Efficient Adversarial Attacks on High-dimensional Offline Bandits

Seyed Mohammad Hadi Hosseini
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
hadi.hosseini17@sharif.edu

Amir Najafi
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
amir.najafi@sharif.edu

Mahdieh Soleymani Baghshah
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
soleymani@sharif.edu

Abstract

Bandit algorithms have recently become a popular tool for evaluating machine learning models, including generative image models and large language models, by efficiently identifying top-performing candidates without exhaustive comparisons. These methods typically rely on a reward model—often distributed with public weights on platforms such as Hugging Face—to provide feedback to the bandit. While online evaluation is expensive and requires repeated trials, offline evaluation with logged data has emerged as an attractive alternative. However, the adversarial robustness of such offline bandit evaluation remains largely unexplored. We investigate the vulnerability of offline bandit training to adversarial perturbations of the reward model. We introduce a novel threat model in which an attacker manipulates the reward function—using only offline data in high-dimensional settings—to hijack the bandit's behavior. Starting with linear reward functions and extending to nonlinear models such as ReLU neural networks, we show that even small, imperceptible perturbations to the reward model's weights can drastically alter the algorithm's behavior. Our analysis reveals a striking high-dimensional effect: as input dimensionality grows, the required perturbation norm decreases, making modern applications (e.g., image evaluation) especially vulnerable. Extensive experiments confirm that naive random perturbations are ineffective, but carefully targeted ones succeed with near-perfect attack success rates. To address computational challenges, we propose efficient heuristics that achieve almost 100% success while dramatically reducing attack cost. Finally, we validate our approach on the UCB bandit and provide theoretical evidence that adversaries can delay optimal arm selection proportionally to the input dimension.

1 Introduction

The multi-armed bandit (MAB) framework is a cornerstone of online learning, balancing exploration and exploitation to efficiently identify the optimal choice (or "arm") within a limited time horizon. MAB algorithms have been widely applied in domains such as online recommendation systems [Kveton et al., 2015, Chapelle et al., 2014, Li et al., 2010], adaptive clinical trials [Kuleshov and Precup, 2014], and dynamic pricing [Tullii et al., 2024]. More recently, they have been adopted for evaluating generative models—from diffusion models [Ho et al., 2020, Rombach et al., 2022,

2nd Workshop on Aligning Reinforcement Learning Experimentalists and Theorists (ARLET 2025).

Podell et al., 2023] and GANs [Goodfellow et al., 2020, Karras et al., 2019, Arjovsky et al., 2017] to large language models (LLMs) such as ChatGPT [Achiam et al., 2023], Gemini [Team et al., 2023], and DeepSeek [Guo et al., 2025, Liu et al., 2024]. In this setting, bandit algorithms (e.g., Upper Confidence Bound (UCB) [Auer, 2002]) offer a sample-efficient alternative to naive pairwise comparisons by rapidly identifying the best-performing model.

While online evaluation remains effective, it is often costly and impractical for rapidly evolving generative models. As a result, practitioners increasingly turn to offline evaluation, where a fixed logged dataset is reused to compare algorithms (e.g., UCB [Auer, 2002], ETC [Garivier et al., 2016], ε-greedy [Dann et al., 2022]) or tune hyperparameters without repeated data collection. Despite its practicality, this setting introduces two overlooked risks. First, it is vulnerable to adversarial manipulation that could bias the selection process. Second, it is prone to overfitting to the reward model when multiple degrees of freedom are available. For instance, recent work has shown a tendency to test numerous performance metrics or continuously adjust metric hyperparameters on the same logged dataset when training MABs [Rezaei et al., 2025, Hu et al., 2024, Saito and Nomura, 2024].

We aim to provide partial answers to the above concerns across a range of settings. Specifically, we study adversaries that seek to manipulate only the reward model in order to hijack the behavior of a bandit algorithm. We show that even a weak adversary can succeed in this task, provided the data is high-dimensional. In our threat model, the adversary has access to the offline (logged) dataset but cannot alter the samples. Moreover, the adversary cannot interfere with the learner's training procedure; its only leverage is to perturb the reward model before training begins.

We analyze two reward-model scenarios. First, we study linear reward functions in high-dimensional spaces. This setting, though simplified, offers clarity and valuable insights into the mechanics of our attack. Our results show that even small perturbations to the weights of a linear model can dramatically alter the bandit's behavior. Second, we extend our analysis to nonlinear reward models such as deep neural networks [LeCun et al., 2015], which are widely used in practice. For instance, modern image-based generative models are often evaluated with neural reward functions such as CLIP [Radford et al., 2021], BLIP [Li et al., 2022], and VQA models [Hu et al., 2023, Singh and Zheng, 2023, Cho et al., 2024]. These models are typically open-source, with publicly available weights on platforms such as Hugging Face—making them convenient, but also vulnerable. We demonstrate that exposing verifier weights can be exploited: by constructing a linear approximation of the reward model, an adversary can craft perturbations of imperceptibly small norm that nonetheless yield highly effective attacks. Moreover, we show that this linear approximation becomes increasingly accurate as the network's hidden layer width grows, consistent with predictions from Neural Tangent Kernel (NTK) theory [Jacot et al., 2018, Golikov et al., 2022].

1.1 Our Contribution

We identify several key insights and introduce novel algorithms that enable highly effective attacks on offline bandit training. Furthermore, we provide theoretical guarantees that support the effectiveness of the proposed methods:

Targeted vs. random perturbations: Effective attacks require solving a targeted optimization problem; naive random perturbations fail to compromise the reward model. Moreover, as input dimensionality increases, the norm of the required perturbation decreases. This inverse relationship implies that high-dimensional data, such as images, are especially vulnerable, since their reward models inherently operate in high-dimensional spaces.

Computational efficiency: Solving the full optimization problem to hijack an entire trajectory is computationally expensive. To address this, we propose heuristic methods that dramatically reduce the computational burden while preserving a near-perfect attack success rate. To validate our findings, we evaluate the proposed attack across multiple bandit algorithms, with particular emphasis on UCB, one of the most widely used approaches. We also apply our attack to other bandit algorithms, such as ETC and ϵ-greedy, to demonstrate that our method is not limited to UCB.

Theorethical Support: Complementing our empirical results, we also provide theoretical support. Assume a multi-armed bandit with K arms, operating over a d-dimensional input space, and a horizon of T arm pulls. We show that such attacks are not only feasible whenever $d \geq KT$ (see Theorem 3.3), but also become easier to construct in high-dimensional regimes (see Theorem 3.4). In particular, we

prove that the ℓ_2-norm of the attack decreases as $\widetilde{\mathcal{O}}(d^{-1/2})$ as the dimension d increases. Our proofs draw upon tools from high-dimensional concentration inequalities and random matrix theory.

2 Problem Definition and Notation

For $K \in \mathbb{N}$, let $[K] = \{1, 2, \ldots, K\}$. We consider a stochastic multi-armed bandit problem with K arms, where each arm $i \in [K]$ is associated with a data-generating distribution P_i over $\mathbb{R}^d$. Here, d denotes the dimension of the input space. At each round, when the bandit pulls arm i, it receives a sample $\boldsymbol{X} \sim P_i$, drawn independently from past rounds. The *reward* is then computed as $r(\boldsymbol{X})$, where $r : \mathbb{R}^d \to \mathbb{R}$ is a known reward function that may be perturbed adversarially before training begins. We study two instantiations of the reward function:

1. **Linear model:** $r(\boldsymbol{X}) = \boldsymbol{w}^\top \boldsymbol{X}$, with a fixed parameter vector $\boldsymbol{w} \in \mathbb{R}^d$.
2. **Neural network model:** $r(\boldsymbol{X}) = \mathrm{NN}_{\boldsymbol{\theta}}(\boldsymbol{X})$, where $\mathrm{NN}_{\boldsymbol{\theta}}$ is a feedforward network with parameters $\boldsymbol{\theta} \in \mathbb{R}^D$ and some fixed activation function $\sigma(\cdot) : \mathbb{R} \to \mathbb{R}$. The parameter dimension is $D = d \cdot W_1 + W_1 \cdot W_2 + \cdots + W_{L-1} \cdot W_L + W_L$, for a network with L hidden layers of widths $W_1, W_2, \ldots, W_L$, respectively.

Throughout, we focus on the high-dimensional regime: either $d \gg 1$ (linear case) or $\max_i W_i \gg 1$ (neural network case). We assume arm $i^* \in [K]$ is optimal, i.e., $\mathbb{E}_{P_{i^*}}[r(\boldsymbol{X})] > \mathbb{E}_{P_i}[r(\boldsymbol{X})]$ for $i \neq i^*$, where $\mathbb{E}[\cdot]$ is expectation operator. Hence, we expect the bandit algorithm to (with high probability) learn to pull the optimal arm i^* after some number $T \geq 1$ of exploratory trials across the K arms. The value T is referred to as the *horizon*. Our goal is to design an adversarial attack on the reward function $r(\cdot)$ such that, with high probability, the bandit fails to identify and exploit the optimal arm.

Suppose we are given K logged datasets consisting of independent samples from each arm distribution $P_1, \ldots, P_K$. Specifically, let $\mathcal{D}_i = \{\boldsymbol{X}_j^{(i)} \mid j = 1, \ldots, n_i\}$, be a dataset of size $n_i \geq 2$ containing i.i.d. samples from P_i. When the bandit pulls arm i for the t_i-th time, it receives the sample $\boldsymbol{X}_{t_i}^{(i)}$, for $t_i \in [n_i]$. We assume that each dataset $\mathcal{D}_1, \ldots, \mathcal{D}_K$ contains sufficiently many samples so that, throughout the T rounds, the bandit always observes a fresh sample upon each arm pull.

Our Problem: Assume an adversary $\mathscr{A}$ with access to the offline logged datasets $\mathcal{D}_1, \ldots, \mathcal{D}_K$, and the ability to perturb the reward model $r(\cdot)$. In the linear case, the adversary modifies the parameter $\boldsymbol{w}$ to $\boldsymbol{w} + \boldsymbol{\delta}$ for some perturbation vector $\boldsymbol{\delta} \in \mathbb{R}^d$. In the neural network case, the adversary modifies the parameter vector $\boldsymbol{\theta}$ to $\boldsymbol{\theta} + \boldsymbol{\delta}$, where $\boldsymbol{\delta} \in \mathbb{R}^D$. In both settings, the perturbation norm is chosen as small as possible while still guaranteeing a prescribed level of damage to the bandit's trajectory, for example, ensuring that the number of times the optimal arm i^* is pulled falls below some threshold $T' < T$. The ultimate objective of $\mathscr{A}$ is to manipulate training such that, over T steps on the offline datasets, the bandit selects a suboptimal arm $i \neq i^*$ more frequently than the optimal arm. In stronger attack scenarios, $\mathscr{A}$ may even enforce a pre-specified target trajectory for the multi-armed bandit.

