IS OFFLINE DECISION MAKING POSSIBLE WITH ONLY FEW SAMPLES? RELIABLE DECISIONS IN DATA-STARVED BANDITS VIA TRUST REGION ENHANCEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

What can an agent learn in a stochastic Multi-Armed Bandit (MAB) problem from a dataset that contains just a single sample for each arm? Surprisingly, in this work, we demonstrate that even in such a data-starved setting it may still be possible to find a policy competitive with the optimal one. This paves the way to reliable decision-making in settings where critical decisions must be made by relying only on a handful of samples. Our analysis reveals that *stochastic policies can be substantially better* than deterministic ones for offline decision-making. Focusing on offline multi-armed bandits, we design an algorithm called Trust Region of Uncertainty for Stochastic policy enhancemenT (TRUST) which is quite different from the predominant value-based lower confidence bound approach. Its design is enabled by localization laws, critical radii, and relative pessimism. We prove that its sample complexity is comparable to that of LCB on minimax problems while being substantially lower on problems with very few samples. Finally, we consider an application to offline reinforcement learning in the special case where the logging policies are known.

1 INTRODUCTION

In several important problems, critical decisions must be made with just very few samples of pre-collected experience. For example, collecting samples in robotic manipulation may be slow and costly, and the ability to learn from very few interactions is highly desirable (Hester & Stone, 2013; Liu et al., 2021). Likewise, in clinical trials and in personalized medical decisions, reliable decisions must be made by relying on very small datasets (Liu et al., 2017). Sample efficiency is also key in personalized education (Bassen et al., 2020; Ruan et al., 2023). However, to achieve good performance, the state-of-the-art algorithms may require millions of samples (Fu et al., 2020). These empirical findings seem to be supported by the existing theories: the sample complexity bounds, even minimax optimal ones, can be large in practice due to the large constants and the warmup factors (Ménard et al., 2021; Li et al., 2022; Azar et al., 2017; Zanette et al., 2019).

In this work, we study whether it is possible to make reliable decisions with only a few samples. We focus on an offline Multi-Armed Bandit (MAB) problem, which is a foundation model for decision-making (Lattimore & Szepesvári, 2020). In online MAB, an agent repeatedly chooses an arm from a set of arms, each providing a stochastic reward. Offline MAB is a variant where the agent cannot interact with the environment to gather new information and instead, it must make decisions based on a pre-collected dataset without playing additional exploratory actions, aiming at identifying the arm with the highest expected reward (Audibert et al., 2010; Garivier & Kaufmann, 2016; Russo, 2016; Ameko et al., 2020).

The standard approach to the problem is the Lower Confidence Bound (LCB) algorithm (Rashidinejad et al., 2021), a pessimistic variant of UCB (Auer et al., 2002) that involves selecting the arm with the highest lower bound on its performance. LCB encodes a principle called *pessimism under uncertainty*, which is the foundation principle for most algorithms for offline bandits and reinforcement learning (RL) (Jin et al., 2020; Zanette et al., 2020; Xie et al., 2021; Yin & Wang, 2021; Kumar et al.,

2020; Kostrikov et al., 2021). Unfortunately, the available methods that implement the principle of pessimism under uncertainty can fail in a data-starved regime because they rely on confidence intervals that are too loose when just a few samples are available. For example, even on a simple MAB instance with ten thousand arms, the best-known (Rashidinejad et al., 2021) performance bound for the LCB algorithm requires 24 samples per arm in order to provide meaningful guarantees, see Section 2. In more complex situations, such as in the sequential setting with function approximation, such a problem can become more severe due to the higher metric entropy of the function approximation class and the compounding of errors through time steps.

These considerations suggest that there is a "barrier of entry" to decision-making, both theoretically and practically: one needs to have a substantial number of samples in order to make reliable decisions even for settings as simple as offline MAB where the guarantees are tighter. Given the above technical reasons, and the lack of good algorithms and guarantees for data-starved decision problems, it is unclear whether it is even possible to find good decision rules with just a handful of samples.

In this paper, we make a substantial contribution towards lowering such barriers of entry. We discover that a carefully-designed algorithm tied to an advanced statistical analysis can substantially improve the sample complexity, both theoretically and practically, and enable reliable decision-making with just a handful of samples. More precisely, we focus on the offline MAB setting where we show that even if the dataset contains just a *single sample* in every arm, it may still be possible to compete with the optimal policy. This is remarkable, because with just one sample per arm—for example from a Bernoulli distribution—it is impossible to estimate the expected payoff of any of the arms! Our discovery is enabled by several key insights:

- We search over *stochastic* policies, which can yield better performance for offline-decision making;
- We use a *localized* notion of metric entropy to carefully control the size of the stochastic policy class that we search over;
- We implement a concept called *relative pessimism* to obtain sharper guarantees.

These considerations lead us to design a trust region policy optimization algorithm called Trust Region of Uncertainty for Stochastic policy enhancemenT (TRUST), one that offers superior theoretical as well as empirical performance compared to LCB in a data-scarce situation.

Moreover, we apply the algorithm to selected reinforcement learning problems from (Fu et al., 2020) in the special case where information about the logging policies is available. We do so by a simple reduction from reinforcement learning to bandits, by mapping policies and returns in the former to actions and rewards in the latter, thereby disregarding the sequential aspect of the problem. Although we rely on the information of the logging policies being available, the empirical study shows that our algorithm compares well with a strong deep reinforcement learning baseline (i.e, CQL from (Kumar et al., 2020)), without being sensitive to partial observability, sparse rewards, and hyper-parameters.

