol{x}' \Leftarrow \arg\min_{\boldsymbol{x}_i, i \in \{1,2,\ldots,|S|\}} g(\boldsymbol{x}_i)$;
      将 $(\boldsymbol{x}', 1)$ 添加到受害者模型手臂 a 的训练数据中;
      攻击者模拟 LinUCB 算法使用 $(\boldsymbol{x}', 1)$ 更新手臂 a 的参数, 并更新 $\hat{\boldsymbol{\theta}}'_a$, $\boldsymbol{A}'_a$, $\boldsymbol{b}'_a$;
    end if
  end for
  if $a^* = \arg\max_a \mathrm{UCB}_a$ **then**
    break;
  else
    $\boldsymbol{x}' \Leftarrow \arg\max_{\boldsymbol{x}_i, i \in \{1,2,\ldots,|S|\}} g(\boldsymbol{x}_i)$;
    将 $(\boldsymbol{x}', 1)$ 添加到受害者模型手臂 a^* 的训练数据中;
    攻击者模拟 LinUCB 算法使用 $(\boldsymbol{x}', 1)$ 更新手臂 a^* 的参数, 并更新 $\hat{\boldsymbol{\theta}}'_{a^*}$, $\boldsymbol{A}'_{a^*}$, $\boldsymbol{b}'_{a^*}$;
  end if
end while

将式 (22) 代入方程 (20) 将得到 $s^2\left(s^2-1\right)=0$, 其根为 $s=\pm 1, 0$. 当 $s=\pm 1$ 时, 所求 $\boldsymbol{x}$ 恰好为 $\pm\boldsymbol{x}^*$, 此时 OCA 退化为 TCA; 当 $s=0$ 时, 将其代入式 (19) 时会出现分母为 0 的不合法情况.

4 实验与结果分析

本节通过对合成数据集与真实数据集 (Jester 与 MovieLens25M) 分别进行攻击研究了 TCA 与 OCA 攻击方案的可行性, 并在合成数据集实验上验证了 OCA 方案设计过程中所做的假设.

4.1 合成数据集实验

4.1.1 数据合成规则

假设 LinUCB 算法运行在这样一个 CMAB 模型上: 有 $K=5$ 条手臂, 每条手臂 a 拥有一个 d 维向量参数 $\boldsymbol{\theta}_a \in \mathbb{R}^d$, 其中 $d=10$, 并且 $\boldsymbol{\theta}_a$ 均匀分布在 d 维空间 $\mathcal{S}^d$ 中. 其抽样方式为: 首先使用 d 维空间中的标准正态分布来生成 $\tilde{\boldsymbol{\theta}}_a$, 即 $\tilde{\boldsymbol{\theta}}_a \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$, 然后再对其进行归一化得到 $\boldsymbol{\theta}_a = \tilde{\boldsymbol{\theta}}_a / \|\tilde{\boldsymbol{\theta}}_a\|_2$.

训练数据的构造规则如下: 首先生成 $n=1000$ 条历史上下文向量 $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, 并保证其均匀分布在 $\mathcal{S}^d$ 空间中. 对于每一条上下文 $\boldsymbol{x}$, 都根据式 (1) 计算对应的 K 个奖励值 $\{r_a : a \in \mathcal{A}\}$, 其中令噪声 η 的参数 σ 为 0.1. 然后根据奖励值集合构造一个多项式分布 $\mathrm{multi}(p_1, p_2, \ldots, p_K)$, 其中 $p_a = \frac{\exp r_a}{\sum_{a \in \mathcal{A}} \exp r_a}$, 从该分布中随机选取一条手臂 a_i 作为 LinUCB 算法在观测到该上下文 $\boldsymbol{x}$ 时拉动的手臂, 重复操作 n 次, 得到每条手臂 a 的训练数据, 即上下文矩阵 $\boldsymbol{X}_a$ 和奖励向量 $\boldsymbol{R}_a$.

表 1 攻击合成数据集结果

Table 1 Results of attacks on synthetic data

Algorithm	Success rate (%)	Average cost ratio (%)	Success rate (%) (only attack target arm)
TCA	100	18.9	78
OCA	100	17.1	100
RA	100	24.0	–

4.1.2 实验设置

具体的实验流程如下, 其中从 LinUCB 算法初始化到攻击者实现攻击目标视为完成一轮攻击:

(1) 根据 4.1.1 小节的规则对 LinUCB 算法的参数进行初始化, 并生成原始训练数据.

