Tractable contextual bandits beyond realizability

Sanath Kumar Krishnamurthy
Stanford University

Vitor Hadad
Stanford University

Susan Athey
Stanford University

Abstract

Tractable contextual bandit algorithms often rely on the realizability assumption – i.e., that the true expected reward model belongs to a known class, such as linear functions. In this work, we present a tractable bandit algorithm that is not sensitive to the realizability assumption and computationally reduces to solving a constrained regression problem in every epoch. When realizability does not hold, our algorithm ensures the same guarantees on regret achieved by realizability-based algorithms under realizability, up to an additive term that accounts for the misspecification error. This extra term is proportional to T times a function of the mean squared error between the best model in the class and the true model, where T is the total number of time-steps. Our work sheds light on the bias-variance trade-off for tractable contextual bandits. This trade-off is not captured by algorithms that assume realizability, since under this assumption there exists an estimator in the class that attains zero bias.

1 Introduction

Contextual bandit algorithms serve as a fundamental tool for online decision making and have been used in a wide range of settings from recommendation systems [Agarwal et al., 2016] to mobile health [Tewari and Murphy, 2017], and due to their applicability over the past couple of decades there has been an increasing amount of research in contextual bandits [Lattimore and Szepesvári, 2020]. However, the performance of many common algorithms relies on an assumption called "realizability", which requires the analyst to possess some knowledge about the underlying data generating process – and often also relies on some luck that the process be relatively simple. When this assumption is satisfied, there exist algorithms that are statistically optimal and computationally tractable (in a sense we'll discuss more below). However, when it is violated, the performance of these algorithms can degrade in unexpected ways. The search for tractable algorithms that do not rely on this assumption an ongoing open problem [Foster et al., 2019]. In this work, we propose an algorithm that is optimal when the "realizability" assumption is satisfied and whose behavior is accurately characterized in its absence. We will also point to directions of research that may help do away with this assumption entirely.

Our underlying setup is the general stochastic contextual bandit setting. Using potential outcome notation, observations are represented as a sequence of iid random variables (x_t, r_t), where $x_t \in \mathcal{X}$ stands for a context in arbitrary set $\mathcal{X}$ and $r_t \in [0,1]^K$ is a vector of rewards, where $K := |\mathcal{A}|$ is the (finite) number of actions. Upon selecting one of action $a_t \in \mathcal{A}$, the algorithm observes $r_t(a_t)$. Therefore, the sequence of observed data points is $(x_t, a_t, r_t(a_t))$. Here t denotes the time-step which and is also the index for the sequence of observations. This sequence has length T, which may be known or unknown. A "policy" is a deterministic mapping from contexts to actions, representing a particular action selection strategy. Relative to the set of all policies $\mathcal{A}^{\mathcal{X}}$, we define the optimal policy as $\pi^* = \arg\max_{\pi \in \mathcal{A}^{\mathcal{X}}} \mathbb{E}_{x_t, r_t}[r_t(\pi(x_t))]$, where the expectation is taken over contexts and rewards.[1] A "reward model" or "outcome model" is a function that (potentially inaccurately) represents the conditional expectation of potential outcomes given action and

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

[1] Uniqueness of the optimal policy is not important for our results.

context. Reward models will often be represented as $f(x, a)$. We say that a reward model "induces a policy π" if $\pi(x) \in \arg\max_a f(x, a)$ for every x.

The goal of bandit algorithms is to find a sequence of actions that maximizes the sum of rewards observed during the experiment or, equivalently, to minimize cumulative regret, defined as the difference between the reward that was observed and that that would have been observed under the optimal policy,

$$R_T := \sum_{t=1}^{T} r_t(\pi^*(x_t)) - r_t(a_t). \tag{1}$$

The statistical performance of different algorithms is characterized by the rate at which (1) grows with the length of the experiment T.

Contextual bandit algorithms can often be categorized into three groups, depending on what is assumed about the underlying data-generating process. The first group of algorithms are the "agnostic" algorithms. These algorithms make no assumptions about the reward model, and they learn the best policy in some fixed class $\Pi \subseteq \mathcal{A}^{\mathcal{X}}$ while balancing the exploration-exploitation trade-off. To do this, these algorithms [Beygelzimer et al., 2011, Dudik et al., 2011, Agarwal et al., 2014] need to construct a distribution over the policies Π in every epoch. Constructing this distribution is computationally challenging, and hence this approach is colloquially referred to as the "Monster" [Langford, 2014]. We now focus on the results in [Agarwal et al., 2014] because computationally and statistically, they provide the state of the art agnostic algorithms. When Π is a finite class, [Agarwal et al., 2014] present an algorithm called ILTCB that constructs a distribution with support size of $\mathcal{O}(\log|\Pi|)$ and the regret [2] against the best policy in Π scales at the rate $\tilde{\mathcal{O}}(KT\log|\Pi|)$. Each policy in the support of this distribution can be computed by solving the following cost-sensitive classification problem:

$$\arg\max_{\pi\in\Pi} \sum_{s=1}^{t} \hat{r}_s(\pi(x_s)), \tag{2}$$

where $(x_s, \hat{r}_s)$ is some sequence in $\mathcal{X} \times [0, 1]^K$. When Π is large, the support of the distribution needed to be computed in every epoch of ILTCB may be large and hence would still be impractical to implement. To overcome this limitation, [Agarwal et al., 2014] propose a heuristic called Online Cover (with parameter l) that computes a distribution over polices using the same approach as ILTCB but stops increasing the support of this distribution after computing some fixed number of policies l for the support. To the best of our knowledge there aren't any theoretical guarantees for Online Cover. Further, finding an exact solutions to (2) is generally intractable, so implementations of Online Cover use heuristics to solve this optimization problem.