2 Data-Starved Multi-Armed Bandits

In this section, we describe the MAB setting and give an example of a "data-starved" MAB instance where prior methods (such as LCB) can fail. We informally say that an offline MAB is "data-starved" if its dataset contains only very few samples in each arm.

Notation. We let $[n] = \{1, 2, ..., n\}$ for a positive integer n. We let $\|\cdot\|_2$ denote the Euclidean norm for vectors and the operator norm for matrices. We hide constants and logarithmic factors in the $\widetilde{O}(\cdot)$ notation. We let $\mathbb{B}_p^d(s) = \{x \in \mathbb{R}^d : \|x\|_p \leq s\}$ for any $s \geq 0$ and $p \geq 1$. $a \lesssim b$ ($a \gtrsim b$) means $a \leq Cb$ ($a \geq Cb$) for some numerical constant C. $a \simeq b$ means that both $a \lesssim b$ and $b \lesssim a$ hold.

Multi-armed bandits. We consider the case where there are d arms in a set $\mathcal{A} = \{a_1, ..., a_d\}$ with expected reward $r(a_i), i \in [d]$. We assume access to an offline dataset $\mathcal{D} = \{(x_i, r_i)\}_{i \in [N]}$ of action-reward tuples, where the experienced actions $\{x_i\}_{i \in [N]}$ are i.i.d. from a distribution μ. Each experienced reward is a random variable with expectation $\mathbb{E}[r_i] = r(x_i)$ and independent Gaussian noises $\zeta_i := r(x_i) - \mathbb{E}[r_i]$. For $i \in [d]$, we denote the number of pulls to arm a_i in $\mathcal{D}$ by $N(a_i)$ or N_i, while the variance of the noise for arm a_i is denoted by σ_i^2. We denote the optimal arm

as $a^* \in \arg\max_{a \in \mathcal{A}}[r(a)]$ and the single policy concentrability as $C^* = 1/\mu(a^*)$ where μ is the distribution that generated the dataset. Without loss of generality, we assume the optimal arm is unique. We also write $r = (r_1, r_2, ..., r_d)^\top$. Without loss of generality, we assume there is at least one sample for each arm (such arm can otherwise be removed).

Lower confidence bound algorithm. One simple but effective method for the offline MAB problem is the Lower Confidence Bound (LCB) algorithm, which is inspired by its online counterpart (UCB) (Auer et al., 2002). Like UCB, LCB computes the empirical mean $\widehat{r}_i$ associated to the reward of each arm i along with its half confidence width b_i. They are defined as

$$\widehat{r}_i := \frac{1}{N(a_i)} \sum_{k:x_k=a_i} x_k,\ b_i := \sqrt{\frac{2\sigma_i^2}{N(a_i)} \log\left(\frac{2d}{\delta}\right)}. \tag{1}$$

This definition ensures that each confidence interval brackets the corresponding expected reward with probability $1-\delta$:

$$\widehat{r}_i - b_i \leq r\left(a_i\right) \leq \widehat{r}_i + b_i \quad \forall i \in [d]. \tag{2}$$

The width of the confidence level depends on the noise level σ_i, which can be exploited by variance-aware methods (Zhang et al., 2021; Min et al., 2021; Yin et al., 2022; Dai et al., 2022). When the true noise level is not accessible, we can replace it with the empirical standard deviation or with a high-probability upper bound. For example, when the reward for each arm is restricted to be within $[0, 1]$, a simpler upper bound is $\sigma_i^2 \leq 1/4$.

Unlike UCB, the half-width of the confidence intervals for LCB is not added, but subtracted, from the empirical mean, resulting in the lower bound $l_i = \widehat{r}_i - b_i$. The action identified by LCB is then the one that maximizes the resulting lower bound, thereby incorporating the principle of pessimism under uncertainty (Jin et al., 2020; Kumar et al., 2020). Specifically, given the dataset $\mathcal{D}$, LCB selects the arm using the following rule:

$$\widehat{a}_{\mathsf{LCB}} := \arg\max_{a_i \in \mathcal{A}}\ l_i, \tag{3}$$

(Rashidinejad et al., 2021) analyzed the LCB strategy. Below we provide a modified version of their theorem.

Theorem 2.1 (LCB Performance)**.** *Suppose the noise of arm a_i is sub-Gaussian with proxy variance σ_i^2. Let $\delta \in (0, 1/2)$. Then, we have*

1. *(Comparison with any arm) With probability at least $1-\delta$, for any comparator policy $a_i \in \mathcal{A}$, it holds that $r(a_i) - r(\widehat{a}_{\mathsf{LCB}}) \leq \sqrt{8\sigma_i^2 \log(2d/\delta)/N(a_i)}$.*
2. *(Comparison with the optimal arm) Assume $\sigma_i = 1$ for any $i \in [d]$ and $N \geq 8C^* \log(1/\delta)$. Then, with probability at least $1-2\delta$, one has $r(a^*) - r(\widehat{a}_{\mathsf{LCB}}) \leq \sqrt{4C^* \log(2d/\delta)/N}$.*

The statement of this theorem is slightly different from that in (Rashidinejad et al., 2021), in the sense that their suboptimality is over $\mathbb{E}_{\mathcal{D}}[r\left(a^*\right) - r\left(\widehat{a}_{\mathsf{LCB}}\right)]$ instead of a high-probability one. (Rashidinejad et al., 2021) proved the minimax optimality of the algorithm when the single policy concentrability $C^* \geq 2$ and the sample size $N \geq \widetilde{O}(C^*)$.