3 Method

Assume we use the UCB [Auer, 2002] algorithm for the multi-armed bandit problem . UCB begins by pulling each arm once, and then proceeds by selecting arms in a way that balances exploitation and exploration: it favors arms with higher observed average rewards, while a carefully designed exploration term encourages underexplored arms to be pulled. It is theoretically known that UCB achieves a regret bound of $\mathcal{O}(\log T)$, which is optimal up to constants, making it widely used in practice. In this work, we focus on this particular algorithm. Let $A_t = A_t(\boldsymbol{\delta})$ for $t \in [T]$ denote the arm chosen at step t when the adversary $\mathscr{A}$ selects the perturbation vector $\boldsymbol{\delta}$. Then, the algorithm proceeds for $t = 1, \ldots, T$ as follows:

- For $t = 1, \ldots, K$, pull each arm once. That is, $A_t = t$.
- For $t \geq K$, select the next arm according to the following deterministic maximization:

$$A_{t+1}(\boldsymbol{\delta}) \triangleq \underset{i \in [K]}{\operatorname{argmax}} \left\{ \Lambda_t(i; \boldsymbol{\delta}) \triangleq \frac{1}{N_i(t)} \sum_{j=1}^{N_i(t)} r\big(\boldsymbol{X}_j^{(i)}; \boldsymbol{\delta}\big) + \sqrt{\frac{2 \log t}{N_i(t)}} \right\}, \tag{1}$$

where $N_i(t)$ denotes the number of times arm i has been pulled up to step t, and $r(\boldsymbol{X}; \boldsymbol{\delta})$ denotes the adversarially perturbed reward value of input $\boldsymbol{X}$ given the perturbation vector $\boldsymbol{\delta}$, e.g., $r(\boldsymbol{X}; \boldsymbol{\delta}) = (\boldsymbol{w} + \boldsymbol{\delta})^\top \boldsymbol{X}$ in the linear case. Also, $\Lambda_t(i; \boldsymbol{\delta})$ is called the UCB score of arm i at time step t.

The adversary aims to keep $\|\boldsymbol{\delta}\|_2$ as small as possible. However, depending on its objective, $\mathscr{A}$ may design $\boldsymbol{\delta}$ to achieve one of the following goals:

Full Trajectory Attack. In this scenario, $\mathscr{A}$ attempts to force the bandit to follow a completely predetermined, arbitrary, and fixed target trajectory $\widetilde{A}_t \in [K]$ for all $t \in [T]$. This requires solving the following constrained optimization problem:

$$\boldsymbol{\delta}^* \triangleq \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \ \|\boldsymbol{\delta}\|_2^2 \quad \text{subject to} \quad \Lambda_t(\widetilde{A}_t; \boldsymbol{\delta}) > \Lambda_t(i; \boldsymbol{\delta}), \quad \forall \big(K < t \leq T,\ i \neq \widetilde{A}_t\big). \tag{2}$$

The optimization in (2) requires $(T - K)(K - 1)$ constraints. In the linear reward model, these constraints are linear in $\boldsymbol{\delta}$.

Trajectory-Free Attack. A less restrictive attack requires the bandit to *avoid* pulling the optimal arm at a given set of time-steps $\mathcal{T} \subset [T]$. Note that $\mathcal{T}$ may include all steps from $K + 1$ to T. Unlike the full trajectory attack, here $\mathscr{A}$ only ensures that i^* is not selected, without prescribing exactly which arm must be pulled. One simplified formalization is as follows: consider an arbitrary trajectory $\widetilde{A}_t \in [K]$ for all $t \in [T]$ such that $\widetilde{A}_t \neq i^*$. Then we can define:

$$\boldsymbol{\delta}^* \triangleq \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \ \|\boldsymbol{\delta}\|_2^2 \quad \text{subject to} \quad \Lambda_t(\widetilde{A}_t; \boldsymbol{\delta}) > \Lambda_t(i^*; \boldsymbol{\delta}), \quad \forall\, K < t \leq T. \tag{3}$$

Here, the auxiliary sequence $\widetilde{A}_t \neq i^*$ for $K < t \leq T$ is arbitrary and can be optimized (e.g., via heuristics) by the attacker to further reduce attack strength. In our experiments, we use a simple round-robin strategy for $\widetilde{A}_t$. The formulation in (3) requires only $T - K$ constraints, assuming $\mathcal{T}$ includes all steps from $K + 1$ to T.

Trajectory-free attacks thus reduce the number of constraints by a factor of $K - 1$, significantly lowering computational cost when K is large. However, for large T, they may still be expensive and difficult to design. To address this, we propose an online attack strategy.

Online Score-Aware (OSA) Attack. Consider a fixed set of time-steps $\mathcal{T} \subset [T]$. We iterate over $t \in \mathcal{T}$ in increasing order. Let $\boldsymbol{\delta}^*_{t-1}$ denote the optimal attack up to time step $t - 1$. At each step t, the attacker checks whether the UCB score $\Lambda_t(i^*; \boldsymbol{\delta}^*_{t-1})$ is the largest among all arms. If not, we simply move on to the next t. Otherwise, we add the constraint

$$\Lambda_t(\widetilde{i}_t; \boldsymbol{\delta}) > \Lambda_t(i^*; \boldsymbol{\delta}),$$

where $\widetilde{i}_t$ is the index of the runner-up arm with the second highest UCB score at time t. The solution $\boldsymbol{\delta}^*_t$ is then updated using the additional constraint, and the process continues until the end of $\mathcal{T}$. Therefore, instead of a single QP, a series of QPs with increasing number of constraints should be solved. However, the number of constraints in each QP is usually very small and the problem can be solved almost instantly.

While a detailed theoretical analysis of the OSA attack is beyond the scope of this paper, we observe that it substantially reduces computational cost in practice. The number of effective constraints generated by the online procedure is usually far smaller than T, resulting in significantly faster attack construction compared to the full trajectory and trajectory-free methods. The algorithms for all methods are provided in Appendix C.

Some interesting and fundamental questions arise regarding the attack designs discussed above:

- **Optimization formulation.** How can these attack designs be explicitly formulated in real-world applications? In the next section, we show that for the linear reward model $r(\boldsymbol{X}) = \boldsymbol{w}^\top \boldsymbol{X}$, all of the above formulations reduce to well-structured and convex Quadratic Programs (QPs), which can be solved efficiently. In this setting, it is evident that reducing the number of constraints significantly accelerates solving the program. Interestingly, a similar argument also holds for wide neural network models.
- **Feasibility.** The efficiency gains mentioned above are only meaningful if the optimization problems are feasible, i.e., if there exists at least one perturbation vector $\boldsymbol{\delta}$ that satisfies all

constraints. In Section 3.2, where we present our theoretical guarantees, we show that under mild assumptions on the reward model and the data-generating distributions $P_1, \ldots, P_K$, the set of constraints is feasible with probability 1, provided that the data dimension d, or in the case of neural networks the maximum width $\max_i W_i$, exceeds the number of constraints.

3.1 Optimization

The following result shows that, under a linear reward model, the proposed attack formulations admit an efficient convex optimization representation.

Theorem 3.1 (Linear Reward Model)**.** *Consider the three attack designs in Section 3 under the linear reward model $r(\boldsymbol{X}) = \boldsymbol{w}^\top \boldsymbol{X}$ for some fixed vector $\boldsymbol{w} \in \mathbb{R}^d$. Then there exist an indexed vector set $\{\boldsymbol{T}_{i,t}\}$ and a scalar set $\{R_{i,t}\}$, for $i \in [K-1]$ and $K < t \leq T$, such that all three attack designs can be expressed as the quadratic program*

$$\boldsymbol{\delta}^* \triangleq \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \ \|\boldsymbol{\delta}\|_2^2 \quad \text{subject to} \quad \boldsymbol{\delta}^\top \boldsymbol{T}_{i,t} > R_{i,t}, \quad \forall (i,t) \in \mathcal{I} \subset [K-1] \times [T]. \tag{4}$$

The values of $\boldsymbol{T}_{i,t}$ and $R_{i,t}$ are determined solely by the offline datasets $\mathcal{D}_1, \ldots, \mathcal{D}_K$ and the original reward model $\boldsymbol{w}$. The joint arm–timestep index set $\mathcal{I}$ depends only on the choice of attack type (full-trajectory, trajectory-free, or OSA) and is determined according to the specification of the method.

Proof is given in Appendix B. The computational cost is governed by the size of $\mathcal{I}$. In particular, for full-trajectory attacks we have $|\mathcal{I}| = \mathcal{O}(TK)$, for trajectory-free attacks $|\mathcal{I}| = \mathcal{O}(T)$, and for OSA attacks the effective size of $\mathcal{I}$ is typically much smaller in practice. Solving a QP of the form (4) with d variables and $|\mathcal{I}|$ constraints requires approximately $\widetilde{\mathcal{O}}\big(|\mathcal{I}|^3 + d|\mathcal{I}|^2\big)$ operations [Gay et al., 1998], highlighting the central role of the constraint set size in the overall complexity. In our experiments (see Figure 4), we observed that OSA can reduce $|\mathcal{I}|$ to as small as $\mathcal{O}(\log T)$.

When the reward model is a neural network $\mathrm{NN}_{\boldsymbol{\theta}}(\cdot)$, the UCB score $\Lambda_i(t; \boldsymbol{\delta})$ in (1) is nonlinear in $\boldsymbol{\delta}$, and the convex reduction in Theorem 3.1 no longer holds. However, results from neural tangent kernel (NTK) theory [Jacot et al., 2018] show that for sufficiently wide networks with randomly initialized weights, the network behaves approximately linearly in its parameters. Formally,

$$\mathrm{NN}_{\boldsymbol{\theta}+\boldsymbol{\delta}}(\boldsymbol{X}) = \mathrm{NN}_{\boldsymbol{\theta}}(\boldsymbol{X}) + \nabla_{\boldsymbol{\theta}} \mathrm{NN}_{\boldsymbol{\theta}}(\boldsymbol{X})^\top \boldsymbol{\delta} + \mathcal{O}\big(W_{\max}^{-1}\big) \quad \forall\, \boldsymbol{X} \in \mathbb{R}^d, \tag{5}$$

where $W_{\max}$ denotes the network width. Thus, for highly overparameterized networks the reward model can be well approximated by an affine function of $\boldsymbol{\delta}$, yielding a quadratic program analogous to (4), with affine constraints. Recall that $\nabla_{\boldsymbol{\theta}} \mathrm{NN}_{\boldsymbol{\theta}}(\boldsymbol{X})$ represents the embedding of the input $\boldsymbol{X}$ in the neural tangent space of the network $\mathrm{NN}_{\boldsymbol{\theta}}$, centered at the parameter vector $\boldsymbol{\theta}$ [Jacot et al., 2018]. In fact, the output of an overparameterized neural network, for any fixed input $\boldsymbol{X}$, asymptotically mimics a linear reward function with respect to the perturbation vector $\boldsymbol{\delta}$—essentially reducing to the purely linear case.