(2) 攻击者从 $\mathcal{S}^d$ 空间中随机生成一个上下文向量作为目标上下文 $\boldsymbol{x}^*$.

(3) 在未被攻击的情况下, LinUCB 算法在观测到 $\boldsymbol{x}^*$ 时会根据式 (4) 选择 UCB 值最大的一条手臂, 而为了使攻击不失一般性, 本实验选择使用最小的 UCB 值对应的手臂作为目标手臂, 即

$$a^* = \arg\min_{a\in\mathcal{A}} \left\{ \boldsymbol{x}_t^{\mathrm{T}} \hat{\boldsymbol{\theta}}_{a_t,t} + \alpha \|\boldsymbol{x}_t\|_{\boldsymbol{A}_{a,t}^{-1}} \right\}, \tag{23}$$

其中 α 中的置信参数 δ 为 0.05.

(4) 攻击者对该 LinUCB 模型进行离线数据投毒攻击直至实现攻击目标.

4.1.3 攻击合成数据集

为了研究 TCA 与 OCA 方案的可行性, 本小节使用 TCA, OCA 分别对使用上述合成数据训练的 LinUCB 算法进行 100 轮攻击, 并与 RA 方案进行对比, 每轮攻击进行如下观察:

(1) 观察攻击是否成功;

(2) 观察攻击成功所需的攻击成本.

从表 1 的前 3 列数据可以看出 3 种投毒攻击方案均以 100% 的成功率实现攻击目标, 其中 OCA 所需的平均攻击成本最低. 为了更直观地表示这 3 种攻击方案的攻击效果, 图 6 展示了 3 种攻击方案各轮的平均攻击成本, 从图中可以看出 RA 的攻击成本波动相当大, 而 OCA 攻击与 TCA 攻击每轮都可以凭借较低的攻击成本实现攻击目标且攻击成本相对更加稳定.

值得注意的是, 本实验的结果与文献 [7] 中的部分实验结论存在矛盾, 即 3.2 小节中提到的结论 (2). 根据结论 (2), 由于 TCA 方案每次都使用能最大化 $\langle \boldsymbol{X}_a^{(i)}, \boldsymbol{x}^* \rangle$ 的上下文进行攻击, 攻击效果理应最强, 但实验结果却是 OCA 方案平均所需要的攻击成本更低. 为了进一步对比 TCA 方案与 OCA 方案设计的投毒上下文对手臂的 UCB 值的影响程度, 额外进行以下观测: 在仅对目标手臂的训练数据进行投毒攻击的情况下实现攻击目标的成功率.

观测结果如表 1 的第 4 列数据所示, 同时图 6 中 TCA 对应的折线上的红色圆点也代表了在该轮中 TCA 无法通过只攻击 a^* 来实现攻击目标. 可以看到如果限制为只能对 a^* 的训练数据进行投毒攻击, TCA 并不能保证 100% 的攻击成功率.

图 7 则更直观地展示了 TCA 攻击失败 (仅攻击目标手臂) 的某一轮中 a^* 对 $\boldsymbol{x}^*$ 的 UCB 值变化曲线. 可以看出 OCA 攻击时每投毒一次 UCB 值的增量都大于 TCA, 且始终呈增长趋势, 直至超过原本所有手臂中的最大 UCB 值, 进而实现攻击目标. 而被 TCA 攻击的目标手臂, 随着投毒条目数