A data-starved MAB problem and failure of LCB. In order to highlight the limitation of a strategy such as LCB, let us describe a specific data-starved MAB instance, specifically one with $d = 10000$ arms, equally partitioned into a set of *good arms* (i.e., $\mathcal{A}_g$) and a set of *bad arms* (i.e., $\mathcal{A}_b$). Each good arm returns a reward following the uniform distribution over $[0.5, 1.5]$, while each bad arm returns a reward which follows $\mathcal{N}(0, 1/4)$.

Assume that we are given a dataset that contains *only one sample per each arm*. Instantiating the LCB confidence interval in equation 2 with $\sigma_i \leq 1/2$ and $\delta = 0.1$, one obtains $\widehat{r}_i - 2.5 \leq r(a_i) \leq \widehat{r}_i + 2.5$. Such bound is uninformative, because the lower bound for the true reward mean is less than the reward value of the worst arm. The performance bound for LCB confirms this intuition, because Theorem 2.1 requires at least $N(a_i) \geq \lceil 8 * \log(1/0.05) \rceil = 24$ samples in each arm to provide any guarantee with probability at least 0.9 (here $C^* = d$).

Can stochastic policies help? At a first glance, extracting a good decision-making strategy for the problem discussed in Section 2 seems like a hopeless endeavor, because it is information-theoretically impossible to reliably estimate the expected payoff of any of the arms with just a single sample on each. In order to proceed, the key idea is to enlarge the search space to contain *stochastic policies*.

Definition 2.2 (Stochastic Policies)**.** A stochastic policy over a MAB is a probability distribution $w \in \mathbb{R}^d, w_i \geq 0, \sum_{i=1}^d w_i = 1$.

To exemplify how stochastic policies can help, consider the *behavioral cloning* policy, which mimics the policy that generated the dataset for the offline MAB in Section 2. Such policy is stochastic, and it plays all arms uniformly at random, thereby achieving a score around 0.5 with high probability. The value of the behavioral cloning policy can be readily estimated using the Hoeffding bound (e.g., Proposition 2.5 in (Wainwright, 2019)): with probability at least $1 - \delta = 0.9$, (here $d = 10000$ is the number of arms and $\sigma = 1/2$ is the true standard deviation), the value of behavioral cloning policy is greater or equal than $1/2 - \sqrt{2\sigma^2 \log(2/\delta)/d} \approx 0.488$. Such value is higher than the one guaranteed for LCB by Theorem 2.1. Intuitively, a stochastic policy that selects multiple arms can be evaluated more accurately because it averages the rewards experienced over different arms. This consideration suggests optimizing over stochastic policies.

By optimizing a lower bound on the performance of the stochastic policies, it should be possible to find one with a provably high return. Such an idea leads to solving an offline *linear bandit* problem, as follows

$$\max_{w \in \mathbb{R}^d, w_i \geq 0, \sum_{i=1}^d w_i = 1} \sum_{i=1}^d w_i \widehat{r}_i - c(w) \tag{4}$$

where $c(w)$ is a suitable confidence interval for the policy w and $\widehat{r}_i$ is the empirical reward for the i-th arm defined in equation 1. While this approach is appealing, enlarging the search space to include all stochastic policies brings an increase in the metric entropy of the function class, and concretely, a $\sqrt{d}$ factor (Abbasi-Yadkori et al., 2011; Rusmevichientong & Tsitsiklis, 2010; Hazan & Karnin, 2016; Jun et al., 2017; Kim et al., 2022) in the confidence intervals $c(w)$ (in equation 4), which negates all gains that arise from considering stochastic policies. In the next section, we propose an algorithm that bypasses the need for such $\sqrt{d}$ factor by relying on a more careful analysis and optimization procedure.

3 Trust Region of Uncertainty for Stochastic policy enhancemenT (TRUST)

In this section, we introduce our algorithm, called Trust Region of Uncertainty for Stochastic policy enhancemenT (TRUST). At a high level, the algorithm is a policy optimization algorithm based on a trust region centered around a reference policy. The size of the trust region determines the degree of pessimism, and its optimal problem-dependent size can be determined by analyzing the supremum of a problem-dependent empirical process. In the sequel, we describe 1) the decision variables, 2) the trust region optimization program, and 3) some techniques for its practical implementation.

3.1 Decision variables

The algorithm searches over the class of stochastic policies given by the weight vector $w = (w_1, w_2, ..., w_d)^\top$ of Definition 2.2. Instead of directly optimizing over the weights of the stochastic policy, it is convenient to center w around a *reference stochastic policy* $\widehat{\mu}$ which is either known to perform well or is easy to estimate. In our theory and experiments, we consider a simple setup and use the behavioral cloning policy weighted by the noise levels $\{\sigma_i\}$ if they are known. Namely, we consider

$$\widehat{\mu}_i = \frac{N_i/\sigma_i^2}{\sum_{j=1}^d N_j/\sigma_j^2} \quad \forall i \in [d]. \tag{5}$$

When the size of the noise σ_i is constant across all arms, the policy $\widehat{\mu}$ is the behavioral cloning policy; when σ_i differs across arms, $\widehat{\mu}$ minimizes the variance of the empirical reward $\widehat{\mu} = \arg\min_{w \in \mathbb{R}^d, w_i \geq 0, \sum_i w_i = 1} \mathrm{Var}\left(w^\top \cdot \widehat{r}\right)$, where $\widehat{r} = (\widehat{r}_1, ..., \widehat{r}_d)^\top$ is defined in equation 1. Using such definition, we define as *decision variable* the *policy improvement* vector $\Delta := w - \widehat{\mu}$. This preparatory step is key: it allows us to implement **relative pessimism**, namely pessimism on the improvement—represented by Δ—rather than on the absolute value of the policy w. Moreover, by restricting the search space to a ball around $\widehat{\mu}$, one can efficiently reduce the metric entropy of the policy class and obtain tighter confidence intervals.