[2]Note that while this notion of regret compares against the best policy in Π, the notion of regret used in this paper compares against the true optimal policy π^*.

The second group of algorithms requires knowledge about some set of functions $\mathcal{F}$ that is assumed to include the true reward model. That is, that there exists a function $f^* \in \mathcal{F}$ such that $f^*(x, a) = \mathbb{E}_{x_t, r_t}[r_t(a)|x_t = x]$ for all contexts and actions. This assumption is called "realizability", and it often allows for algorithms that computationally tractable and typically easier to implement. Computationally, algorithms in this category rely on being able to solve the regression problem

$$\hat{f}_t = \arg\min_{f\in\mathcal{F}} \sum_{s=1}^{t-1} (f(x_s, a_s) - r_s(a_s))^2, \tag{3}$$

or a weighted version of it, either online or offline. A routine that solves (3) is called "regression oracle". This class includes algorithms built on upper confidence bounds [Li et al., 2010, Abbasi-Yadkori et al., 2011, Foster et al., 2018] or Thompson sampling [Agrawal and Goyal, 2013, Russo et al., 2018], and algorithms built on simple probabilistic selection strategies [Abe and Long, 1999, Foster and Rakhlin, 2020, Simchi-Levi and Xu, 2020]. Regret rates for this class of algorithms are related to the complexity class of the outcome model $\mathcal{F}$, and under realizability optimal algorithms attain a rate of $\tilde{\mathcal{O}}(\sqrt{TK\log|\mathcal{F}|})$ for finite classes $\mathcal{F}$ (similar results are available for more general classes). In particular, the FALCON algorithm of [Simchi-Levi and Xu, 2020] will serve as the basis of our method attains this statistically optimal rate (so long as realizability holds) and is computationally tractable, in that the algorithm only needs to solve the problem (3) a small (at most logarithmic) number of times during the experiment.

A third set of bandit algorithms that does not fall neatly into any of the two categories above are algorithms that allow for a non-parametric model class. For example, in [Rigollet and Zeevi, 2010], [Perchet et al., 2013] the reward model is assumed to be Hölder continuous but non-differentiable, and in [Hu et al., 2020], [Gur et al., 2019] it satisfies a

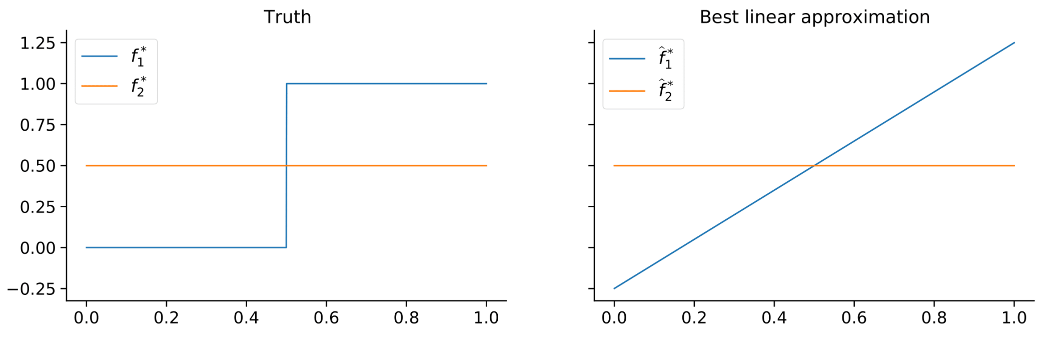

Figure 1: **Example**. Data-generating process (left) and its best approximation in the class of linear functions (right). The induced policies are the same.

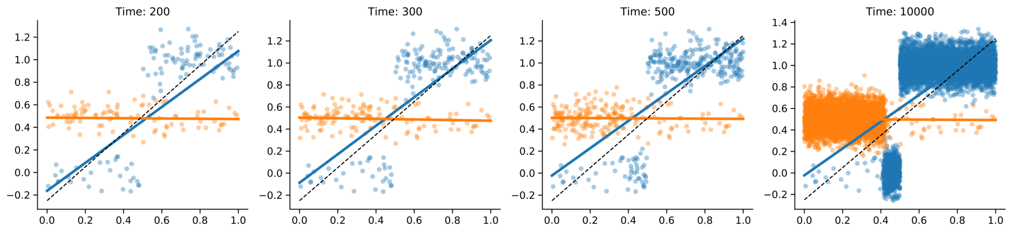

Figure 2: **Evolution of model estimates** for the UCB example. Although the model is initially correct, the distribution shift caused by the assignment mechanism ends up biasing reward estimates over time. Best linear approximation in black, dotted line. In our proposed algorithm, the blue and black lines are kept appropriately close, preventing this phenomenon.

Hölder smoothness assumption. The main characteristic of this class of algorithms is that they partition the covariate space into hypercubes of appropriate size and run multi-armed bandit algorithms within each cube. Depending on the smoothness of the reward model, there can be some information sharing across cubes that induces correlation across assignments in adjacent hypercubes and decreases regret. Although this is a very interesting direction of research, the structure of these algorithms forces the running time to exponentially depend on the context dimension, making them computationally intractable and hard to implement for most real life problems. Hence, for the rest of the paper, we will focus on the first two classes of algorithms.