Corollary 3.2 (Overparameterized Neural Networks)**.** *For asymptotically wide neural networks, linearization via the NTK approximation reduces the attack design to a convex quadratic program with D variables as in Theorem 3.1, where D is the number of parameters in $\boldsymbol{\theta}$. This program can be solved in $\widetilde{\mathcal{O}}(|\mathcal{I}|^3 + D|\mathcal{I}|^2)$ operations.*

3.2 Theoretical Guarantees

In this section, we present two sets of theoretical guarantees. The first one, stated in Theorem 3.3, ensures that under mild conditions of "non-degeneracy" of the data-generating distributions $P_1, \ldots, P_K$, and as long as we are in a high-dimensional regime (i.e., $d \geq TK$), there always exists an attack (with probability one over the sampling of datasets $\mathcal{D}_i$) that can derail the bandit to *any* desired trajectory.

Theorem 3.3 (Feasibility Guarantee)**.** *Assume the data-generating distributions $P_1, \ldots, P_K$ are non-singular, i.e.,*

$$\lambda_{\min}(\mathsf{Cov}(P_i)) > 0,$$

where $\mathsf{Cov}(\cdot)$ denotes the $d \times d$ covariance operator and $\lambda_{\min}(\cdot)$ denotes the minimum eigenvalue. If the dimension satisfies $d > (T-K)(K-1)$, then with probability one, all optimization problems corresponding to the full-trajectory attack, the trajectory-free attack, and the OSA attack are feasible.

The proof can be found in Appendix B, and relies on simple arguments from linear algebra and anti-concentration of non-degenerate distributions. A stronger result, requiring more technical machinery, is stated in Theorem 3.4. There, we show that in the high-dimensional regime, and under stronger assumptions on the distributions $P_1, \ldots, P_K$, the magnitude of the attack decays as $\widetilde{\mathcal{O}}(d^{-1/2})$ as the dimension increases. We conjecture that a more general result can be established for near-product measures, which capture a much broader family of distributions; however, this falls beyond the scope of the current paper.

In particular, let $\boldsymbol{\mu}_i \triangleq \mathbb{E}_{P_i}[\boldsymbol{X}]$ for $i \in [K]$, and assume for simplicity that $\|\boldsymbol{\mu}_i\|_2 = 1$ for all i. In the case of the linear reward model, we further assume that $\|\boldsymbol{w}\|_2 = 1$, with $\boldsymbol{\mu}_{i^*}^\top \boldsymbol{w} = 1$, while for $i \neq i^*$ we have $\boldsymbol{\mu}_i^\top \boldsymbol{w} = 0$. Such a configuration typically arises in high-dimensional settings, where independently drawn random vectors tend to be nearly orthogonal. We emphasize that these assumptions are introduced solely to simplify notation and exposition in subsequent sections; they are not essential to our analysis and do not affect the design of our algorithms.

Theorem 3.4 (Imperceptibility of High-dimensional Attack)**.** *Consider the assumptions mentioned above for $d > 2$. For simplicity, assume all $P_1, \ldots, P_K$ are product measures and $\mathsf{Cov}(P_i) = \boldsymbol{I}$ for all $i \in [K]$, where $\boldsymbol{I}$ is the $d \times d$ identity matrix. Then, provided $d \geq KT$ in the high-dimensional regime, the full-trajectory attack of (2) satisfies*

$$\|\boldsymbol{\delta}^*\|_2 \;\leq\; \mathcal{O}\left(\sqrt{\frac{T^3 \log T \cdot \log d}{Kd}}\right), \tag{6}$$

with probability at least $1 - 2d^{-1}$.

It should be emphasized that Theorem 3.4 is a non-asymptotic result that applies to any triple (d, T, K) satisfying the stated conditions. The full proof, deferred to Appendix B, leverages tools from linear algebra, optimization, and random matrix theory. Here, we give a simplified proof sketch explaining why attacks on high-dimensional bandits are feasible:

It is well known that estimating the *mean* of high-dimensional random vectors becomes increasingly difficult as the dimension grows. In other words, fluctuations across the many coordinates of a high-dimensional vector accumulate, making accurate estimation challenging. A multi-armed bandit, viewed naively, attempts to estimate the means of several high-dimensional distributions and then select the one that optimizes its reward function. However, when the number of observations per arm, roughly $\simeq T/K$, is much smaller than the dimension d, such estimations are unreliable. In this setting, a simple manipulation of the reward function—provided it has sufficiently many degrees of freedom, such as in the linear case or in overparameterized neural networks—can easily deceive the bandit. Theorem 3.4 further shows that the required attack magnitude *decreases* as the dimension grows.

4 Experiments

In this section, we conduct a series of experiments to assess the effectiveness of our proposed attack method. Our study focuses on four central questions: Q1: How does the attack's performance differ between linear and non-linear reward models, and to what extent does the proposed heuristic enhance its effectiveness? (Sec. 4.1) Q2: What is the impact of high dimensionality on attack performance? (Sec. 4.2) Q3: Can random noise perturbations achieve results comparable to our method? (Sec. 4.3) Q4: How does increasing the width of the non-linear reward model affect the attack success rate? (Sec. 4.4) Q5: Can our attack be applied to bandit algorithms other than UCB? (Sec. 4.5)

4.1 Attack Performance Across Reward Models

In this experiment, we evaluate the performance of our attack on reward models, examining its effectiveness in both linear and non-linear settings. To provide a more comprehensive assessment, we vary key parameters such as K and D. Furthermore, We evaluate all variants of our method: *OSA* , *Trajectory-Free*, and *Full-Trajectory*. Among them, the *Full-Trajectory* method offers the adversary the greatest control, enabling them to shape the entire trajectory according to their objective. However, this advantage comes at a cost: as time progresses, the number and complexity of the constraints grow, making it increasingly difficult to identify a perturbation that satisfies all conditions.

Reward Models	Linear Reward Model			Non-linear Reward Model		
	K	d	Attack Time	K	$\max_i\ W_i$	Attack Time
OSA Attack						
	3	10000	00:01	3	3000	00:01
	3	1000	00:00	3	6000	00:03
	5	10000	00:00	5	3000	00:04
	5	1000	00:00	5	6000	00:09
Trajectory-Free Attack						
	3	10000	00:25	3	3000	01:40
	3	1000	00:03	3	6000	04:05
	5	10000	00:23	5	3000	01:21
	5	1000	00:03	5	6000	03:21
Full Trajectory Attack						
	3	10000	01:31	3	3000	06:11
	3	1000	00:10	3	6000	13:34
	5	10000	03:45	5	3000	13:51
	5	1000	00:23	5	6000	30:45

Table 1: Comparison of attack performance on linear and non-linear reward models using OSA, Trajectory-Free *and* Full Trajectory *methods. All attacks achieve 100% success rate ($ASR = 100$). The table highlights the efficiency of the* OSA *method in terms of attack time (minute:second), and the effect of varying parameters K and D (or $\max_i\ W_i$) on attack duration.*

This experiment focuses on two aspects: (1) demonstrating that both linear and non-linear reward models can be successfully attacked, and (2) comparing the attack time of the *OSA* and *Trajectory* methods. As shown in Table 1, all variants of our attack achieve a 100% attack success rate (ASR) on the UCB algorithm. From the results, we observe that the *OSA* method substantially reduces the time required to generate perturbations. For instance, with $K = 5$ and $\max_i\ W_i = 6000$, the *OSA* attack completes in approximately seconds, while the *Trajectory-Free* method takes 3 minutes and 21 seconds, and the *Full-Trajectory* method requires 30 minutes and 45 seconds. This demonstrates the clear efficiency advantage of the *OSA* approach. We also present additional experiments in Appendix D, showing that the number of constraints in *OSA* scales as $O(\log T)$.

4.2 Impact of High Dimensionality on Attack Performance

In this experiment, we validate our claim that increasing input dimensionality makes the bandit algorithm more vulnerable to attacks. To investigate this, we vary the input dimensionality while keeping all other parameters fixed. We observe that both the ℓ_2 and ℓ_∞ norms of the perturbation decrease rapidly, indicating that in higher-dimensional settings, smaller perturbations suffice to successfully attack the algorithm. Additionally, we reformulate the optimization problem from a feasibility program to a quadratic program to evaluate how effectively our method approximates the optimal solution. In the low-dimensional regime, the *OSA* solution shows a small gap relative to the *Optimal* solution. However, as the input dimensionality increases, this gap diminishes, demonstrating that our *OSA* method achieves near-optimal attacks in higher-dimensional settings. We can conclude that our *OSA* method not only identifies a perturbation quickly, but also produces one that is very close to the optimal solution.

Moreover, we investigate whether our attack in high-dimension is trivial. We visualize the geometric of the attack vector relative to other vectors, we reduce the perturbation space to three dimensions using PCA. As shown in Fig. 3a and Fig. 3b, the perturbations are neither trivial nor simply the reverse of the vector w. This demonstrates that the attack is subtle, imperceptible, and not easily detectable from the bandit algorithm's perspective.