3.2 TRUST REGION OPTIMIZATION

Trust region. TRUST (Algorithm 1) returns the stochastic policy $\pi_{TRUST} = \widehat{\Delta} + \widehat{\mu} \in \mathbb{R}^d$, where $\widehat{\mu}$ is the reference policy defined in equation 5 and $\widehat{\Delta}$ is the policy improvement vector. In order to accurately quantify the effect of the improvement vector Δ, we constrain it to a trust region $\mathsf{C}(\varepsilon)$ centered around $\widehat{\mu}$ where $\varepsilon > 0$ is the radius of the trust region. More concretely, for a given radius $\varepsilon > 0$, the trust region is defined as

$$\mathsf{C}(\varepsilon) := \Bigg\{ \Delta : \Delta_i + \widehat{\mu}_i \geq 0, \quad \|\Delta + \widehat{\mu}\|_1 = 1, \quad \sum_{i=1}^{d} \Delta_i^2 \frac{\sigma_i^2}{N_i} \leq \varepsilon^2 \Bigg\}. \tag{6}$$

The trust region above serves two purposes: it ensures that the policy $\widehat{\Delta} + \widehat{\mu}$ still represents a valid stochastic policy, and it regularizes the policy around the reference policy $\widehat{\mu}$. We then search for the best policy within $\mathsf{C}(\varepsilon)$ by solving the optimization program

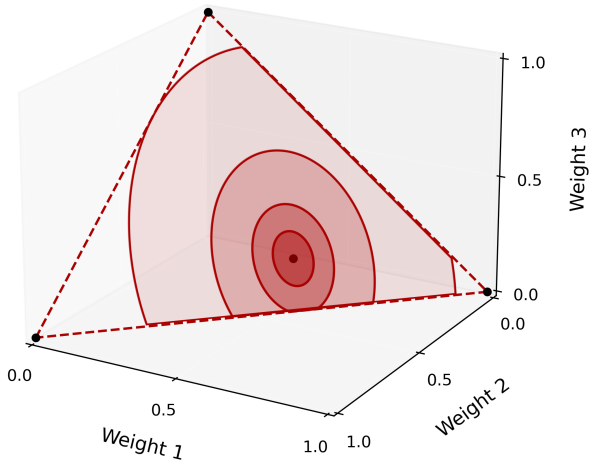

Figure 1: A simple diagram for the trust regions on a 3-dim simplex. The central point is the reference (stochastic) policy, while red ellipses are trust regions around this reference policy.

$$\widehat{\Delta}_\varepsilon := \underset{\Delta \in \mathsf{C}(\varepsilon)}{\arg\max}\, \Delta^\top \widehat{r}. \tag{7}$$

Computationally, the program equation 7 is a second-order cone program (Alizadeh & Goldfarb, 2003; Boyd & Vandenberghe, 2004), which can be solved efficiently with standard off-the shelf libraries (Diamond & Boyd, 2016).

When $\varepsilon = 0$, the trust region only includes the vector $\Delta = 0$, and the reference policy $\widehat{\mu}$ is the only feasible solution. When $\varepsilon \to \infty$, the search space includes all stochastic policies. In this latter case, the solution identified by the algorithm coincides with the greedy algorithm which chooses the arm with the highest empirical return. Rather than leaving ε as a hyper-parameter, in the following we highlight a selection strategy for ε based on localized Gaussian complexities.

Critical radius. The choice of ε is crucial to the performance of our algorithm because it balances optimization with regularization. Such consideration suggests that there is an optimal choice for the radius ε which balances searching over a larger space with keeping the metric entropy of such space under control. The optimal problem-dependent choice $\widehat{\varepsilon}_*$ can be found as a solution of a certain equation involving a problem-dependent *supremum of an empirical process*. More concretely, let E be the feasible set of ε (e.g., $E = \mathbb{R}^+$). We define the critical radius as

Definition 3.1 (Critical Radius)**.** The critical radius $\widehat{\varepsilon}_*$ of the trust region is the solution to the program

$$\widehat{\varepsilon}_* = \underset{\varepsilon \in E}{\arg\max} \left[\widehat{\Delta}_\varepsilon^\top \cdot \widehat{r} - \mathcal{G}(\varepsilon) \right]. \tag{8}$$

Such equation involves a quantile of the localized gaussian complexity $\mathcal{G}(\varepsilon)$ of the stochastic policies identified by the trust region. Mathematically, this is defined as

Definition 3.2 (Quantile of the supremum of Gaussian process)**.** We denote the noise vector as $\eta = \widehat{r} - r$, which by our assumption is coordinate-wise independent and satisfies $\eta_i \sim \mathcal{N}\left(0, \sigma_i^2 / N(a_i)\right)$. We define $\mathcal{G}(\varepsilon)$ as the smallest quantity such that with probability at least $1 - \delta$, for any $\varepsilon \in E$, it holds that $\sup_{\Delta \in \mathsf{C}(\varepsilon)} \Delta^\top \eta \leq \mathcal{G}(\varepsilon)$.