1.1 The problem with realizability

As we have mentioned before, realizability is an extremely convenient and pervasive assumption in many tractable contextual bandit algorithms, but it is nevertheless very strong. In this section, we attempt to shed light on some issues that may arise in its absence.

To fix ideas, start from the following illustration. There are two actions and a single context is distributed uniformly on the unit interval. However, unbeknownst to the researcher, the conditional average rewards for each arm are a step function $f_1^*(x_t) := \mathbb{I}\{x_t > 0.5\}$ and a constant $f_2^*(x_t) \equiv 0.5$ (Figure 1). Rewards are observed with error $\epsilon_t \sim \mathcal{N}(0, .01)$. The researcher erroneously assumes that both can be realized in the class $\mathcal{F}$ of linear functions. Fortunately, in this example the best linear approximation $(\hat{f}_1^*, \hat{f}_2^*)$ induces a good policy. In fact, it coincides with the one the researcher would obtain if they had knowledge about the true function class – i.e., the policy induced by the best linear approximation $\hat{\pi}^*$ defined by $\hat{\pi}(x) = 1$ if $x > 0.5$ and $\hat{\pi}(x) = 2$ otherwise actually coincides with the policy induces by the true model

$\pi^*(x) := \arg\max_a f^*(x, a)$. Therefore, if the sequence of fitted models $(\hat{f}_{1,t}, \hat{f}_{2,t})$ converges to the best linear approximation $(\hat{f}^*_{1,t}, \hat{f}^*_{2,t})$, regret should decay to zero asymptotically. However, as we will see next, this convergence may not happen.

Let us assume that the researcher collects data via LinUCB [Li et al., 2010], with model updates in batches of 100 observations. Figure 2 shows the evolution of the estimated models $(\hat{f}_{1,t}, \hat{f}_{2,t})$ over time for a single simulation. After about a few hundred observations, the estimated model approximates the best linear approximation well and regret is small since the induced policy is nearly optimal. However, by continuing to assign treatments following this policy (plus some negligible exploration), the distribution of observations changes, which pushes the model away from the best linear approximation. In turn, this causes per-period regret to increase over time, as we show on Figure 3.

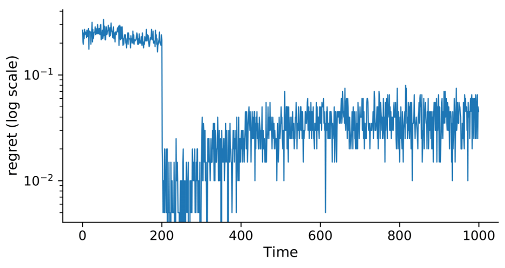

Figure 3: **Evolution of per-period regret** for the UCB example described in the text. Regret initially decreases because the estimated models $(\hat{f}_{1,t}, \hat{f}_{2,t})$ are close to the best linear approximations $(\hat{f}^*_1, \hat{f}^*_2)$, which in this example induces a good policy. However, as we gather more data that was collected with an exploitation objective, the estimated model diverges and the performance of the algorithm degrades. (Average across 100 simulations.)

The previous example demonstrates that in the absence of realizability the dynamics of adaptive data collection can lead the algorithm to learn a policy that is suboptimal relative to the one that it would have learned under non-adaptive data collection. As an extreme thought example, one may also consider a situation in which actions are assigned via the optimal policy π^*. If we were to fit a linear model using exclusively this data, we would estimate that $\hat{f}_1 \equiv 1$ and $\hat{f}_2 \equiv 0.5$, which would in turn induce the policy $\pi(x) \equiv 1$ – a policy that always assigns arm 1 everywhere and therefore clearly suboptimal. In fact, more can be said. We can construct examples where even when the approximation error b is arbitrarily small, given data from the optimal policy, the confidence intervals used by LinUCB would tightly concentrate around a high regret policy, showing that the confidence intervals used by LinUCB are extremely sensitive to the realizability assumption (See Appendix E).

To prevent this phenomenon, in the next section we consider an algorithm that constraints the estimate of the outcome model $\hat{f}$ to be close to $\hat{f}^*$. This also allows us to derive upper bounds on regret in terms of the deviation of the best in-class model $\hat{f}^*$ from the true model f^*. This characterization is important as it allows us to take into account regret incurred due to model misspecification – a cost that often assumed away under realizability.

1.2 Related work on misspecification

As we have discussed above, bandit algorithms relying on regression oracles are computationally tractable, and when their model is well-specified they often exhibit attractive statistical properties. More recently, there has been interest in developing algorithms that are robust to misspecification. These works differ in how they define and measure misspecification, and how their regret bound degrade as the level of misspecification increases.

[Neu and Olkhovskaya, 2020, Zanette et al., 2020] assume that the absolute deviation between the true reward function and its best linear approximation is at most ϵ uniformly across contexts and actions. Under this assumption, they develop bandit algorithms whose regret overhead due to misspecification is bounded in terms of this measure of misspecification ϵ. Under the same measure of misspecification, [Foster and Rakhlin, 2020] provide similar results that hold for any class of models that have an online regression oracle.

This type of uniform bound on model misspecification can be arbitrarily large even in relatively benign examples (see Appendix E). Concurrent work of [Foster et al., 2020] use a different measure of misspecification that allows them to derive tighter regret bounds while relying on online regression oracles. Their measure of misspecification turns out to be very similar to the one we use in this work, however we rely on constrained offline regression oracles instead. Moreover, [Foster et al., 2020] also adapt to unknown misspecification by relying on master algorithms (see Section 3).

[Lattimore et al., 2020] and [Ghosh et al., 2017] also study the related problem of misspecified non-contextual linear bandits.