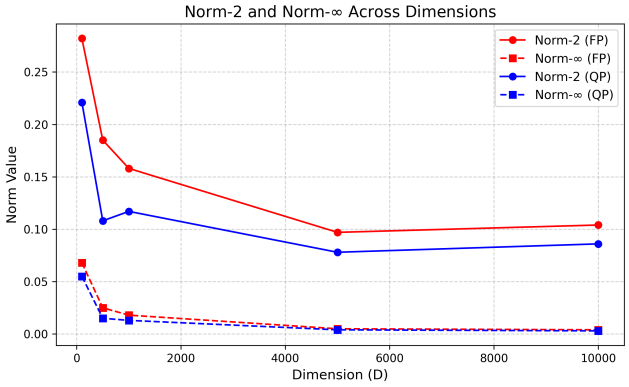


Figure 1: Effect of increasing input dimensionality on attack magnitude. Both the ℓ_2 and ℓ_∞ norms of the perturbation decrease as the input dimensionality increases, indicating that higher-dimensional inputs are more vulnerable to attacks. The plot also shows that the difference between the OSA *and* Optimal *perturbations diminishes with higher dimensionality. Experimental settings: $T = 100$, $K = 3$, ASR$= 100\%$.*

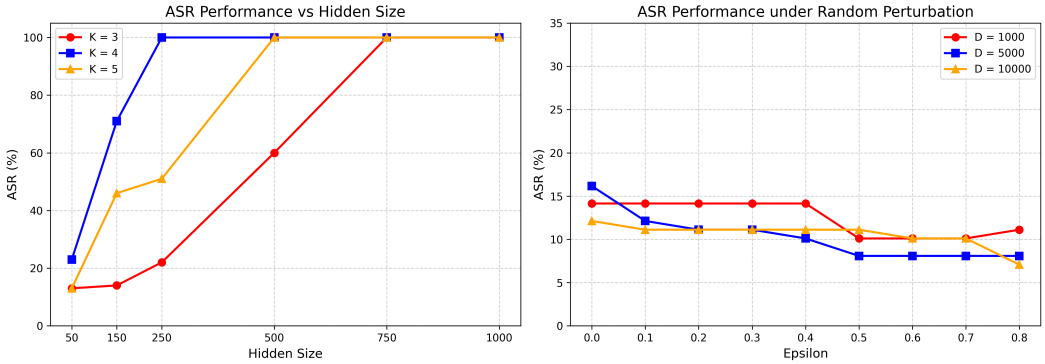


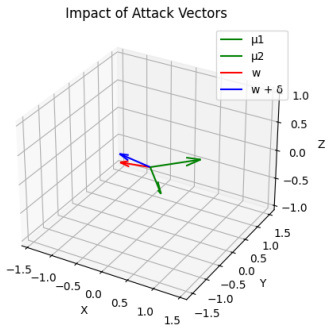
*Figure 2: **Left**: Attack success rate as a function of the hidden layer width in the reward model ($T = 100$). The results show that a sufficiently wide hidden layer is required for the attack to succeed; once the width exceeds 750 neurons, the ASR reaches 100%. **Right**: Attack success rate versus the ℓ_2 norm of random noise perturbations. The results show that even with increasing perturbation magnitude, the ASR remains nearly constant, demonstrating that random noise is largely ineffective in attacking the bandit algorithm.*

4.3 Effect of Random Noise Perturbations on Attack Performance

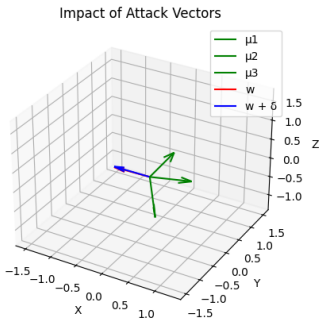
We demonstrate that applying a random noise perturbation, even with a larger ℓ_2 norm, fails to produce effective attacks. As shown in the right part of Figure 2, when no perturbation is applied, the attack success rate (ASR) is approximately 25%. Increasing the ℓ_2 norm of the random perturbation has little impact on the ASR, which remains nearly constant across a wide range of perturbation magnitudes. This indicates that simple random noise is largely ineffective at compromising the bandit algorithm. In contrast, our targeted attack methods are able to achieve significantly higher ASR, highlighting the necessity of carefully designed perturbations rather than relying on random noise.

4.4 Effect of Non-Linear Reward Model Width on Attack Success

In this experiment, we investigate the impact of the reward model's width on attack performance. For our theoretical assumptions to hold in practice, the network's linear approximation must be valid. To achieve this, the reward model is designed with a single hidden layer of sufficient width. As the number of neurons increases, the linear approximation becomes more accurate [Jacot et al., 2018]. Our results indicate that having enough neurons in this hidden layer is critical for a successful attack. As illustrated in the left part of Fig. 2, once the number of neurons exceeds 750, the attack success rate reaches 100%. Implementation details of the reward model training are provided in Appendix E.



(a) $T = 100, K = 3, D = 1000$, OSA



(b) $T = 100, K = 4, D = 1000$, OSA

Figure 3: 3D visualization of perturbations for different values of K.

Reward Models	Linear Reward Model			Non-linear Reward Model		
	K	d	Attack Time	K	$\max_i W_i$	Attack Time
ϵ-greedy Attack						
	3	10000	00:00	3	3000	00:22
	3	1000	00:00	3	6000	00:12
	5	10000	00:00	5	3000	00:18
	5	1000	00:00	5	6000	00:13
Explore-Then-Commit (ETC)						
	3	10000	00:00	3	3000	00:03
	3	1000	00:00	3	6000	00:06
	5	10000	00:00	5	3000	00:04
	5	1000	00:00	5	6000	00:09

Table 2: Comparison of attack performance on linear and non-linear reward models using the ϵ-greedy *and* ETC *methods. All attacks achieve a* 100% *success rate (*$ASR = 100$*). The table illustrates how varying the parameters* K *and* D *(or* $\max_i W_i$*) affects the attack duration. In this experiment, we set* $T = 100$ *and the number of exploration rounds for* ETC *to* $m = 5$*.*

4.5 Attacking Bandit Algorithms Beyond UCB

Throughout this paper, we use the UCB [Auer, 2002] algorithm as our main bandit algorithm. To evaluate the robustness of our attack, we also consider scenarios where the adversary employs other bandit algorithms, specifically ETC [Garivier et al., 2016] and ϵ-greedy [Dann et al., 2022]. Table 2 presents the results of our attack on these algorithms, showing that it achieves a 100% ASR across all these algorithms. Additionally, our OSA method significantly accelerates the attack on the ϵ-greedy algorithm, with the attack time remaining low for both methods. For the ETC algorithm, we set the number of exploration rounds to $m = 5$, meaning each arm is selected exactly five times before the commit phase begins. For the ϵ-greedy algorithm, we set the initial ϵ to 0.1 and apply decay over time, with a minimum value of 0.01. Additional details about these algorithms are provided in Appendix F.

5 Conclusion

In this work, we investigate the vulnerabilities of offline bandit algorithms used to evaluate machine learning models, including generative models and large language models. We first show that publicly sharing reward model weights on platforms like Hugging Face exposes these models to adversarial attacks and highlight the critical role of logged data in offline bandit algorithms. We then introduce a novel adversarial threat, demonstrating that even small, imperceptible perturbations to reward model weights can completely hijack bandit behavior. Our analysis begins with linear reward models, where we show that such models are highly susceptible to manipulation in high-dimensional settings. We

then extend our approach to more complex non-linear reward models, such as ReLU-based neural networks. Furthermore, when an attacker's goal is not to hijack the entire trajectory but only to prevent the selection of the optimal arm, we propose an on–off heuristic that is both faster and as effective as our main attack method, activating only when necessary. Finally, we provide theoretical support showing how perturbations can systematically steer bandit behavior in proportion to input dimensionality.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Kingma DP Ba J Adam et al. A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 1412(6), 2014.

Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of machine learning research*, 3(Nov):397–422, 2002.

Olivier Chapelle, Eren Manavoglu, and Romer Rosales. Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):1–34, 2014.

Jaemin Cho, Yushi Hu, Jason Baldridge, Roopal Garg, Peter Anderson, Ranjay Krishna, Mohit Bansal, Jordi Pont-Tuset, and Su Wang. Davidsonian scene graph: Improving reliability in fine-grained evaluation for text-to-image generation. In *ICLR*, 2024.

Ferdinando Cicalese, Eduardo Laber, Marco Molinaro, et al. Teaching with limited information on the learner's behaviour. In *International Conference on Machine Learning*, pages 2016–2026. PMLR, 2020.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*, pages 1310–1320. PMLR, 2019.

Chris Dann, Yishay Mansour, Mehryar Mohri, Ayush Sekhari, and Karthik Sridharan. Guarantees for epsilon-greedy reinforcement learning with function approximation. In *International conference on machine learning*, pages 4666–4689. PMLR, 2022.

Sanjoy Dasgupta, Daniel Hsu, Stefanos Poulis, and Xiaojin Zhu. Teaching a black-box learner. In *International Conference on Machine Learning*, pages 1547–1555. PMLR, 2019.

Evrard Garcelon, Baptiste Roziere, Laurent Meunier, Jean Tarbouriech, Olivier Teytaud, Alessandro Lazaric, and Matteo Pirotta. Adversarial attacks on linear contextual bandits. *Advances in Neural Information Processing Systems*, 33:14362–14373, 2020.

Aurélien Garivier, Tor Lattimore, and Emilie Kaufmann. On explore-then-commit strategies. *Advances in Neural Information Processing Systems*, 29, 2016.

David M Gay, Michael L Overton, and Margaret H Wright. A primal-dual interior method for nonconvex nonlinear programming. In *Advances in Nonlinear Programming: Proceedings of the 96 International Conference on Nonlinear Programming*, pages 31–56. Springer, 1998.

Eugene Golikov, Eduard Pokonechnyy, and Vladimir Korviakov. Neural tangent kernel: A survey. *arXiv preprint arXiv:2208.13614*, 2022.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144, 2020.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Seyed Mohammad Hadi Hosseini, Amir Mohammad Izadi, Ali Abdollahi, Armin Saghafian, and Mahdieh Soleymani Baghshah. T2i-fineeval: Fine-grained compositional metric for text-to-image evaluation. *arXiv preprint arXiv:2503.11481*, 2025.

Xiaoyan Hu, Ho-fung Leung, and Farzan Farnia. An online learning approach to prompt-based selection of generative models and llms. In *Forty-second International Conference on Machine Learning*, 2024.

Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A Smith. Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20406–20417, 2023.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Kwang-Sung Jun, Lihong Li, Yuzhe Ma, and Jerry Zhu. Adversarial attacks on stochastic bandits. *Advances in neural information processing systems*, 31, 2018.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028*, 2014.

Branislav Kveton, Csaba Szepesvari, Zheng Wen, and Azin Ashkan. Cascading bandits: Learning to rank in the cascade model. In *International conference on machine learning*, pages 767–776. PMLR, 2015.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

Fang Liu and Ness Shroff. Data poisoning attacks on stochastic bandits. In *International Conference on Machine Learning*, pages 4042–4050. PMLR, 2019.

Guanlin Liu and Lifeng Lai. Action-manipulation attacks on stochastic bandits. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3112–3116. IEEE, 2020.