In essence, $\mathcal{G}(\varepsilon)$ is an upper quantile of the supremum of the Gaussian process $\sup_{\Delta \in \mathsf{C}(\varepsilon)} \Delta^\top \eta$ which holds uniformly for every $\varepsilon \in E$. We also remark that this quantity depends on the feasible set E and the trust region $\mathsf{C}(\varepsilon)$, and hence, is highly problem-dependent.

The critical radius plays a crucial role: it is the radius of the trust region that *optimally* balances optimization with uncertainty. Enlarging ε enlarges the search space for Δ, enabling the discovery of policies with potentially higher return. However, this also brings an increase in the metric entropy of the policy class encoded by $\mathcal{G}(\varepsilon)$, which means that each policy can be estimated less accurately. The critical radius represents the optimal tradeoff between these two forces. The final improvement vector that TRUST returns, which we denote as $\widehat{\Delta}_*$, is determined by solving equation 7 with the critical radius $\widehat{\varepsilon}_*$ defined in equation 8. In mathematical terms, we express this as

$$\widehat{\Delta}_* := \underset{\Delta \in \mathsf{C}(\widehat{\varepsilon}_*)}{\arg\max}\, \Delta^\top \widehat{r}. \tag{9}$$

Implementation details. Since it can be difficult to solve equation 8 for a continuous value of $\varepsilon \in E = \mathbb{R}^+$, we use a discretization argument by considering the following candidate subset:

$$E = \left\{ \varepsilon_0, \frac{\varepsilon_0}{\alpha}, ..., \frac{\varepsilon_0}{\alpha^{|E|-1}} \right\}, \tag{10}$$

where $\alpha > 1$ is the decaying rate and ε_0 is the largest possible radius, which is the maximal weighted distance from the reference policy to any vertex. Mathematically, this is defined as $\varepsilon_0 = \max_{i \in [d]} \sqrt{\sum_{j \neq i} \widehat{\mu}_j^2 \sigma_j^2 / N_j + (1 - \widehat{\mu}_i)^2 \sigma_i^2 / N_i}$. Our analysis that leads to Theorem 4.1 takes into account such discretization argument.

In line 2 of Algorithm 1, the algorithm works by estimating the quantile of the supremum of the localized Gaussian complexity $\mathcal{G}(\varepsilon)$ that appears in Definition 3.2, and then choose the ε that maximizes the objective function in equation 8. Although $\mathcal{G}(\varepsilon)$ can be upper bounded analytically, in our experiments we aim to obtain tighter guarantees and so we estimate it via Monte-Carlo. This can be achieved by 1) sampling independent noise vectors η, 2) solving $\sup_{\Delta \in \mathsf{C}(\varepsilon)} \Delta^\top \eta$ and 3) estimating the quantile via order statistics. More details can be found in Appendix D.

In summary, our practical algorithm can be seen as solving the optimization problem

$$(\widehat{\varepsilon}_*, \widehat{\Delta}_*) = \underset{\varepsilon \in E, \Delta \in \mathsf{C}(\varepsilon)}{\arg\max} \left\{ \Delta^\top \widehat{r} - \widehat{\mathcal{G}}(\varepsilon) \right\}$$

where $\widehat{r} \in \mathbb{R}^d$ is the empirical reward vector with $\widehat{r}_i$ defined in equation 1. Here, $\widehat{\mathcal{G}}(\varepsilon)$ is computed according to the Monte-Carlo method defined in Algorithm 2 in Appendix D and E is the candidate subset for radius defined in equation 10. This indicates a balance between the empirical reward of a stochastic policy and the local entropy metric it induces.

Algorithm 1 Trust Region of Uncertainty for Stochastic policy enhancemenT (TRUST)

Input: Offline dataset $\mathcal{D}$, failure probability δ, the candidate set for the trust region widths E (in practice, this is chosen as equation 10).

1. For $\varepsilon \in E$, compute $\widehat{\Delta}_\varepsilon$ from equation 7.
2. For $\varepsilon \in E$, estimate $\mathcal{G}(\varepsilon)$ via Monte-Carlo method (see Algorithm 2 in Appendix D).
3. Solve equation 8 to obtain the critical radius $\widehat{\varepsilon}_*$.
4. Compute the optimal improvement vector in $\mathsf{C}(\widehat{\varepsilon}_*)$ via equation 9, denoted as $\widehat{\Delta}_*$.
5. Return the stochastic policy $\pi_{TRUST} = \widehat{\mu} + \widehat{\Delta}_*$.

4 THEORETICAL GUARANTEES

Problem-dependent analysis In this section, we provide some theoretical guarantees for the policy π_{TRUST} returned by TRUST. We present 1) an improvement over the reference policy $\widehat{\mu}$, 2) a sub-optimality gap with respect to any comparator policy π and 3) an actionable lower bound on the performance of the output policy. Given a stochastic policy π, we let $V^\pi = \mathbb{E}_{a \sim \pi}[r(a)]$ denote its value function. Furthermore, we denote a comparator policy π by a triple $(\varepsilon, \Delta, \pi)$ such that $\varepsilon > 0, \Delta \in \mathsf{C}(\varepsilon), \pi = \widehat{\mu} + \Delta$.