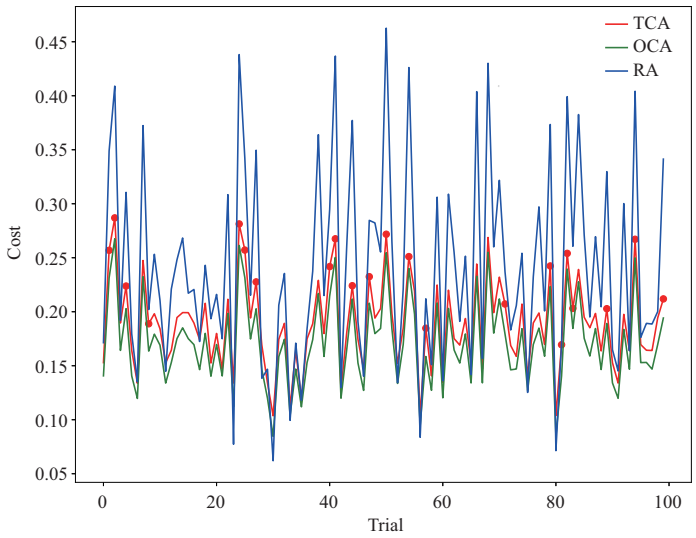

图 6 (网络版彩图) TCA, OCA, RA 的平均攻击成本

Figure 6 (Color online) Average cost ratio of TCA, OCA and RA

的提高, 其 UCB 值呈现出先提高后缓慢降低的趋势, 即使用 $\boldsymbol{x}^*$ 作为投毒数据并不能持续提高 a^* 的 UCB 值.

4.1.4 $\boldsymbol{A}_{a,t}^{-1}$ 的最大特征值

第 3.3.2 小节中为了方便推导公式进而得到可行解, 对 $g(\boldsymbol{x})$ 中含分式的部分做了近似操作并给出了理论证明. 本小节旨在通过实验来观察 $\boldsymbol{A}_{a,t}^{-1}$ 的最大特征值随着手臂 a 更新次数 (真实训练数据的条数) 增加时的变化曲线, 以验证近似操作的合理性.

采用与 4.1.3 小节相同的攻击设置对合成数据集进行攻击, 实验结果如图 8 所示. 可以看到, 在手臂更新次数达到约 150 次左右时, $\boldsymbol{A}_{a,t}^{-1}$ 的最大特征值已经下降到了 0.1; 在约 250 次更新之后, 进一步下降至 0.05, 且仍呈下降趋势. 而在 CMAB 模型的实际应用中, 绝大部分手臂的更新次数都是大于上述数量级的, 因此在攻击时认为 $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}_{a,t}^{-1}\boldsymbol{x} \ll 1$ 是合理的.

4.2 真实数据集实验

4.2.1 数据集介绍与预处理

为了测试 TCA 与 OCA 在真实场景中的应用效果, 本节使用两个经典开源数据集 MovieLens-25M [21] 数据集与 Jester [22] 数据集来模拟两种方案对真实 CMAB 模型的攻击, 下面对这两个数据集进行简要介绍.

(1) MovieLens25M 数据集. 该数据集是由 GroupLens 研究组收集和提供的最大的数据集之一, 内容主要包含了大量用户对电影的评分数据. 25M 指的是该数据集包含了从 1995 年到 2019 年期间收集的超过 2500 万条评分数据, 由 162541 个用户对 62423 部电影进行评分, 其中用户对电影的评分采用 0~5 分制. 此外, 数据集还包含了许多电影的元数据信息, 如电影的标题、类型、导演、演员等.

(2) Jester 数据集. 该数据集是由马里兰大学帕克分校 (University of Maryland, College Park) 的研究人员创建和提供的一个开放数据集, 内容主要包括了一系列来自 Jester 网站的笑话评分数据与笑话的文本描述. 该数据集包含了超过 400 万条笑话评分数据, 由 73421 个用户对 100 个笑话进行了评