Yuzhe Ma, Kwang-Sung Jun, Lihong Li, and Xiaojin Zhu. Data poisoning attacks in contextual bandits. In *International Conference on Decision and Game Theory for Security*, pages 186–204. Springer, 2018.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

Parham Rezaei, Farzan Farnia, and Cheuk Ting Li. Be more diverse than the most diverse: Optimal mixtures of generative models via mixture-UCB bandit algorithms. In *The Thirteenth International Conference on Learning Representations*, 2025.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

Yuta Saito and Masahiro Nomura. Hyperparameter optimization can even be harmful in off-policy learning and how to deal with it. *arXiv preprint arXiv:2404.15084*, 2024.

Jaskirat Singh and Liang Zheng. Divide, evaluate, and refine: Evaluating and improving text-to-image alignment with iterative vqa feedback. *Advances in Neural Information Processing Systems*, 36:70799–70811, 2023.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International conference on machine learning*, 2014.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Matilde Tullii, Solenne Gaucher, Nadav Merlis, and Vianney Perchet. Improved algorithms for contextual dynamic pricing. *Advances in Neural Information Processing Systems*, 37:126088–126117, 2024.

Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge university press, 2019.

A Related Work

While adversarial attacks on supervised learning models have been extensively studied [Szegedy et al., 2014, Moosavi-Dezfooli et al., 2017, Cicalese et al., 2020, Cohen et al., 2019, Dasgupta et al., 2019], the vulnerabilities of bandit algorithms remain less explored. This is particularly important in settings where reward models [Singh and Zheng, 2023, Cho et al., 2024, Hu et al., 2023, Hosseini et al., 2025] are combined with bandit algorithms as evaluators [Rezaei et al., 2025, Hu et al., 2024], highlighting the need to understand prior adversarial attacks on bandit algorithms and to prepare for developing new attack strategies. Recent studies have investigated adversarial attacks on multi-armed bandits (MABs) [Jun et al., 2018, Liu and Shroff, 2019, Liu and Lai, 2020] and linear contextual bandits [Ma et al., 2018, Garcelon et al., 2020]. In reward poisoning attacks, an adversary manipulates the reward signals received by the agent from the environment [Jun et al., 2018, Liu and Shroff, 2019]. By subtly altering these rewards, the attacker can bias the learning process, causing the agent to favor specific actions or arms, which may lead to suboptimal or adversary-driven behavior. In linear contextual bandits, adversarial attacks have focused on two main strategies. The first is reward poisoning, where the adversary modifies the rewards associated with chosen actions to steer the agent toward particular arms [Ma et al., 2018, Garcelon et al., 2020]. The second is context poisoning, in which the attacker alters the observed context vectors while leaving the rewards unchanged, misleading the agent into making suboptimal or attacker-desired decisions [Garcelon et al., 2020]. Together, these attacks demonstrate the vulnerabilities of linear contextual bandits to both reward and context-based manipulations. Our work differs from prior studies in that, rather than directly attacking the rewards, we target the reward models themselves. This approach is particularly relevant in the emerging usage of bandit algorithms as evaluation metrics. We show that in high-dimensional settings, reward models are vulnerable, making it possible for an attacker to easily manipulate them and thereby hijack the behavior of the bandit algorithm.

B Proofs

Proof of Theorem 3.1. We first establish the result for the case of full-trajectory attacks. The other two variants differ only in that they employ a subset of the $(T-K)(K-1)$ constraints; hence, both the optimization formulation and the feasibility properties can be deduced directly from the analysis of the full-trajectory case. In this regard, we define the residual outputs $\boldsymbol{Z}$ as $\boldsymbol{Z}_j^{(i)} \triangleq \boldsymbol{X}_j^{(i)} - \boldsymbol{\mu}_i$, therefore we have $\mathbb{E}[\boldsymbol{Z}_j^{(i)}] = \mathbf{0}$ for all valid i and j. Let us define $\boldsymbol{S}_j^{(i)} \in \mathbb{R}^d$ for $i \in [K]$ and $k \in [n_i]$ as

$$\boldsymbol{S}_k^{(i)} \triangleq \frac{1}{k}\sum_{j=1}^{k} \boldsymbol{X}_j^{(i)} = \boldsymbol{\mu}_i + \frac{1}{k}\sum_{j=1}^{k} \boldsymbol{Z}_j^{(i)}\,.$$

In this regard, we have $\mathbb{E}[\boldsymbol{S}_k^{(i)}] = \boldsymbol{\mu}_i$, for all i and regardless of $k \geq 1$. Therefore, the second sub-problem above reduces to

$$\begin{aligned} &\underset{\boldsymbol{\delta}}{\operatorname{argmin}} \ \| \boldsymbol{\delta} \|_2^2 \\ &\text{subject to} \ \ (\boldsymbol{w}+\boldsymbol{\delta})^\top \left[\boldsymbol{S}_{N_i(t)}^{(i)} - \boldsymbol{S}_{N_j(t)}^{(j)}\right] > \sqrt{2\log t}\left(N_j^{-1/2}(t) - N_i^{-1/2}(t)\right) \\ &\qquad\qquad (\text{for } i = \widetilde{A}_t,\ \forall j \neq i,\ K < t \leq T), \end{aligned} \tag{7}$$

and $N_i(t) \triangleq \sum_{t'=1}^{t-1} \mathbb{1}(\widetilde{A}_{t'} = i)$. The above set of inequality constraints can be further simplified via introducing new symbols, and hence the above convex feasibility problem can be rewritten as

$$\begin{aligned} &\underset{\boldsymbol{\delta}}{\operatorname{argmin}} \ \| \boldsymbol{\delta} \|_2^2 \\ &\text{subject to} \ \ \boldsymbol{\delta}^\top \boldsymbol{T}_{j,t} > R_{j,t}, \quad \forall j \in [K-1],\ K < t \leq T, \end{aligned} \tag{8}$$

where the vectors $\boldsymbol{T}_{j,t}$ and scalars $R_{j,t}$ depend on samples in datasets $\mathcal{D}_i$ and the target sequence $\widetilde{A}_t$. More precisely, we have

$$\boldsymbol{T}_{j,t} \triangleq (\boldsymbol{\mu}_i - \boldsymbol{\mu}_{j'}) + \left[\frac{1}{N_i(t)}\sum_{l=1}^{N_i(t)} \boldsymbol{Z}_l^{(i)} - \frac{1}{N_{j'}(t)}\sum_{l=1}^{N_{j'}(t)} \boldsymbol{Z}_l^{(j')}\right], \tag{9}$$

$$R_{j,t} \triangleq -\boldsymbol{w}^\top \boldsymbol{T}_{j,t} + \sqrt{2\log t}\left(N_{j'}^{-1/2}(t) - N_i^{-1/2}(t)\right),$$

where we have $i \triangleq \widetilde{A}_t$, and $j' \in [K-1]$ is the index of the set $[K]$ after removing $\widetilde{A}_t$.

In the case where the reward model is a neural network rather than linear, the NTK approximation in (5), together with the assumption $\max_i W_i \gg 1$, yields an analogous QP form, with modified definitions of $\boldsymbol{T}_{j,t}$ and $R_{j,t}$:

$$\boldsymbol{T}_{j,t} \triangleq \frac{1}{N_i(t)} \sum_{l=1}^{N_i(t)} \nabla_{\boldsymbol{\theta}} \mathrm{NN}_{\boldsymbol{\theta}}(\boldsymbol{X}_l^{(i)}) - \frac{1}{N_{j'}(t)} \sum_{l=1}^{N_{j'}(t)} \nabla_{\boldsymbol{\theta}} \mathrm{NN}_{\boldsymbol{\theta}}(\boldsymbol{X}_l^{(j')}), \tag{10}$$

$$R_{j,t} \triangleq -\left[\frac{1}{N_i(t)} \sum_{l=1}^{N_i(t)} \mathrm{NN}_{\boldsymbol{\theta}}(\boldsymbol{X}_l^{(i)}) - \frac{1}{N_{j'}(t)} \sum_{l=1}^{N_{j'}(t)} \mathrm{NN}_{\boldsymbol{\theta}}(\boldsymbol{X}_l^{(j')})\right] + \sqrt{2\log t}\left(N_{j'}^{-1/2}(t) - N_i^{-1/2}(t)\right).$$

This completes the proof. □

Proof of Theorem 3.3. Without loss of generality, we assume $i^* = 1$. Recalling the notation from the proof of Theorem 3.1, for all valid j and t, we have

$$\mathbb{E}\left[\boldsymbol{w}^\top \boldsymbol{T}_{j,t}\right] = \begin{cases} 1 & \widetilde{A}_t = 1 \\ 0 & \widetilde{A}_t \neq 1, j \neq 1 \\ -1 & \widetilde{A}_t \neq 1, j = 1 \end{cases}, \tag{11}$$

$$\mathsf{Var}\left[\boldsymbol{w}^\top \boldsymbol{T}_{j,t}\right] = \frac{1}{N_i(t)} \boldsymbol{w}^\top \mathsf{Cov}_{P_i} \boldsymbol{w} + \frac{1}{N_{j'}(t)} \boldsymbol{w}^\top \mathsf{Cov}_{P_{j'}} \boldsymbol{w}.$$

Note that both $\boldsymbol{T}_{j,t}$ and $R_{j,t}$ are deterministic. Also, we have

$$|R_{j,t}| \leq \left|\boldsymbol{w}^\top \boldsymbol{T}_{j,t}\right| + \sqrt{2\log t}\left(1 - t^{-1/2}\right). \tag{12}$$

The set of constraints can be rewritten in the following matrix form:

$$\mathbb{T}\,\boldsymbol{\delta} \succeq \boldsymbol{R}, \tag{13}$$

where $C \times d$ matrix $\mathbb{T}$ and C-dimensional vector $\boldsymbol{R}$ are formed via row-wise concatenation of $\boldsymbol{T}_{j,t}$s and $R_{j,t}$s, respectively. Here, C denotes the number of constraints which is at most $(K-1)(T-K)$. Note that we have replaced $\succ$ with $\succeq$ by assuming infinitesimally small margins in the constraints. This modification simplifies the subsequent part of the proof. Also, it should be noted that both $\mathbb{T}$ and $\boldsymbol{R}$ are random.