Theorem 4.1 (Main theorem)**.** *TRUST has the following properties.*

1. *With probability at least* $1-\delta$*, the improvement over the behavioral policy is at least*

$$V^{\pi_{TRUST}} - V^{\widehat{\mu}} \geq \sup_{\varepsilon \leq \varepsilon_0, \Delta \in \mathsf{C}(\varepsilon)} \left[\Delta^\top r - 2\mathcal{G}\left(\lceil \varepsilon \rceil\right)\right], \quad \textit{where} \quad \lceil \varepsilon \rceil = \inf\{\varepsilon' \in E, \varepsilon' \geq \varepsilon\}. \tag{11}$$

2. *With probability at least* $1-\delta$*, for any stochastic comparator policy* $(\varepsilon, \Delta, \pi)$*, the sub-optimality of the output policy can be upper bounded as*

$$V^{\pi} - V^{\pi_{TRUST}} \leq 2\mathcal{G}\left(\lceil \varepsilon \rceil\right). \tag{12}$$

3. *With probability at least* $1-2\delta$*, the data-dependent lower bound on* $V^{\pi_{TRUST}}$ *satisfies*

$$V^{\pi_{TRUST}} \geq \pi_{TRUST}^\top \widehat{r} - \mathcal{G}\left(\lceil \widehat{\varepsilon}_* \rceil\right) - \sqrt{\frac{2\log(1/\delta)}{\sum_{j=1}^d N_j/\sigma_j^2}}, \tag{13}$$

where $\pi_{TRUST} = \widehat{\mu} + \widehat{\Delta}_*$ *is the policy output by Algorithm 1.*

The proof of Theorem 4.1 is deferred to Appendix B. A fine-grained analysis for the suboptimality is contained in Appendix E. Our guarantees are problem-dependent as a function of the Gaussian process $\mathcal{G}(\cdot)$; in Section 5 we show how these can be instantiated on an actual problem, highlighting the tightness of the analysis. Equation (11) highlights the improvement with respect to the behavioral policy. It is expressed as a trade-off between maximizing the improvement $\Delta^\top r$ and minimizing its uncertainty $\mathcal{G}\left(\lceil \varepsilon \rceil\right)$. The presence of the $\sup_\varepsilon$ indicates that TRUST achieves an *optimal* balance between these two factors. The state of the art guarantees that we are aware of highlight a trade-off between value and variance (Jin et al., 2021; Min et al., 2021). The novelty of our result lies in the fact that TRUST optimally balances the uncertainty implicitly as a function of the 'coverage' as well as the metric entropy of the search space. That is, TRUST selects the most appropriate search space by trading off its metric entropy with the quality of the policies that it contains. The right-hand side in Equation (13) gives actionable statistical guarantees on the quality of the final policy and it can be fully computed from the available dataset; we give an example of the tightness of the analysis in Section 5.

Localized Gaussian complexity $\mathcal{G}(\varepsilon)$**.** In Theorem 4.1, we upper bound the suboptimality $V^\pi - V^{\pi_{TRUST}}$ via a notion of localized metric entropy $\mathcal{G}(\cdot)$. It is the quantile of the supremum of a Gaussian process, which can hardly be calculated analytically but can be efficiently estimated via Monte Carlo method (which does not collect additional samples, e.g., see Appendix D). It can also be concentrated around its expectation, which is also *localized Gaussian width*, a concept well-established in statistical learning theory (Bellec, 2019; Wei et al., 2020; Wainwright, 2019). More concretely, this is the localized Gaussian width for an affine simplex: $\mathbb{E}[\sup_{\Delta \in \mathsf{C}(\varepsilon)} \Delta^\top \eta] = \mathbb{E}[\sup_{\mathbb{S}^{d-1} \cap \{\Delta : \|\Delta\|_\Sigma \leq \varepsilon\}} \Delta^\top \eta]$, where $\mathbb{S}^{d-1}$ denotes the simplex in $\mathbb{R}^d$ and $\Sigma := \mathrm{diag}\left(\frac{\sigma_1^2}{N_1}, \frac{\sigma_2^2}{N_2}, ..., \frac{\sigma_d^2}{N_d}\right)$ is the weighted matrix. Moreover, this localized Gaussian width can be upper bound via

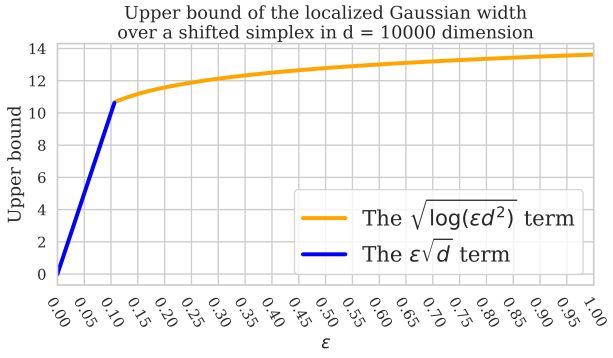

Figure 2: The upper bound for the localized Gaussian width over a shifted simplex on $d = 10000$ dimension. The shifted simplex is $\{\Delta \in \mathbb{R}^d : \sum_{i=1}^d \Delta_i = 0\}$. The two-staged upper bound is based on Theorem 1 in (Bellec, 2019)

$$\mathbb{E}\left[\sup_{\Delta \in \mathsf{C}(\varepsilon)} \Delta^\top \eta\right] \lesssim \min\left\{\sqrt{\log\left(d\varepsilon^2\right)}, \varepsilon\sqrt{d}\right\}. \tag{14}$$

To make it clearer, we plot this upper bound for localized Gaussian width in Figure 2. In equation 14, the rate matches the minimax lower bound up to universal constant (Gordon et al., 2007; Lecué & Mendelson, 2013; Bellec, 2019). To see the implication of the upper bound equation 14, let's consider a simple example where the logging policy is uniform over all arms. We denote the optimal arm as a^* and define $C^* := 1/\mu(a^*)$ as the concentrability coefficient. By applying equation 14 and some concentration techniques (see Wainwright, 2019), we can perform a fine-grained analysis for the suboptimality induced by π_{TRUST}. Specifically, with probability at least $1-\delta$, one has

$$V^{\pi_*} - V^{\pi_{TRUST}} \lesssim \sqrt{C^* \log(2d|E|/\delta)/N}. \tag{15}$$