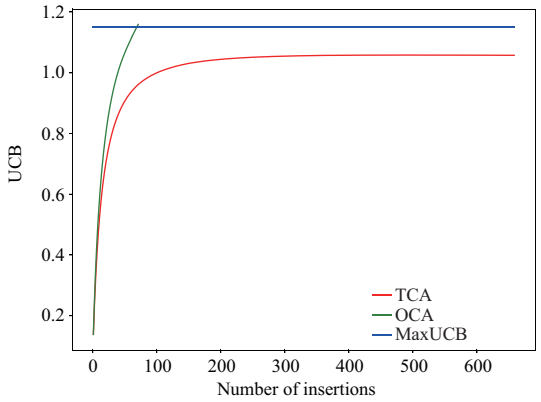

图 7 (网络版彩图) 攻击过程中目标手臂的 UCB 变化曲线

Figure 7 (Color online) UCB of the target arm during attacks

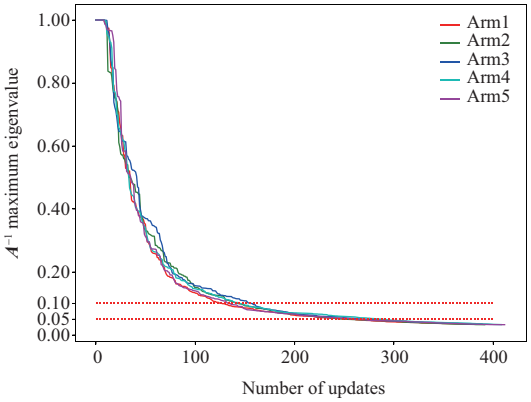

图 8 (网络版彩图) $\boldsymbol{A}_{a,t}^{-1}$ 的最大特征值与手臂更新次数的关系

Figure 8 (Color online) Maximum eigenvalue of $\boldsymbol{A}_{a,t}^{-1}$ with the arm updates

分, 其中笑话的评分规则是正负 10 分制, 10 表示最喜欢, 而 -10 则表示最不喜欢. Jester 数据集的优点在于它 "题材" 的独特性, 与传统的电影或商品评分数据不同, 它是一个用于推荐系统和情感分析等领域的有趣且经典的研究资源.

对于 MovieLens25M 数据集, 只取其中拥有不少于 1000 条评分数据的电影, 筛选后可以得到一个拥有 162539 个用户对 3794 个电影的评分数据的数据集, 对评分矩阵进行低秩矩阵分解 [23, 24], 从评分矩阵中提取出维度为 $d = 100$ 的用户特征向量与电影特征向量, 并在计算奖励时向其中添加服从 $\mathcal{N}(0, 0.01)$ 的高斯噪声.

对于 Jester 数据集, 选取了其中评价人数最多的 40 个笑话, 并筛选出对这 40 个笑话均做出过评分的 37276 个用户, 即处理后的评分矩阵是一个完备矩阵, 依然通过低秩矩阵分解的方式提取用户特征向量与笑话特征向量, 维度为 $d = 35$, 然后进行同样的噪声添加.

4.2.2 攻击真实数据集

假设分别存在一个电影推荐网站 MovieRec 与一个笑话推荐网站 JokeRec, 其服务器均使用 LinUCB 算法给访问网站的用户进行电影与笑话推荐, 二者的日访问量均为 $n = 1000$, 访问网站的用户的特征向量分别来自对应数据集. 两种网站均已正常运营了 30 天, 攻击者决定在第 31 天开始进行攻击, 每天攻击一次, 持续攻击 30 天. 其中第 m 天进行攻击时选择第 $(m - 30)$ 天中第一个访问网站的用户的特征向量作为目标上下文 $\boldsymbol{x}^*$, 选择第 $(m - 30)$ 天中最后一个用户访问网站时网站推荐的电影或笑话的特征向量作为目标手臂 a^*, 分别应用 TCA 与 OCA 对两个网站展开攻击.

对 MovieLens25M 数据集应用两种攻击方案所得结果如图 9 所示, 可以看出 TCA 的攻击成本在真实数据集上的表现存在较大波动, 且整体的平均攻击成本较高, 远高于 OCA 攻击, 同时也导致后者的变化幅度在图中表现不明显. 尽管如此, 该实验也验证了两种方案应用在真实场景时均具有一定可行性, 且 OCA 攻击具有更加优秀的表现.

为了更清晰地观察 OCA 攻击在两种不同类型的数据集上的表现, 图 10 展示了对 MovieLens25M 数据集与 Jester 数据集应用 OCA 的结果. 可以看出实际上 OCA 攻击的平均攻击成本波动也较大, 但整体所需的平均成本较低, 仅为 2% 左右, 验证了 OCA 在攻击真实应用时的高效性.