2 Main results

We propose an algorithm that we call Epsilon-FALCON, which is a modification of the "FAst Least-squares-regression-oracle CONtextual bandits", or FALCON algorithm described in [Simchi-Levi and Xu, 2020]. The main departure from FALCON is that although we do posit some "tentative" set $\mathcal{F}$ that could contain the true outcome model, our regret guarantees do not depend on this assumption being satisfied. For simplicity of exposition we will initially assume that $\mathcal{F}$ is a convex subset of a d-dimensional linear space [3], but our results can be extended to more complex classes as we show later.

We will need some additional notation. Let f^* represent the true outcome model, i.e., $f^*(x,a) = \mathbb{E}_{x_t,r_t}[r_t(a)|x_t = x]$ for all x and a. Moreover, let $\hat{f}^*$ denote the best in-class approximation to the true outcome model when data is collected non-adaptively, or

$$\hat{f}^* := \arg\min_{f\in\mathcal{F}} \mathop{\mathbb{E}}_{x\sim D_{\mathcal{X}}} \mathop{\mathbb{E}}_{a\sim \text{Unif}(\mathcal{A})}[(f(x,a) - f^*(x,a))^2], \tag{4}$$

where $D_{\mathcal{X}}$ is the distribution of contexts, and $\text{Unif}(\mathcal{A})$ is a probability distribution that assigns equal probability to every arm. The approximation error between these two functions is denoted as

$$b := \mathop{\mathbb{E}}_{x\sim D_{\mathcal{X}}} \mathop{\mathbb{E}}_{a\sim \text{Unif}(\mathcal{A})}[(\hat{f}^*(x,a) - f^*(x,a))^2]. \tag{5}$$

Naturally, the approximation error (5) will be zero when realizability holds. And when it doesn't hold, we will show that the algorithm will incur some regret whose upper bound increases with the approximation error. This is what allows us to accurately characterize the cost that we pay when we $\mathcal{F}$ is misspecified (i.e., $f^* \notin \mathcal{F}$).

Algorithm: Epsilon-FALCON is implemented in increasing epochs (batches) that are indexed by m. Each epoch m begins at period τ_{m-1}, we set epoch schedule so that $\tau_0 = 0$, $\tau_1 \geq 4$, and $\tau_{m+1} = 2\tau_m$ for any epoch $m \geq 1$. Every epoch m starts out with an estimated reward model $\hat{f}_m$ obtained at the end of the last batch, with $\hat{f}_1 \equiv 0$. For a fraction ϵ of each epoch, called the "passive" phase, Epsilon-FALCON draws actions uniformly at random. For the remaining $1-\epsilon$ fraction of the epoch, in what we call the "active" phase, it acts as a modified version of FALCON.[4]

Our action selection mechanism is the same as FALCON's, so let's briefly review it. At each epoch m, given the current reward model estimate $\hat{f}_m$ and a scaling parameter $\gamma_m > 0$, actions are drawn from the probability distribution described by the following "action selection kernel",

$$p_m(a|x) := \begin{cases} \frac{1}{K+\gamma_m\left(\hat{f}_m(x,\hat{a})-\hat{f}_m(x,a)\right)} & \text{for } a \neq \hat{a} \\ 1 - \sum_{a'\neq\hat{a}} p(a'|x) & \text{for } a = \hat{a}. \end{cases} \tag{6}$$

where $\hat{a} = \max_a \hat{f}_m(a,x)$ is the best predicted action. The assignment rule (6) ensures that actions that are predicted to be good according to the current model estimate $\hat{f}_m$ are given higher probability. The scaling parameter, set to $\gamma_m \simeq \sqrt{K(\tau_{m-1} - \tau_{m-2})/(d\ln(m/\delta))}$ with initial values $\gamma_1 = 1$, control the degree of exploration during the active phase, with higher values of γ_m indicating less exploration. We may sometimes refer to γ_m and ϵ as the active and passive exploration parameters respectively.

The main difference between our method and FALCON is in how we estimate the outcome model $\hat{f}_{m+1}$ from data collected in the previous epoch m. The original algorithm simply uses the estimator that minimizes empirical risk on data collected in the previous time-steps, but as we saw in the example in Section 1.1, when realizability fails the sequence of estimators $\hat{f}_m$ may not converge to $\hat{f}^*$. This is due to the fact that the empirical risk minimizer when data is collected adaptively may be very different from the one attained when data is collected non-adaptively, and its performance may not be well understood (See Figure 4). In order to ensure that our estimates converge to $\hat{f}^*$, our algorithm uses a "*constrained* regression oracle" that ensures that the estimated model is always close to the best approximation $\hat{f}^*$. Let's see how this is done.

Denote the data collected using the passive and active phases of the epoch m by S'_m and S_m respec-

[3]Consider the class of estimators $\mathcal{F}$ where linear functions estimate rewards for each arm using a total of d parameters. Note that this is a special case of requiring $\mathcal{F}$ to be a convex subset of a d-dimensional linear space. Hence, the guarantees in Theorem 1 hold for stochastic linear bandits.