Note that, the high-probability upper bound here is minimax optimal up to constant and logarithmic factor (Rashidinejad et al., 2021) when $C^* \geq 2$. Moreover, this example of uniform logging policy is an instance where LCB achieves minimax sub-optimality (up to constant and log factors) (see the proof of Theorem 2 in Rashidinejad et al., 2021). In this case, TRUST will achieve the same level of guarantees for the suboptimality of the output policy. We also empirically show the effectiveness of TRUST in Section 5. The full theorem for a fine-grained analysis for the suboptimality and its proof are deferred to Appendix E.

Augmentation with LCB. Compared to classical LCB, Algorithm 1 considers a much larger searching space, which encompasses not only the vertices of the simplex but the inner points as well. This enlargement of searching space shows great advantage, but this also comes with the price of larger uncertainty, especially when the width ε is large. In LCB, one considers the uncertainty by upper bound the noise at each vertex uniformly, while in our case, the uniform upper bound for a sub-region of the shifted simplex must be considered. When ε is large, the trust region method will induce larger uncertainty and tend to select a more stochastic policy than LCB and hence, can achieve worse performance. Moreover, when each arm has sufficiently many data samples to roughly estimate its mean return to reasonable accuracy, LCB works well because it chooses the arm with a tight lower bound. However the current results for LCB do not cover the important case where only few samples (e.g., less than 24 as described in Section 2) are available. Encouragingly our work shows strong results in such settings. To determine the most effective final policy, one can always combine TRUST (Algorithm 1) with LCB and select the better one between them based on the lower bound induced by two algorithms. By comparing the lower bounds of LCB and TRUST, the value of the finally output policy is guaranteed to outperform the lower bound for either LCB or TRUST with high probability. We defer the detailed algorithm and its theoretical guarantees to Appendix G.

5 Experiments

We present simulated experiments where we show the failure of LCB and the strong performance of TRUST. Moreover, we also present an application of TRUST to offline reinforcement learning.

Simulated experiments: A data-starved MAB. We consider a data-starved MAB problem with $d = 10000$ arms denoted by $a_i, i \in [d]$. The reward distributions are

$$r(a_i) \sim \mathsf{Uniform}(0.5, 1.5) \text{ for } i \leq 5000; \quad r(a_i) \sim \mathcal{N}(0, 1/4) \text{ for } i > 5000. \tag{16}$$

Namely, the set of good arms have reward random variables from a uniform distribution over $[0.5, 1.5]$ with unit mean, while the bad arms return a Gaussian reward with zero mean. We consider a dataset that contains a single sample for each of these arms.

We test Algorithm 1 on this MAB instance with fixed variance level $\sigma_i = 1/2$. We set the reference policy $\widehat{\mu}$ to be the behavioral cloning policy, which coincides with the uniform policy. We also test LCB and the greedy method which simply chooses the policy with the highest empirical reward.

In this example, the greedy algorithm fails because it erroneously selects an arm with a reward > 1.5, but such reward can only originate from a normal distribution with mean zero. Despite LCB incorporates the principle of pessimism under uncertainty, it selects an arm with average return equal to zero; its performance lower bound given by the confidence intervals is -1.5, which is almost vacuous and very uninformative. The behavioral cloning policy performs better, because it selects an arm uniformly at random, achieving the score 0.5.

Behavior Policy	Greedy	LCB	LCB Lower Bound	Improvement by TRUST	TRUST	TRUST Lower Bound
0.5	0	0	-1.5	0.42	0.92	0.6

Table 1: Results of simulated experiments in a 10000-arm bandit. The reward distribution is described in equation 16. The offline dataset includes one sample for each arm. The greedy method chooses the arm with the highest empirical reward. LCB selects an arm based on equation 3. The lower bound for LCB and TRUST follow equation 2 and equation 13, respectively.

Algorithm 1 achieves the best performance: the value of the policy that it identifies is 0.92, which *almost matches the optimal policy*. The lower bound on its performance computed by instantiating the RHS in equation 13 is around 0.6, a guarantee much tighter than that for LCB.

In order to gain intuition on the learning mechanics of TRUST, in Figure 3 we progressively enlarge the radius of the trust region from zero to the largest possible radius (on the x axis) and plot the value of the policy that maximizes the linear objective $\Delta^\top \widehat{r},\ \Delta \in \mathsf{C}(\varepsilon)$ for each value of the radius ε. Note that we rescale the range of ε to make the largest possible ε be one. In the same figure we also plot the lower bound computed with the help of equation equation 13.

Initially, the value of the policy increases because the optimization in equation 7 is performed over a larger set of stochastic policies. However, when ε approaches the maximal possible radius, all stochastic policies are included in the optimization program. In this case, TRUST greedily selects the arm with the highest empirical reward, which is from a normal distribution with a mean zero. The optimal balance between the size of the policy search space and its metric entropy is given by the critical radius $\varepsilon = 0.0116\varepsilon_0$, which is the point where the lower bound is the highest.