We need to show that the stochastic inequality system $\mathbb{T}\,\boldsymbol{\delta} \succeq \boldsymbol{R}$ is feasible with probability 1. We proceed by contradiction. Without loss of generality, assume that the first inequality $\boldsymbol{T}_1^\top \boldsymbol{\delta} \geq R_1$ is infeasible (with positive probability), given that the remaining $C-1$ inequalities are satisfied. Formally, assume

$$\mathbb{P}\Big(\Big\{\,\boldsymbol{\delta} \;\Big|\; \boldsymbol{T}_1^\top \boldsymbol{\delta} \geq R_1\Big\} \cap \Big\{\,\boldsymbol{\delta} \;\Big|\; \boldsymbol{T}_i^\top \boldsymbol{\delta} \geq R_i,\ i = 2, \ldots, C\Big\} = \varnothing\Big) > 0.$$

Then we must have

$$\mathbb{P}\Big(\Big\{\,\boldsymbol{\delta} \;\Big|\; \mathbb{T}\,\boldsymbol{\delta} = \boldsymbol{R}\Big\} = \varnothing\Big) > 0,$$

that is, the stochastic equation system $\mathbb{T}\,\boldsymbol{\delta} = \boldsymbol{R}$ is unsolvable with positive probability. Since the number of variables d exceeds the maximum number of equations $(T-K)(K-1)$, this can only occur if the rows of $\mathbb{T}$ fail to be linearly independent with positive probability.

However, this is impossible by construction of $\mathbb{T}$ in (9). Indeed, since all residuals $\{\boldsymbol{Z}_j^{(i)}\}_{j \in [n_i]}$ are independently drawn and each distribution P_i is non-degenerate (i.e., $\lambda_{\min}(\mathsf{Cov}(P_i)) > 0$ for every $i \in [K]$), the rows of $\mathbb{T}$ are distributed in general position and are therefore linearly independent with probability 1. This completes the proof. □

Proof of Theorem 3.4. We begin with the following lemma:

Lemma B.1. *Assume $(T-K)(K-1) < d$ and consider the constrained optimization problem*

$$\boldsymbol{\delta}^* \triangleq \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \ \| \boldsymbol{\delta} \|_2^2 \quad \textit{subject to} \quad \mathbb{T}\,\boldsymbol{\delta} \succeq \boldsymbol{R}. \tag{14}$$

Define $\widetilde{\boldsymbol{\delta}} \triangleq \mathbb{T}^\dagger \boldsymbol{R}$, where $\dagger$ denotes the Moore–Penrose pseudo-inverse, i.e., $\widetilde{\boldsymbol{\delta}} = (\mathbb{T}\mathbb{T}^\top)^{-1}\mathbb{T}^\top \boldsymbol{R}$. Then we have $\| \boldsymbol{\delta}^* \|_2 \leq \|\widetilde{\boldsymbol{\delta}}\|_2$.

Proof. The argument is straightforward. Removing any constraint from (14) (equivalently, removing a row of $\mathbb{T}$) cannot increase $\| \boldsymbol{\delta}^* \|_2$. Hence, the maximum possible ℓ_2-norm of $\boldsymbol{\delta}^*$ occurs when all constraints are active. Formally,

$$\begin{aligned} \| \boldsymbol{\delta}^* \|_2 &\leq \left\| \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \ \| \boldsymbol{\delta} \|_2^2 \quad \text{subject to} \quad \mathbb{T}\,\boldsymbol{\delta} = \boldsymbol{R} \right\|_2 \\ &= \|\widetilde{\boldsymbol{\delta}}\|_2, \end{aligned} \tag{15}$$

where the last equality follows from the definition of the Moore–Penrose pseudo-inverse, which yields the unique minimum-ℓ_2-norm solution to an underdetermined linear system. Also, according to Theorem 3.3, we know the linear equation system $\mathbb{T}\,\boldsymbol{\delta} = \boldsymbol{R}$ is solvable (hence $\widetilde{\boldsymbol{\delta}}$ exists) with probability 1. This completes the proof. □

Next, we bound $\|\widetilde{\boldsymbol{\delta}}\|_2$. In order to do this, first note that

$$\|\widetilde{\boldsymbol{\delta}}\|_2 \ \leq \ \frac{\|\boldsymbol{R}\|_2}{\sigma_{\min}(\mathbb{T})}, \tag{16}$$

where $\sigma_{\min}(\cdot)$ denotes the minimum singular value. Therefore, we need to: i) find a high-probability bound for $\sigma_{\min}(\mathbb{T})$, and ii) establish a similar bound for $\|\boldsymbol{R}\|_2$. This requires an appropriate matrix representation for the construction of $\mathbb{T}$, whose rows are defined in (9):

$$\text{any row of } \mathbb{T} = \boldsymbol{\mu}_i - \boldsymbol{\mu}_j + \left[\frac{1}{N_i(t)} \sum_{l=1}^{N_i(t)} \boldsymbol{Z}_l^{(i)} - \frac{1}{N_j(t)} \sum_{l=1}^{N_j(t)} \boldsymbol{Z}_l^{(j)} \right], \tag{17}$$

for any two properly defined indices i, j. We then leverage existing tools from random matrix theory to establish our main results.

Matrix-form generation of $\mathbb{T}$. Let us define the residual matrix Γ as follows:

$$\Gamma \triangleq \left[\boldsymbol{Z}_1^{(1)} \mid \cdots \mid \boldsymbol{Z}_{n_1}^{(1)} \mid \cdots \mid \boldsymbol{Z}_1^{(K)} \mid \cdots \mid \boldsymbol{Z}_{n_K}^{(K)} \right]^\top, \tag{18}$$

which is an $(n_1 + \cdots + n_K) \times d$ dimensional matrix that collects all the $\boldsymbol{Z}$ vectors as its rows. According to the assumptions in the statement of the theorem, Γ is a random matrix with independent rows. Moreover, since each data-generating distribution is assumed to be a product measure, the entries in each row are also independent and have variance 1. We are particularly interested in a sub-matrix of Γ, denoted by $\Gamma^* \in \mathbb{R}^{(T-K)\times d}$, which consists only of those vectors $\boldsymbol{Z}_j^{(i)}$ that correspond to arms pulled during the target trajectory for time-steps $K < t \leq T$. Since the target trajectory is fixed and deterministic, Γ^* is still a random matrix with independently distributed entries of unit variance.

Recall that $N_i(T)$ denotes the number of times arm i has been pulled up to the horizon T for a given fixed target trajectory. For any non-degenerate trajectory $\widetilde{A}_t$, we have $N_i(T) \leq \mathcal{O}(T/K)$, $\forall i \in [K]$. Let m denote the maximum number of times any arm has appeared during the trajectory, so that $m \leq \mathcal{O}(T/K)$. Now, consider the following family of matrices:

$$\Lambda_n \triangleq \begin{bmatrix} 1 & 1/2 & \cdots & 1/n \\ 0 & 1/2 & \cdots & 1/n \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/n \end{bmatrix}, \quad \forall n \in \mathbb{N}. \tag{19}$$

Lemma B.2. *we have* $\lambda_{\min}(\Lambda_n) = 1/n$.

Proof. Proof is straightforward. For some fixed $n \in \mathbb{N}$, let us denote the eigenvalues of Λ_n by $\lambda_1, \dots, \lambda_n$. Then, we have $\det(\Lambda_n - \lambda_i \boldsymbol{I}) = 0$ for all $i \in [n]$. On the other hand, since $\Lambda_n - \lambda_i \boldsymbol{I}$ is upper-triangular, we have

$$\det(\Lambda_n - \lambda_j \boldsymbol{I}) = \prod_{i=1}^{n} \left(\frac{1}{i} - \lambda_j \right), \quad \forall j \in [n].$$

Therefore, we must have $\lambda_j = 1/j$ for all $j \in [n]$ and consequently $\lambda_{\min}(\Lambda_n) = 1/n$, which completes the proof. □

Consider the matrix Λ_m, and let us denote its columns by $\boldsymbol{v}_1, \dots, \boldsymbol{v}_m \in \mathbb{R}^m$. Based on the definition of $\mathbb{T}$ in (9), each row of $\mathbb{T}$ consists of a mean-difference term $\boldsymbol{\mu}_i - \boldsymbol{\mu}_j$ (for properly defined indices i, j), together with a term that depends solely on residuals. This residual part can be rewritten as

$$\frac{1}{N_i(t)} \sum_{l=1}^{N_i(t)} \boldsymbol{Z}_l^{(i)} - \frac{1}{N_j(t)} \sum_{l=1}^{N_j(t)} \boldsymbol{Z}_l^{(j)} = \left(\left[\boldsymbol{0} \;\middle|\; \boldsymbol{v}_{N_i(t)} \;\middle|\; \boldsymbol{0} \right] - \left[\boldsymbol{0} \;\middle|\; \boldsymbol{v}_{N_j(t)} \;\middle|\; \boldsymbol{0} \right] \right)^\top \Gamma^*, \tag{20}$$

where $\left[\boldsymbol{0} \;\middle|\; \boldsymbol{v}_i \;\middle|\; \boldsymbol{0} \right]$ denotes a vector of dimension $T - K$ obtained by padding $\boldsymbol{v}_i$ with zeros both before and after. In this regard, the random matrix $\mathbb{T}$ can be expressed as

$$\mathbb{T} = \boldsymbol{M} + \boldsymbol{V}^\top \Gamma^*, \tag{21}$$

where $\boldsymbol{M}$ is a low-rank matrix consisting of the mean-difference terms $\boldsymbol{\mu}_i - \boldsymbol{\mu}_j$ as its rows. This includes (potentially) all possible pairwise differences between the means of the K distributions $P_1, \dots, P_K$, with repetitions as required. Note that $\boldsymbol{M}$ can have at most K nonzero singular values. The matrix $\boldsymbol{V} \in \mathbb{R}^{(T-K) \times (T-K)(K-1)}$ collects all combinations of vectors of the form

$$\left[\boldsymbol{0} \;\middle|\; \boldsymbol{v}_{N_i(t)} \;\middle|\; \boldsymbol{0} \right] - \left[\boldsymbol{0} \;\middle|\; \boldsymbol{v}_{N_j(t)} \;\middle|\; \boldsymbol{0} \right],$$

as its columns. The matrix $\boldsymbol{V}$ is fully determined by the target trajectory and can be regarded as a row/column-wise augmentation of Λ_m. Therefore,

$$\sigma_{\min}(\boldsymbol{V}) \;\geq\; \mathcal{O}(\sigma_{\min}(\Lambda_m)) = \mathcal{O}(1/m) \;\geq\; \mathcal{O}(K/T)\,. \tag{22}$$

Consequently, we obtain

$$\sigma_{\min}(\mathbb{T}) \;\geq\; \mathcal{O}(K/T) \cdot \sigma_{\min}(\Gamma^*). \tag{23}$$

What remains, with respect to bounding the denominator in (16), is to establish a high-probability bound on $\sigma_{\min}(\Gamma^*)$. To this aim, we invoke a well-established result from random matrix theory. There are several known high-probability bounds for the minimum and maximum singular values of random matrices with independently drawn entries of equal variance 1. In particular, we have:

Theorem B.3 (Based on Theorem 6.1 of Wainwright [2019])**.** *For any positive* δ*, and assuming* $d \gg T - K$*, we have*

$$\sigma_{\min}(\Gamma^*) \;\geq\; \sqrt{d}\,(1 - \delta) - \sqrt{T - K}, \tag{24}$$

with probability at least $1 - e^{-d\delta^2/2}$.