[4]More precisely, it acts as a modified version of the FALCON+ algorithm in the same paper, but the distinction is minor enough that we will ignore it for the purposes of naming our method.

tively. Moreover, let $\mathcal{F}'_m$ denote the subset of functions $f \in \mathcal{F}$ for which the following constraint in satisfied,

$$\sum_{(x,a,r(a))\in S'_m} (f(x,a)-r(a))^2 \leq \alpha_m + C_1 d \ln(12m^2/\delta) \tag{7}$$

where $\alpha_m := \min_{g\in\mathcal{F}} \sum_{(x,a,r(a))\in S'_m} (g(x,a)-r(a))^2$ is the sum of squared residuals in the model fitted on the data collected in the "passive" phase, and C_1 is a constant chosen appropriately to ensure that $\hat{f}^*$ also lies in $\mathcal{F}'$ with probability at least $1-\delta/(12m^2)$. [5]

The estimated model $\hat{f}_{m+1}$ will be constrained to lie in this set. More specifically, it is the output of the following constrained regression problem:

$$\begin{aligned} \min_{f\in\mathcal{F}} \quad & \sum_{(x,a,r(a))\in S_m} (f(x,a)-r(a))^2 \\ \text{s.t.} \quad & f \in \mathcal{F}'_m. \end{aligned} \tag{8}$$

The intuition, again, is that since $\hat{f}_{m+1} \in \mathcal{F}'_m$ by construction, and since $\hat{f}^* \in \mathcal{F}'_m$ with high probability, the two will likely remain close. And since $\mathcal{F}'_m$ shrinks over time, $\hat{f}_{m+1}$ must ultimately converge to $\hat{f}^*$. Therefore, the convergence issues we saw in our example in Section 1.1 cannot happen. This is what allows us to derive regret guarantees even when realizability fails (see Figure 4 for an intuitive illustration). The full description and the pseudocode for the general algorithm can be found in the Appendix (Algorithm 1). [6]

Computational tractability of the constrained regression problem: Note that Epsilon-FALCON is very easy to implement given a constrained regression oracle. Hence, for the computational tractability of Epsilon-FALCON, it is sufficient to argue that the constrained regression problem is computationally tractable. When $\mathcal{F}$ is the class of linear reward models, then clearly the constrained regression problem is a convex and can be solved efficiently. In general, when $\mathcal{F}$ is any convex class, we show that the constrained regression problem can be solved efficiently with a weighted regression oracle (see Appendix D). Hence we can use any of the many existing algorithms for weighted regression as a subroutine to solve the constrained regression problem. While this is one approach to solve the constrained regression problem, in practice directly solving the constrained regression problem may be faster.

[5]We pin down the value of this constant in the Appendix

[6]Except for the choice of γ_m and the RHS of the constraint Equation (7), the algorithm for general $\mathcal{F}$ is the same as the description in this section.

Theorem 1 provides a high probability regret guarantee for Epsilon-FALCON when $\mathcal{F}$ is a convex subset of some d-dimensional linear space.

Theorem 1 (Linear case)**.** *Suppose $\mathcal{F}$ is a convex subset of a d-dimensional linear space. With probability at least $1-\delta$, Epsilon-FALCON with passive exploration parameter $\epsilon > 0$ attains the following regret guarantee:*

$$R_T \leq \mathcal{O}\left(\sqrt{KTd\ln\left(\frac{\ln(T)}{\delta}\right)} + KT\sqrt{\frac{b}{\sqrt{\epsilon}}} + \epsilon T\right). \tag{9}$$

The guarantees in (9) consist of three terms. The first term is the regret due to the complexity of the class $\mathcal{F}$, and is the bound guaranteed by realizability based algorithms like FALCON under realizability. The second term can be interpreted as the "cost of misspecification", this term depends on the approximation error b and the passive exploration parameter ϵ. Finally, the third term is the regret incurred in the passive phase and depends only on the passive exploration parameter ϵ.

At first glance, the result in (9) may look rather weak due to the linear dependence in the horizon T. However, we contend that any algorithm that that works with a restricted class of policies or reward models, including agnostic algorithms like ILTCB [Agarwal et al., 2014], will incur some linear regret if these restrictions are violated. In Theorem 1 we simply make this issue explicit, as one of our goals is to accurately characterize the bias-variance trade-off in our problem. Our results show that, if the practitioner is willing to spend ϵT regret in the passive phase, then in the active phase excess regret due to misspecification will be $\mathcal{O}(KT\sqrt{b/\sqrt{\epsilon}})$. On the other hand, realizability based approaches do not have any guarantees under general misspecification.

As a thought experiment, suppose we knew the approximation error b or could make an educated guess about it. In that case we could choose ϵ as a function of b so as to optimize (9) and obtain the next result.

Corollary 1 (Linear case with known b)**.** *In the setting of Theorem 1, if the passive exploration parameter is set to $\epsilon = cK^{4/5}b^{2/5}$ for some constant*

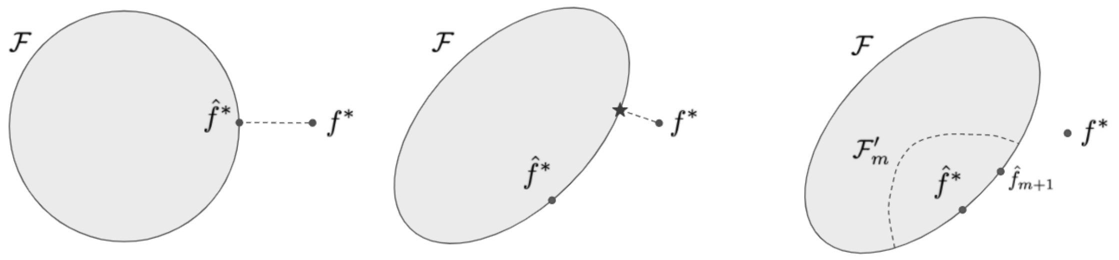

Figure 4: Intuition for our method. Left: the function $\hat{f}^*$ is the best in-class approximation to f^* under non-adaptive data collection. Middle: under a different distribution, the best in-class approximation (starred) may lie very far away from $\hat{f}^*$, and there are no guarantees on its performance. Right: in our method, we construct a shrinking sequence of sets $\mathcal{F}'_m$ that contain $\hat{f}^*$ with high probability, and ensure that our model estimates lie in this set.