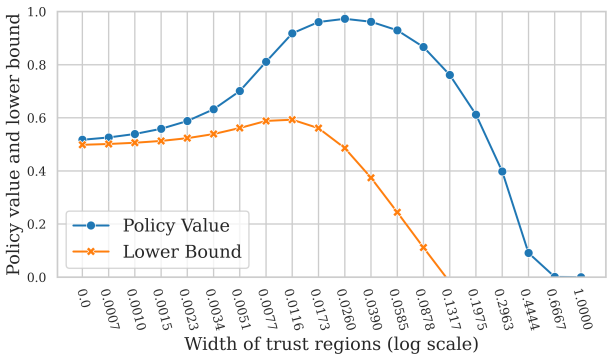

Figure 3: Policy values and their lower bounds for a data-starved MAB instance with 10000 arms whose reward distribution is described in equation 16.

A more general data-starved MAB. Besides the data-starved MAB we constructed, we also show that in general MABs, the performance of TRUST is on par with LCB, but TRUST will have a much tighter statistical guarantee, i.e., a larger lower bound for the value of the returned policy. We did experiments on a $d = 1000$-arm MAB where the reward distribution is $r(a_i) \sim \mathcal{N}(i/1000, 1/4), \quad \forall i \in [d]$. We ran TRUST Algorithm 1 and LCB over 8 different random seeds. When we have a single sample for each arm, TRUST will get a similar score as LCB. However, TRUST give a much tighter statistical guarantee than LCB, in the sense that the lower bound output by TRUST is much higher than that output by LCB so that TRUST can output a policy that is guaranteed to achieved a higher value. Moreover, we found the policies output from TRUST are much more stable than those from LCB. In all runs, while the lowest value of the arm chosen by LCB is around 0.24, all policies returned by TRUST have values above 0.65 with a much smaller variance, as shown in Table 2.

Offline reinforcement learning. In this section, we apply Algorithm 1 to the offline reinforcement learning (RL) setting under the assumption that the logging policies which generated the dataset are accessible. To be clear, our goal is not to exceed the performance of the state of the art deep RL algorithms—our algorithm is designed for bandit problems—but rather to illustrate the usefulness of our algorithm and theory.

Since our algorithm is designed for bandit problems, in order to apply it to the sequential setting, we map MDPs to MABs. Each policy in the MDP maps to an action in the MAB, and each trajectory

return in the MDP maps to an experienced return in the MAB setting. Notice that this reduction disregards the sequential aspect of the problem and thus our algorithm cannot perform 'trajectory stitching' (Levine et al., 2020; Kumar et al., 2020; Kostrikov et al., 2021). Furthermore, it can only be applied under the assumption that the logging policies are known.

Specifically we consider a setting where there are multiple known logging policies, each generating few trajectories. We test Algorithm 1 on some selected environments from the D4RL dataset (Fu et al., 2020) and compare its performance to the (CQL) algorithm (Kumar et al., 2020), a popular and strong baseline for offline RL algorithms. Since the D4RL dataset does not directly include the logging policies, we generate new datasets by running Soft Actor Critic (SAC) (Haarnoja et al., 2018) for 1000 episodes. We store 100 intermediate policies generated by SAC, and roll out 1 trajectory from each policy.

	LCB	TRUST
mean reward	0.718	0.725
mean lower bound	0.156	0.544
variance	0.265	0.038
minimal reward	0.239	0.658

Table 2: Comparison between LCB and TRUST (Algorithm 1) on a data-starved MAB with 1000 arms whose reward distribution follows $r(a_i) \sim \mathcal{N}(i/1000, 1/4)$. Both methods are repeated on 8 random seeds.

We use some default hyper-parameters for CQL.[1] We report the unnormalized scores in Table 3, each averaged over 4 random seeds. Algorithm 1 achieves a score on par with or higher than that of CQL, especially when the offline dataset is of poor quality and when there are very few—or just one—trajectory generated from each logging policy. Notice that while CQL is not guaranteed to outperform the behavioral policy, TRUST is backed by Theorem 4.1.

Additionally, while CQL took around 16-24 hours on one NVIDIA GeForce RTX 2080 Ti, TRUST only took 0.5-1 hours on 10 CPUs. The experimental details are included in Appendix H. Moreover, while the performance of CQL is highly reliant on the choice of hyper-parameters, TRUST is essentially hyper-parameters free.

		CQL	TRUST
Hopper	1-traj-low	499	999
	1-traj-high	2606	3437
Ant	1-traj-low	748	763
	1-traj-high	4115	4488
Walker2d	1-traj-low	311	346
	1-traj-high	4093	4097
HalfCheetah	1-traj-low	5775	5473
	1-traj-high	9067	10380

Table 3: Unnormalized score of CQL and TRUST in 4 environments from D4RL. In 1-traj-low case, we take the first 100 policies in the running of SAC. In 1-traj-high case, we take the $(10x + 1)$-th policy for $x \in [100]$. We sample one trajectory from each policy we take in all experiments.

6 Conclusion

In this paper we make a substantial contribution towards sample efficient decision making, by designing a data-efficient policy optimization algorithm that leverages offline data for the MAB setting. The key intuition of this work is to search over stochastic policies, which can be estimated more easily than deterministic ones. The design of our algorithm is enabled by a number of key insights, such as the use of the localized gaussian complexity which leads to the definition of the critical radius for the trust region. We believe that these concepts can be used more broadly to help design truly sample efficient algorithms, which can in turn enable the application of decision making to new settings where a high sample efficiency is critical.

References

Yasin Abbasi-Yadkori, David Pal, and Csaba Szepesvari. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[1] We use the codebase and default hyper-parameters in `https://github.com/young-geng/CQL`.