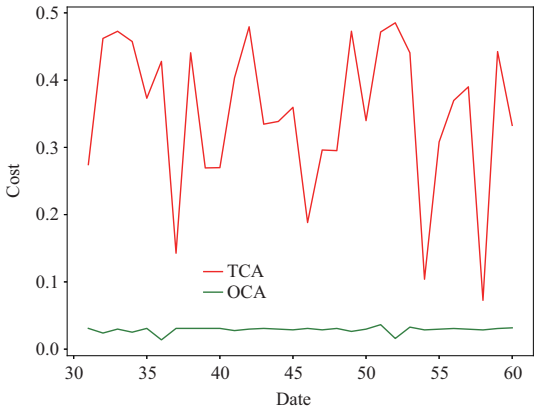

图 9 (网络版彩图) 使用 TCA 与 OCA 攻击 MovieRec

Figure 9 (Color online) Use TCA and OCA to attack MovieRec

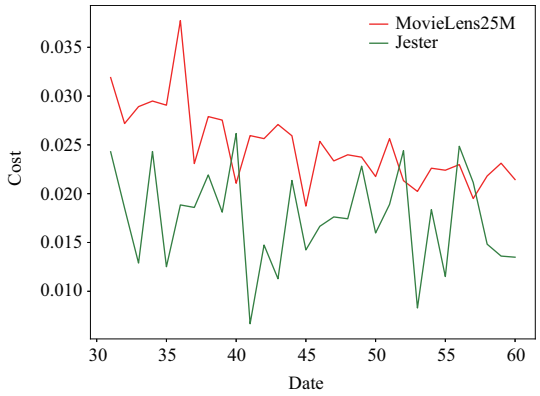

图 10 (网络版彩图) 使用 OCA 攻击 MovieRec 与 JokeRec

Figure 10 (Color online) Use OCA to attack MovieRec and JokeRec

此外, OCA 攻击 Jester 数据集时的平均攻击成本与 MovieLens25M 数据集相比偏低, 主要原因在于后者对应的 CMAB 模型中的手臂数量 $K_M = 3794$ 是远大于前者的 $K_J = 40$ 的, 即当需要降低 a^+ 的 UCB 值时, 后者需要向 a^+ 的训练集中添加更多的虚假数据, 造成了攻击成本的增加.

5 结论

本文提出了两种通过向训练数据中添加虚假数据的方式来对 LinUCB 算法进行离线数据投毒攻击的方案, 分别为 TCA 攻击方案与 OCA 攻击方案, 其中设计方案的主要贡献在于如何构造投毒数据中的上下文部分. 据我们所知, 这是首次通过向训练数据中添加虚假数据的方式来对 LinUCB 算法进行离线数据投毒攻击. TCA 方案借鉴了奖励投毒攻击方案的思路, 旨在通过向上下文多臂老虎机的训练集中添加形如 $(\pm\boldsymbol{x}^*, 1)$ 的虚假数据来改变手臂的 UCB 值, 从而让目标手臂符合 LinUCB 算法的手臂选择规则, 以实现攻击目标. OCA 方案在 TCA 的基础上进行了优化, 将构造投毒上下文的问题建模为优化问题, 并通过合理近似来对目标函数进行化简, 最终简化为求解一个必有实数根的一元四次方程的问题, 极大地降低了求解难度, 且攻击效果更佳.

一方面, 在设计 OCA 方案的过程中, 对公式推导的关键步骤给出了详细的理论证明, 包括化简矩阵和求逆公式与合理近似操作; 另一方面, 通过实验验证了方案设计过程的正确性与严谨性, 同时也给两种方案的可行性与 OCA 的高效性提供了实验验证. 综上所述, 本文提出的 TCA 方案原理简单且易实现, OCA 方案运算简洁且更高效. 未来的研究目标是希望进一步控制方案的攻击成本波动使之更加平稳并研究能保证攻击成功的成本边界.