$c > 0$, *we have the following bound:*

$$R_T \le \mathcal{O}\left(\sqrt{KTd\ln\left(\frac{\ln(T)}{\delta}\right)} + K^{4/5}b^{2/5}T\right). \quad (10)$$

This result is interesting because it tells us that if we were able to tune the passive exploration parameter optimally, we get improved regret rates that only depend on the complexity of $\mathcal{F}$ and the approximation error b, thus achieving a bias-variance trade-off over the entire horizon T. This suggests that tuning ϵ by estimating b may be a promising direction for future work to get algorithms with better regret guarantees.

Understanding the constrained regression problem: Having explained the overall algorithm, let's now understand the constrained regression problem in a bit more detail, so the reader will be able to follow the proof steps in the Appendix.

At the end of epoch m, we have the two kinds of data, that is the data from the passive phase of the epoch and the data from the active phase of the epoch. The data from the passive phase is used to construct $\mathcal{F}'_m$, and contains the best in-class approximation of the true outcome model $\hat{f}^*$ with high probability (see Lemma 7). The data from the active phase is used to select a "good" estimate within $\mathcal{F}'_m$ which in turn induces a "good" action selection kernel. A good action selection kernel has low regret, and ensures that the data generated by this kernel can be used to construct "good" estimates in the next epoch. In terms of exploration, there is a trade-off between these two properties as more exploration helps you generate "good" data but incurs higher regret. In terms of estimates, both these properties are related because good estimates come from good data. For simplicity let us focus on arguing that the action selections kernels we estimate generate "good" data and believe that the active exploration parameter γ_m is set optimally. In particular, we say the action selection kernel generates "good" data if the reward of the policy induced by $\hat{f}^*$ can be estimated using the data generated by this kernel. Note that this is trivially ensured when actions are selected uniformly at random, as we did in the first epoch. In later epochs, as the kernel gets less explorative (γ_m increases), to ensure this we need the estimator ($\hat{f}_{m+1}$) that induces this action selection kernel to be close to the best in-class model ($\hat{f}^*$). More mathematically, as shown in Lemma 9, we need the root mean squared difference between $\hat{f}_{m+1}$ and $\hat{f}^*$ to shrink at the same rate as γ_m increases. This property is guaranteed by the fact that both $\hat{f}_{m+1}$ and $\hat{f}^*$ lie in $\mathcal{F}'_m$ with high probability, and by the fact that $\mathcal{F}'_m$ is sufficiently small as we have collected enough data in the passive phase to ensure this (see Lemma 7). Additionally, this property helps us ensure that our action selection kernels p_m are stable over time, in the sense that if the reward of a policy could be estimated from the data generated by p_{m+1} (in expectation) then the reward of this policy could also be estimated by the data generated by p_m (in expectation), see Lemma 10 for a more formal statement. In other words, the set of policies that we implicitly consider do not erratically change over time and only decrease.

General classes of outcome models: Although for concreteness we have explained our results when $\mathcal{F}$ is a convex subset of a d-dimensional linear space, Theorem 1 readily extends to more general classes of functions. In particular we can extend Theorem 1 whenever $\mathcal{F}$ is a convex and satisfies Assumption 1. In terms of the algorithm, except for the choice of γ_m and the RHS of the constraint Equation (7), Epsilon-FALCON for general $\mathcal{F}$ is the same as the description in this section. See Algorithm 1 in the Appendix for more details. Stating Assumption 1 can get cumbersome quickly, here we state an informal version of this assumption, followed by Theorem 2, and applications of this Theorem to various convex classes $\mathcal{F}$. In what follows $\mathbf{comp}(\mathcal{F})$ will denote an appropriate measure of complexity, like VC subgraph dimension or entropy.

Main Assumption: We now state an informal version of Assumption 1. Let n denote the number of data points collected from some distribution. Suppose we have $\rho \in (0,1]$, $\rho' \in [0,\infty)$, and $C > 0$. Further suppose for any convex subset $\mathcal{F}'$ of $\mathcal{F}$ and $\zeta \in (0,1/2)$, with probability $1-\zeta$, for any $\eta \geq C\ln^{\rho'}(n)\ln(1/\zeta)\mathbf{comp}(\mathcal{F})/n^{\rho}$, the empirical and true risks of any estimators in $\mathcal{F}'$ are "close" in the following sense:

- If the population risk of any estimator in $\mathcal{F}'$ is smaller than η, then its empirical risk is not larger than $3\eta/2$.
- If the empirical risk of any estimator in $\mathcal{F}'$ is smaller than η, then its population risk is not larger than 2η.

Theorem 2 (Main result)**.** *Suppose $\mathcal{F}$ is a convex set and suppose Assumption 1 holds. Then with probability at least $1-\delta$, Epsilon-FALCON with passive exploration parameter $\epsilon > 0$ attains the following regret guarantee:*

$$R_T \leq \mathcal{O}\left(\sqrt{KT^{2-\rho}\ln^{\rho'}(T)\ln(\frac{\ln(T)}{\delta})\boldsymbol{comp}(\mathcal{F})} + KT\sqrt{\frac{b}{\sqrt{\epsilon^{\rho}}}} + \epsilon T\right). \tag{11}$$

In Appendix C, we provide convenient Lemmas to prove Assumption 1 for various convex classes $\mathcal{F}$. These Lemmas directly follow from results in [Koltchinskii, 2011]. In fact, Theorem 1 is implied by Theorem 2 and results stated in Appendix C. We now go over similar results that follow from Theorem 2 and Appendix C.