The proof can be found in the reference. Therefore, for any $\delta > 0$,

$$\mathbb{P}\left(\sigma_{\min}(\mathbb{T}) \;\geq\; \mathcal{O}\left(\frac{K\sqrt{d}}{T} \right) \left(1 - \delta - \sqrt{\tfrac{T-K}{d}} \right) \right) \;\geq\; 1 - e^{-d\delta^2/2}. \tag{25}$$

Here, we used the fact that data-generating distributions are product measures with independent entries, such as $\mathcal{N}(\boldsymbol{\mu}, \sigma^2 \boldsymbol{I})$ for any mean vector $\boldsymbol{\mu}$ and variance-per-dimension σ^2. Next, we try to (with high probability) upper-bound the nominator of (16).

High-probability bounds on $\|\boldsymbol{R}\|_2$. What remains is to bound $\|\boldsymbol{R}\|_2$ in (16). From (11) and (12), and using Chernoff bound [Wainwright, 2019], we obtain a naive high-probability bound as follows:

$$\mathbb{P}\Big(|R_i| \;\geq\; c^{1/2}\,\mathcal{O}\big(1+\sqrt{\log T}\big)\Big) \;\leq\; e^{-c}, \quad \forall\, c>0,\; i\in[(K-1)(T-K)]. \tag{26}$$

Therefore, using $(T-K)(K-1)\leq KT$ and assuming $T\gg 1$, we have

$$\mathbb{P}\Big(\|\boldsymbol{R}\|_2 \;\geq\; \mathcal{O}\big(\sqrt{cKT\log T}\big)\Big) \;\leq\; e^{-c}, \quad \forall\, c>0. \tag{27}$$

Combining (25) and (27), and applying the union bound, we obtain the following high-probability bound for $\|\widetilde{\boldsymbol{\delta}}\|_2$ (and hence $\|\boldsymbol{\delta}^*\|_2$), for any $\delta, c>0$:

$$\mathbb{P}\left(\|\widetilde{\boldsymbol{\delta}}\|_2 \;\leq\; \mathcal{O}\left(\frac{T^{3/2}\sqrt{\log T}}{\sqrt{Kd}}\right)\cdot\frac{\sqrt{c}}{1-\delta-\sqrt{(T-K)/d}}\right) \;\geq\; 1-e^{-c}-e^{-d\delta^2/2}. \tag{28}$$

Finally, choosing $c=\log d$ and $\delta=d^{-1/3}$ yields the bound stated in the theorem and completes the proof. □

C Algorithms

The detailed algorithms for the *Full Trajectory Attack* (Algorithm 1), *OSA Attack* (Algorithm 2), and *Trajectory-Free Attack* (Algorithm 3) are provided below. Each follows the same structure with three functions: `select_arm`, `find_perturbation`, and `update`, with differences noted in the comments.

Algorithm 1: Full Trajectory Attack

Function `select_arm`(t, K, $\widetilde{A}$)**:**
- **if** $t \leq K$ **then**
  - **return** t, *False*
- **else**
  - // Adversary selects an arm according to the target trajectory
  - $arm \leftarrow \widetilde{A}_t$
  - **return** arm, *True*

Function `find_perturbation`(K, $\boldsymbol{S}$, $\boldsymbol{G}$, $\boldsymbol{F}$, $\mathcal{C}$, $\{\mu_i\}_{i=1}^K$, *use_NN*)**:**
- // Consider all the inequalities
- **for** $j \leftarrow 1$ ***to*** K, $j \neq arm$ **do**
  - **if** *use_NN* **then**
    - $d_j \leftarrow \boldsymbol{G}^{(arm)} - \boldsymbol{G}^{(j)}$
    - $c_j \leftarrow \sqrt{\frac{2\log t}{N_j}} - \sqrt{\frac{2\log t}{N_{arm}}} + (\boldsymbol{F}^{(j)} - \boldsymbol{F}^{(arm)})$
  - **else**
    - $d_j \leftarrow \boldsymbol{S}^{(arm)} - \boldsymbol{S}^{(j)}$
    - $c_j \leftarrow \sqrt{\frac{2\log t}{N_j}} - \sqrt{\frac{2\log t}{N_{arm}}} - \mu_j^\top d_j$
  - Append (d_j, c_j) to $\mathcal{C}$
- $\mathcal{M} \leftarrow \varnothing$
- **foreach** $(d, c) \in \mathcal{C}$ **do**
  - $\mathcal{M} \leftarrow \mathcal{M} \cup \{\delta^\top d \geq c + 10^{-6}\}$
- $\delta \leftarrow$ solve QP in Eq. (2) with constraints $\mathcal{M}$
- **return** δ

Function `update`($\boldsymbol{X}$, $\boldsymbol{S}^{(arm)}$, $\boldsymbol{F}^{(arm)}$, $\boldsymbol{G}^{(arm)}$, *use_NN*)**:**
- $N_{arm} \leftarrow N_{arm} + 1$
- **if** *use_NN* **then**
  - $\boldsymbol{G}^{(arm)} \leftarrow \boldsymbol{G}^{(arm)} + \frac{1}{N_{arm}}(\nabla NN_\theta(\boldsymbol{X}) - \boldsymbol{G}^{(arm)})$
  - $\boldsymbol{F}^{(arm)} \leftarrow \boldsymbol{F}^{(arm)} + \frac{1}{N_{arm}}(NN_\theta(\boldsymbol{X}) - \boldsymbol{F}^{(arm)})$
- **else**
  - $\boldsymbol{S}^{(arm)} \leftarrow \boldsymbol{S}^{(arm)} + \frac{1}{N_{arm}}(\boldsymbol{X} - \boldsymbol{S}^{(arm)})$

// Inputs: $\{\mathcal{D}_i\}_{i=1}^K$ Logged data, $\widetilde{A}$ is target trajectory, $\{\mu_i\}_{i=1}^K$ are inferred means, use_NN is a boolean flag, NN_θ is the reward model

Input: $K, d, T, \{\mathcal{D}_i\}_{i=1}^K$, `use_NN`, NN_θ, $\{\mu_i\}_{i=1}^K$, $\widetilde{A}$

Output: Perturbation δ

// N: counts, $\boldsymbol{S}$: sample mean, $\boldsymbol{G}$: mean gradient, $\boldsymbol{F}$: mean network output, $\mathcal{C}$: constraint set

Initialize $N, \boldsymbol{S}, \boldsymbol{G}, \boldsymbol{F} \leftarrow 0$

Initialize $\mathcal{C} \leftarrow \varnothing$, $\delta \leftarrow 0$

for $t \leftarrow 1$ ***to*** T **do**
- $arm, do\_attack \leftarrow$ `select_arm`(t, K, $\widetilde{A}$)
- **if** do_attack **then**
  - $\delta \leftarrow$ `find_perturbation`(K, $\boldsymbol{S}$, $\boldsymbol{G}$, $\boldsymbol{F}$, $\mathcal{C}$, $\{\mu_i\}_{i=1}^K$, *use_NN*)
- Pull arm, observe $\boldsymbol{X} \sim \mathcal{D}_{arm}$
- `update`($\boldsymbol{X}$, $\boldsymbol{S}^{(arm)}$, $\boldsymbol{F}^{(arm)}$, $\boldsymbol{G}^{(arm)}$, *use_NN*)

return δ

Algorithm 2: OSA Attack: Online Score-Aware Attack

Function `select_arm`$(t, K, N, \boldsymbol{R})$**:**
- **if** $t \leq K$ **then**
  - **return** t, *False*
- **else**
  - // Select the runner-up arm for attack
  - **for** $j \leftarrow 1$ ***to*** K **do**
    - $\mathrm{UCB}[j] \leftarrow \boldsymbol{R}^{(j)} + \sqrt{\frac{2\log t}{N_j}}$
  - $arm \leftarrow \arg\max_j \mathrm{UCB}[j]$
  - // Attack if the arm with the highest UCB is the optimal arm
  - $do\_attack \leftarrow (arm == 1)$
  - **return** arm, do_attack

Function `find_perturbation`$(\boldsymbol{S}, \boldsymbol{G}, \boldsymbol{F}, \mathcal{C}, \mu_1,$ *use_NN*$)$**:**
- // Construct inequality constraints to prevent selection of optimal arm
- **if** *use_NN* **then**
  - $d \leftarrow \boldsymbol{G}^{(arm)} - \boldsymbol{G}^{(1)}$
  - $c \leftarrow \left(\sqrt{\frac{2\log t}{N_1}} - \sqrt{\frac{2\log t}{N_{arm}}}\right) + (\boldsymbol{F}^{(1)} - \boldsymbol{F}^{(arm)})$
- **else**
  - $d \leftarrow \boldsymbol{S}^{(arm)} - \boldsymbol{S}^{(1)}$
  - $c \leftarrow \left(\sqrt{\frac{2\log t}{N_1}} - \sqrt{\frac{2\log t}{N_{arm}}}\right) - \mu_1^\top d$
- Append (d, c) to $\mathcal{C}$
- $\mathcal{M} \leftarrow \varnothing$
- **foreach** $(d, c) \in \mathcal{C}$ **do**
  - $\mathcal{M} \leftarrow \mathcal{M} \cup \{\delta^\top d \geq c + 10^{-6}\}$
- $\delta \leftarrow$ solve QP in Eq. (2) with constraints $\mathcal{M}$
- **return** δ