参考文献

1 Lattimore T, Szepesvári C. Bandit Algorithms. Cambridge: Cambridge University Press, 2020. 8–9

2 Zhang M, Nguyen-Tang T, Wu F, et al. Two-stage neural contextual bandits for personalised news recommendation. 2022. ArXiv:2206.14648

3 Avadhanula V, Baldeschi R C, Leonardi S, et al. Stochastic bandits for multi-platform budget optimization in online advertising. In: Proceedings of the Web Conference 2021, 2021. 2805–2817

4 Lu M, Chen Y, Lee S I. A deep Bayesian bandits approach for anticancer therapy: exploration via functional prior. 2022. ArXiv:2205.02944
5 Li L, Chu W, Langford J, et al. A contextual-bandit approach to personalized news article recommendation. In: Proceedings of the 19th International Conference on World Wide Web, 2010. 661–670
6 Slivkins A. Introduction to multi-armed bandits. FNT Machine Learn, 2019, 12: 1–286
7 Ma Y, Jun K S, Li L, et al. Data poisoning attacks in contextual bandits. In: Proceedings of the 9th International Conference on Decision and Game Theory for Security, 2018. 186–204
8 Ma Y, Zhou Z. Adversarial attacks on adversarial bandits. In: Proceedings of the 11th International Conference on Learning Representations, 2023. 1–12
9 Garcelon E, Roziere B, Meunier L, et al. Adversarial attacks on linear contextual bandits. In: Proceedings of the Advances in Neural Information Processing Systems 33, 2020. 14362–14373
10 Liu G, Lai L. Efficient action poisoning attacks on linear contextual bandits. 2021. ArXiv:2112.05367
11 Thompson W R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika, 1933, 25: 285–294
12 Agrawal S, Goyal N. Thompson sampling for contextual bandits with linear payoffs. In: Proceedings of the 30th International Conference on Machine Learning, 2013. 127–135
13 Abeille M, Lazaric A. Linear Thompson sampling revisited. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, 2017. 176–184
14 Walsh T J, Szita I, Diuk C, et al. Exploring compact reinforcement-learning representations with linear regression. 2012. ArXiv:1205.2606
15 Abbasi-Yadkori Y, Pál D, Szepesvári C. Improved algorithms for linear stochastic bandits. In: Proceedings of the Advances in Neural Information Processing Systems 24, 2011. 2312–2320
16 Liu F, Shroff N. Data poisoning attacks on stochastic bandits. In: Proceedings of the 36th International Conference on Machine Learning (ICML), 2019. 4042–4050
17 Miller K S. On the inverse of the sum of matrices. Math Mag, 1981, 54: 67–72
18 Horn R A, Johnson C R. Matrix Analysis. 2nd ed. New York: Cambridge University Press, 2013. 234–235
19 Spiegel M R. Mathematical Handbook of Formulas and Tables. 3rd ed. New York: McGraw-Hill, 2009. 13–14
20 Shmakov S L. A universal method of solving quartic equations. Int J Pure Appl Math, 2011, 71: 251–259
21 Harper F M, Konstan J A. The MovieLens datasets: history and context. ACM Trans Interact Intell Syst, 2016, 5: 1–19
22 Goldberg K, Roeder T, Gupta D, et al. Eigentaste: a constant time collaborative filtering algorithm. Inf Retrieval, 2001, 4: 133–151
23 Ouyang T. Feature learning for stacked ELM via low-rank matrix factorization. Neurocomputing, 2021, 448: 82–93
24 Su Y, Xu J, Hong D, et al. Deep low-rank matrix factorization with latent correlation estimation for micro-video multi-label classification. Inf Sci, 2021, 575: 587–598

Data poisoning attacks on the LinUCB algorithm

Weilong JIANG & Kun HE*

School of Computer Science & Technology, Huazhong University of Science and Technology, Wuhan 430074*, China*
* Corresponding author. E-mail: brooklet60@hust.edu.cn

Abstract The LinUCB algorithm is a typical algorithm for solving the contextual multi-armed bandit problem, which is widely used in scenarios such as news delivery, product recommendation, and medical resource allocation. There is very little research on the security of this algorithm, which requires further investigation of their attack methods in order to make targeted and even universal defense measures. In this work, we first propose two attack schemes for offline data poisoning attacks on the LinUCB algorithm by adding fake data, namely TCA (target context attack) and OCA (optimized context attack). The former generates poisoning data based on the similarity between training data and target context, while the latter models an optimization problem to construct the poisoning data, which is an optimized version of the former. Experimental evaluations show that only by adding a small amount of poisoning data we could achieve a 100% attack success rate.

Keywords contextual multi-armed bandit, LinUCB, data poisoning attack, white-box attack, optimization problem