Example 1: Suppose $\mathcal{F}$ is convex and has VC-subgraph dimension V. Then with probability $1-\delta$, Epsilon-FALCON guarantees the following bound on the regret R_T:

$$\mathcal{O}\left(\sqrt{KTV\ln\left(\frac{T}{V}\right)\ln\left(\frac{\ln(T)}{\delta}\right)} + KT\sqrt{\frac{b}{\sqrt{\epsilon}}} + \epsilon T\right).$$

Example 2: Suppose $\mathcal{F}$ is a convex hull of class with VC-subgraph dimension V. Then with probability $1-\delta$, Epsilon-FALCON guarantees the following bound on the regret R_T:

$$\mathcal{O}\left(\sqrt{KT^{\frac{2+3V}{2+2V}}V^{\frac{2+V}{2+2V}}\ln\left(\frac{\ln(T)}{\delta}\right)} + KT\sqrt{\frac{b}{\sqrt{\epsilon}}} + \epsilon T\right).$$

Example 3: Suppose for some $\rho \in (0,1)$, the empirical entropy is bounded by $\mathcal{O}(\epsilon^{-2\rho})$ for all empirical distributions. Then with probability $1-\delta$, Epsilon-FALCON guarantees the following bound on the regret R_T:

$$\mathcal{O}\left(\sqrt{KT^{\frac{1+2\rho}{1+\rho}}\ln\left(\frac{\ln(T)}{\delta}\right)} + KT\sqrt{\frac{b}{\sqrt{\epsilon^{1/(1+\rho)}}}} + \epsilon T\right).$$

3 Discussion

This paper's contribution is twofold. First, to illustrate how algorithms that rely on realizability may incur unexpected regret when this assumption is violated. We saw in Section 1.1 that one can construct examples where regret is large even in relatively benign settings. Second, to propose a flexible family of computationally tractable algorithm that are less sensitive to realizability. Our analysis in Section 2 characterizes the behavior of regret under misspecification and gives us insight into the bias-variance trade-off in contextual bandits.

In terms of algorithm design, our proposed algorithm Epsilon-FALCON inherits the computational elegance of realizability based approaches like FALCON. In particular, a single estimator gives you an implicit distribution over policies via the action selection kernel and bypasses the need to explicitly construct a distribution over policies. Our key insight is that by using a constrained regression estimator, we can make this approach robust to misspecification at the expense of some additional regret in the passive phase.

We believe this work represents an important step towards the development of contextual bandit algorithms that are robust to misspecification. Natural extensions include the following.

Adapting to misspecification The performance of Epsilon-FALCON depends on the input parameter (ϵ). One natural way to address this deficiency may be to initialize multiple base algorithms with different choices of ϵ and use a master algorithm [Agarwal et al., 2017] to choose the best performing base algorithm. In fact, the recent work of [Foster et al., 2020] take this approach to adapt to unknown misspecification. The idea of using a master algorithm to adapt to unknown misspecification also appears in [Pacchiano et al., 2020], they use the algorithm in [Zanette et al., 2020] as a base algorithm to adapt to an unknown uniform misspecification error for linear contextual bandits. The final drawback is that we use naive uniform sampling for the passive phase. One may be able to achieve a tighter regret bound by using a more sophisticated exploration scheme for the passive phase.

More general classes Epsilon-FALCON requires the model class $\mathcal{F}$ to be convex. We use the convexity of $\mathcal{F}$ in several ways. When $\mathcal{F}$ is convex and has finite VC-dimension we get that $\rho = 1$ in Assumption 1. Convexity of $\mathcal{F}$ also allows us to solve the constrained regression oracle using only an offline weighted regression oracle (Section D). More importantly, convexity of $\mathcal{F}$ helps ensure that there is a unique best in-class estimator ($\hat{f}^*$) up to evaluation on a non-zero measure set, which in turn helps ensure that the active policy ($\pi_{\hat{f}_m}$) converges to our target policy ($\pi_{\hat{f}^*}$); see Lemmas 7, 9, and 10. As many online regression algorithms rely on convexity, one may expect this drawback to implicitly hold for algorithms that rely on online regression oracles such as [Foster and Rakhlin, 2020, Foster et al., 2020].

To close, we note that these results hint at the possibility of exploiting the bias-variance trade-off in tractable contextual bandits to perform good model selection. This would be an interesting direction for future work.

4 Acknowledgments

We are grateful for the generous financial support provided by the Sloan Foundation, Schmidt Futures and the Office of Naval Research grant N00014-19-1-2468. SKK acknowledges generous support from the Dantzig-Lieberman Operations Research Fellowship.

References

[Abbasi-Yadkori et al., 2011] Abbasi-Yadkori, Y., Pál, D., and Szepesvári, C. (2011). Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 2312–2320.

[Abe and Long, 1999] Abe, N. and Long, P. M. (1999). Associative reinforcement learning using linear probabilistic concepts. In *ICML*, pages 3–11. Citeseer.

[Agarwal et al., 2016] Agarwal, A., Bird, S., Cozowicz, M., Hoang, L., Langford, J., Lee, S., Li, J., Melamed, D., Oshri, G., Ribas, O., et al. (2016). Making contextual decisions with low technical debt. *arXiv preprint arXiv:1606.03966*.