Function `update`$(\boldsymbol{X}, \boldsymbol{R}, \boldsymbol{S}^{(arm)}, \boldsymbol{F}^{(arm)}, \boldsymbol{G}^{(arm)},$ *use_NN*$, \delta)$**:**
- $N_{arm} \leftarrow N_{arm} + 1$
- **if** *use_NN* **then**
  - $\boldsymbol{G}^{(arm)} \leftarrow \boldsymbol{G}^{(arm)} + \frac{1}{N_{arm}}(\nabla NN_\theta(\boldsymbol{X}) - \boldsymbol{G}^{(arm)})$
  - $\boldsymbol{F}^{(arm)} \leftarrow \boldsymbol{F}^{(arm)} + \frac{1}{N_{arm}}(NN_\theta(\boldsymbol{X}) - \boldsymbol{F}^{(arm)})$
- **else**
  - $\boldsymbol{S}^{(arm)} \leftarrow \boldsymbol{S}^{(arm)} + \frac{1}{N_{arm}}(\boldsymbol{X} - \boldsymbol{S}^{(arm)})$
- // Update empirical rewards using current δ
- Update R using δ

// Inputs: $\{\mathcal{D}_i\}_{i=1}^K$ Logged data, $\{\mu_i\}_{i=1}^K$ are inferred means, use_NN is a boolean flag, NN_θ is the reward model

Input: $K, d, T, \{\mathcal{D}_i\}_{i=1}^K$, `use_NN`, $NN_\theta, \{\mu_i\}_{i=1}^K$ are inferred means

Output: Perturbation δ

// N: counts, $\boldsymbol{S}$: sample mean, $\boldsymbol{G}$: mean gradient, $\boldsymbol{F}$: mean network output, $\boldsymbol{R}$: empirical rewards, $\mathcal{C}$: constraint set

Initialize $N, \boldsymbol{S}, \boldsymbol{G}, \boldsymbol{F} \leftarrow 0$

Initialize $\mathcal{C} \leftarrow \varnothing$, $\delta \leftarrow 0$

for $t \leftarrow 1$ ***to*** T **do**
- $arm, do\_attack \leftarrow$ `select_arm`$(t, K, N, \boldsymbol{R})$
- **if** do_attack **then**
  - $\delta \leftarrow$ `find_perturbation`$(\boldsymbol{S}, \boldsymbol{G}, \boldsymbol{F}, \mathcal{C}, \mu_1,$ *use_NN*$)$
- Pull arm, observe $\boldsymbol{X} \sim \mathcal{D}_{arm}$
- `update`$(\boldsymbol{X}, \boldsymbol{R}, \boldsymbol{S}^{(arm)}, \boldsymbol{F}^{(arm)}, \boldsymbol{G}^{(arm)},$ *use_NN*$, \delta)$

return δ

Algorithm 3: Trajectory-Free Attack

Function `select_arm`(t, K)**:**
  if $t \leq K$ **then**
    return t, *False*
  else
    // Select a sub-optimal arm in round-robin order
    $arm \leftarrow$ next sub-optimal arm
    return arm, *True*

Function `find_perturbation`$(\boldsymbol{S}, \boldsymbol{G}, \boldsymbol{F}, \mathcal{C}, \mu_1, \textit{use\_NN})$**:**
  // Construct constraints only w.r.t. optimal arm
  if *use_NN* **then**
    $d \leftarrow \boldsymbol{G}^{(arm)} - \boldsymbol{G}^{(1)}$
    $c \leftarrow \left(\sqrt{\frac{2\log t}{N_1}} - \sqrt{\frac{2\log t}{N_{arm}}}\right) + (\boldsymbol{F}^{(1)} - \boldsymbol{F}^{(arm)})$
  else
    $d \leftarrow \boldsymbol{S}^{(arm)} - \boldsymbol{S}^{(1)}$
    $c \leftarrow \left(\sqrt{\frac{2\log t}{N_1}} - \sqrt{\frac{2\log t}{N_{arm}}}\right) - \mu_1^\top d$
  Append (d, c) to $\mathcal{C}$
  $\mathcal{M} \leftarrow \varnothing$
  foreach $(d, c) \in \mathcal{C}$ **do**
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{\delta^\top d \geq c + 10^{-6}\}$
  $\delta \leftarrow$ solve QP in Eq. (2) with constraints $\mathcal{M}$
  return δ

Function `update`$(\boldsymbol{X}, \boldsymbol{S}^{(arm)}, \boldsymbol{F}^{(arm)}, \boldsymbol{G}^{(arm)}, \textit{use\_NN})$**:**
  $N_{arm} \leftarrow N_{arm} + 1$
  if *use_NN* **then**
    $\boldsymbol{G}^{(arm)} \leftarrow \boldsymbol{G}^{(arm)} + \frac{1}{N_{arm}}(\nabla NN_\theta(\boldsymbol{X}) - \boldsymbol{G}^{(arm)})$
    $\boldsymbol{F}^{(arm)} \leftarrow \boldsymbol{F}^{(arm)} + \frac{1}{N_{arm}}(NN_\theta(\boldsymbol{X}) - \boldsymbol{F}^{(arm)})$
  else
    $\boldsymbol{S}^{(arm)} \leftarrow \boldsymbol{S}^{(arm)} + \frac{1}{N_{arm}}(\boldsymbol{X} - \boldsymbol{S}^{(arm)})$

// Inputs: $\{\mathcal{D}_i\}_{i=1}^K$ Logged data, $\{\mu_i\}_{i=1}^K$ are inferred means, use_NN is a boolean flag, NN_θ is the reward model
Input: $K, d, T, \{\mathcal{D}_i\}_{i=1}^K, \{\mu_i\}_{i=1}^K$, `use_NN`
Output: Perturbation δ
// N: counts, $\boldsymbol{S}$: sample mean, $\boldsymbol{G}$: mean gradient, $\boldsymbol{F}$: mean network output, $\mathcal{C}$: constraints
Initialize $N, \boldsymbol{S}, \boldsymbol{G}, \boldsymbol{F} \leftarrow 0$
Initialize $\mathcal{C} \leftarrow \varnothing$, $\delta \leftarrow 0$
for $t \leftarrow 1$ ***to*** T **do**
  $arm, do\_attack \leftarrow$ `select_arm`(t, K)
  if do_attack **then**
    $\delta \leftarrow$ `find_perturbation`$(\boldsymbol{S}, \boldsymbol{G}, \boldsymbol{F}, \mu_1, \mathcal{C}, \textit{use\_NN})$
  Pull arm, observe $\boldsymbol{X} \sim \mathcal{D}_{arm}$
  `update`$(\boldsymbol{X}, \boldsymbol{S}^{(arm)}, \boldsymbol{F}^{(arm)}, \boldsymbol{G}^{(arm)}, \textit{use\_NN})$
return δ

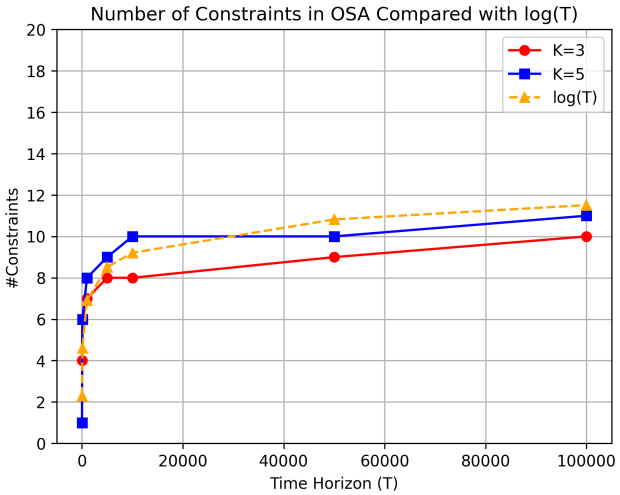


Figure 4: Number of constraints in the OSA *method as a function of the time horizon T. The results show that the number of constraints grows logarithmically with T ($O(\log T)$) for both $K = 3$ and $K = 5$ arms, explaining the high efficiency of the method.*

D Analysis of Constraint Number in the OSA Method

We study the effect of increasing the time horizon T on the number of constraints in the *OSA* method. Our observations show that, while the number of constraints grows with T, this growth follows an $O(\log T)$ pattern, as illustrated in Fig. 4 for T ranging from 10 to 100,000. To confirm this behavior, we repeat the experiment for two values of the number of arms, $K = 3$ and $K = 5$, and observe the same pattern in both cases. This small number of constraints is one of the main reasons that makes the *OSA* method so fast.

E Implementation Details of Reward Model Training

We train several neural networks, each with a single hidden layer whose size varies depending on the task. The networks are optimized using Adam [Adam et al., 2014] and employ the ReLU activation function [Agarap, 2018]. Training is performed using the mean squared error (MSE) loss. The logged data is split into training and validation sets, with 80% used for training and 20% reserved for validation. Networks are trained for 100 epochs with a batch size of 1024 and a learning rate of 10^{-3}. For supervision, the ground truth is computed as the inner product of the data with the mean of the optimal arm.

F Additional Details on ETC and ϵ-greedy Algorithms

F.1 ETC Algorithm

The Explore-Then-Commit (ETC) [Garivier et al., 2016] algorithm is a simple bandit strategy that divides the learning process into two distinct phases. During the exploration phase, the algorithm uniformly samples each arm a predetermined number of times to gather initial reward estimates. Once the exploration phase is complete, ETC commits to the arm with the highest estimated reward for the remainder of the time horizon. This approach provides a balance between exploration and exploitation while remaining easy to implement and analyze. The length of the exploration phase can be tuned to optimize performance depending on the problem setting.

To attack ETC, the adversary only needs to wait until the exploration phase is complete. At the transition between the exploration and commit phases, the adversary manipulates the rewards so that the empirical mean of the target arm exceeds that of the optimal arm. During the subsequent commit phase, the algorithm then selects the target arm. Since the ETC constraint set involves only a single constraint, the attack is fast and straightforward, particularly in high-dimensional settings.

F.2 ϵ-greedy Algorithm

The ϵ-greedy [Dann et al., 2022] algorithm selects a random arm with probability ϵ (exploration) and the arm with the highest estimated reward with probability $1 - \epsilon$ (exploitation) at each time step. In our experiments, we use an adaptive ϵ that decreases over time, allowing the algorithm to explore more during the early stages and gradually exploit the best-known arm as more information is gathered. This time-dependent strategy enables the algorithm to balance exploration and exploitation effectively, and the choice of the initial ϵ and its decay schedule directly influences its performance.

To attack ϵ-greedy, the adversary follows the same principle as *OSA*. During random arm selections, no attack is performed. However, when the algorithm selects the arm with the highest empirical mean, the adversary intervenes to prevent the optimal arm from being pulled. Specifically, the target arm is chosen as the runner-up, and its empirical value is increased to surpass the optimal arm, as modifying the runner-up is simpler than other arms. This approach makes the attack fast and highly efficient.