[Agarwal et al., 2014] Agarwal, A., Hsu, D., Kale, S., Langford, J., Li, L., and Schapire, R. (2014). Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning*, pages 1638–1646.

[Agarwal et al., 2017] Agarwal, A., Luo, H., Neyshabur, B., and Schapire, R. E. (2017). Corralling a band of bandit algorithms. In *Conference on Learning Theory*, pages 12–38. PMLR.

[Agrawal and Goyal, 2013] Agrawal, S. and Goyal, N. (2013). Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pages 127–135.

[Bertsekas and Scientific, 2015] Bertsekas, D. P. and Scientific, A. (2015). *Convex optimization algorithms*. Athena Scientific Belmont.

[Beygelzimer et al., 2011] Beygelzimer, A., Langford, J., Li, L., Reyzin, L., and Schapire, R. (2011). Contextual bandit algorithms with supervised learning guarantees. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 19–26.

[Dudik et al., 2011] Dudik, M., Hsu, D., Kale, S., Karampatziakis, N., Langford, J., Reyzin, L., and Zhang, T. (2011). Efficient optimal learning for contextual bandits. *arXiv preprint arXiv:1106.2369*.

[Foster et al., 2018] Foster, D. J., Agarwal, A., Dudík, M., Luo, H., and Schapire, R. E. (2018). Practical contextual bandits with regression oracles. *arXiv preprint arXiv:1803.01088*.

[Foster et al., 2020] Foster, D. J., Gentile, C., Mohri, M., and Zimmert, J. (2020). Adapting to misspecification in contextual bandits. *Advances in Neural Information Processing Systems*, 33.

[Foster et al., 2019] Foster, D. J., Krishnamurthy, A., and Luo, H. (2019). Model selection for contextual bandits. In *Advances in Neural Information Processing Systems*, pages 14741–14752.

[Foster and Rakhlin, 2020] Foster, D. J. and Rakhlin, A. (2020). Beyond ucb: Optimal and efficient contextual bandits with regression oracles. *arXiv preprint arXiv:2002.04926*.

[Ghosh et al., 2017] Ghosh, A., Chowdhury, S. R., and Gopalan, A. (2017). Misspecified linear bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[Gur et al., 2019] Gur, Y., Momeni, A., and Wager, S. (2019). Smoothness-adaptive stochastic bandits. *arXiv preprint arXiv:1910.09714*.

[Hu et al., 2020] Hu, Y., Kallus, N., and Mao, X. (2020). Smooth contextual bandits: Bridging the parametric and non-differentiable regret regimes. In *Conference on Learning Theory*, pages 2007–2010.

[Jin et al., 2019] Jin, C., Netrapalli, P., and Jordan, M. I. (2019). What is local optimality in nonconvex-nonconcave minimax optimization? *arXiv preprint arXiv:1902.00618*.

[Koltchinskii, 2011] Koltchinskii, V. (2011). *Oracle Inequalities in Empirical Risk Minimization and Sparse Recovery Problems: Ecole d'Eté de Probabilités de Saint-Flour XXXVIII-2008*, volume 2033. Springer Science & Business Media.

[Langford, 2014] Langford, J. (2014). Interactive machine learning. *http://hunch.net/ jl/projects/interactive/index.html*.

[Lattimore and Szepesvári, 2020] Lattimore, T. and Szepesvári, C. (2020). *Bandit algorithms*. Cambridge University Press.

[Lattimore et al., 2020] Lattimore, T., Szepesvari, C., and Weisz, G. (2020). Learning with good feature representations in bandits and in rl with a generative model. In *International Conference on Machine Learning*, pages 5662–5670. PMLR.

[Li et al., 2010] Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM.

[Neu and Olkhovskaya, 2020] Neu, G. and Olkhovskaya, J. (2020). Efficient and robust algorithms for adversarial linear contextual bandits. In *Conference on Learning Theory*, pages 3049–3068. PMLR.

[Pacchiano et al., 2020] Pacchiano, A., Phan, M., Abbasi-Yadkori, Y., Rao, A., Zimmert, J., Lattimore, T., and Szepesvari, C. (2020). Model selection in contextual stochastic bandit problems. *arXiv preprint arXiv:2003.01704*.

[Perchet et al., 2013] Perchet, V., Rigollet, P., et al. (2013). The multi-armed bandit problem with covariates. *The Annals of Statistics*, 41(2):693–721.

[Rigollet and Zeevi, 2010] Rigollet, P. and Zeevi, A. (2010). Nonparametric bandits with covariates. *arXiv preprint arXiv:1003.1630*.

[Russo et al., 2018] Russo, D. J., Roy, B. V., Kazerouni, A., Osband, I., and Wen, Z. (2018). A tutorial on thompson sampling. *Now Publishers Inc.*

[Simchi-Levi and Xu, 2020] Simchi-Levi, D. and Xu, Y. (2020). Bypassing the monster: A faster and simpler optimal algorithm for contextual bandits under realizability. *Available at SSRN*.

[Tewari and Murphy, 2017] Tewari, A. and Murphy, S. A. (2017). From ads to interventions: Contextual bandits in mobile health. In *Mobile Health*, pages 495–517. Springer.

[Zanette et al., 2020] Zanette, A., Lazaric, A., Kochenderfer, M., and Brunskill, E. (2020). Learning near optimal policies with low inherent bellman error. In *International Conference on Machine Learning*, pages 10978–10989. PMLR.