Adapting to Misspecification in Contextual Bandits with Offline Regression Oracles

Sanath Kumar Krishnamurthy [1] **Vitor Hadad** [2] **Susan Athey** [2]

Abstract

Computationally efficient contextual bandits are often based on estimating a predictive model of rewards given contexts and arms using past data. However, when the reward model is not well-specified, the bandit algorithm may incur unexpected regret, so recent work has focused on algorithms that are robust to misspecification. We propose a simple family of contextual bandit algorithms that adapt to misspecification error by reverting to a good safe policy when there is evidence that misspecification is causing a regret increase. Our algorithm requires only an offline regression oracle to ensure regret guarantees that gracefully degrade in terms of a measure of the average misspecification level. Compared to prior work, we attain similar regret guarantees, but we do no rely on a master algorithm, and do not require more robust oracles like online or constrained regression oracles (e.g., (Foster et al., 2020a); (Krishnamurthy et al., 2020)). This allows us to design algorithms for more general function approximation classes.

1. Introduction

Contextual bandit algorithms are a fundamental tool in sequential decision making and have been used in a variety of applications (see e.g., Lattimore & Szepesvári, 2020, Section 1.4 for a review).

The finite-armed (stochastic) contextual bandit setting that this paper is concerned with can be described as follows. Over a sequence of rounds, a bandit algorithm receives some side information or "contexts", which is drawn from a fixed distribution. Upon receiving each context, the algorithm selects an action, and then receives a probabilistic reward whose distribution may depend on the context and action. The objective of the algorithm is to interactively learn a mapping from contexts to actions so as to maximize the rewards received during the experiment. In order to do so, it must efficiently trade off the need for resolving uncertainty about the value of each action (exploration), with the objective of maximizing rewards (exploitation).

Many contextual bandit algorithms make use of an estimate of the conditional mean function of rewards given contexts and arms, along with some measure of the uncertainty around this function. At a high level, if the model predicts that an action has high expected reward with low uncertainty, then the algorithm is more likely to select this action. This intuition leads to heuristics that are statistically optimal in some settings (e.g., Agrawal & Goyal, 2013; Li et al., 2010). Such algorithms also tend to be computationally tractable relative to alternatives (Agarwal et al., 2014), since all that is required is a predictive model and the ability to produce appropriate confidence intervals.

However, the success of such algorithms depends heavily on tenuous assumptions about the underlying data-generating process, as their performance guarantees often rely on the conditional mean function belonging to a particular class; e.g., that it be linear under some transformation of the contexts. This is often called "realizability" in the literature, and when violated can cause the algorithm to behave erratically.

In this paper, we suggest an algorithm that adapts to model misspecification. To illustrate the problem, we consider an example described in (Krishnamurthy et al., 2020) and use it to get insights into the behavior of a realizability-based algorithm `FALCON+` (Simchi-Levi & Xu, 2020) in a setting where realizability does not hold. Consider a two-arm contextual bandit setting:

$$\mathbb{E}[r_t | x_t = x, a_t = a] = \begin{cases} \mathbb{I}\{x_t > 0.5\} & \text{if a = 1} \\ 0.5 & \text{if a = 2} \end{cases} \quad (1)$$

where $x_t \sim \text{Uniform}[0, 1]$ represents the contexts observed at the beginning of the t-th round, a_t is the action taken, r_t is the reward, which is observed by the experimenter with noise $e_t \sim \mathcal{N}(0, 1)$. Although the conditional expectation of rewards (1) is clearly non-linear, the model is simple

[1]Management Science and Engineering, Stanford University, Stanford, CA, USA [2]Graduate School of Business, Stanford University, Stanford, CA, USA. Correspondence to: Sanath Kumar Krishnamurthy <sanathsk@stanford.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

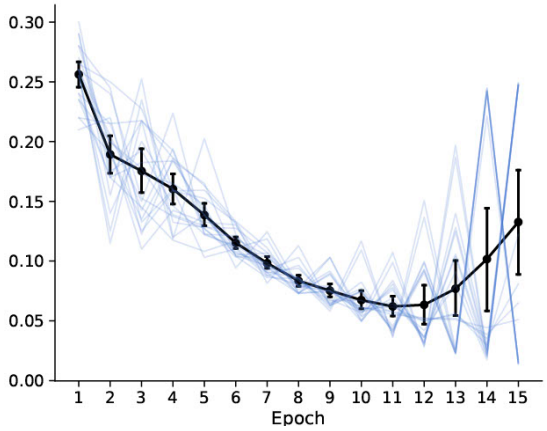

Figure 1. Illustration of the failure of a contextual bandit algorithm (FALCON+) when the model is not well-specified. Each epoch starts at round 2^m. Vertical bars are 95% confidence intervals around the average per-epoch average regret, aggregated over 50 simulations. Spaghetti plots are average per-epoch regret for 20 representative simulations.

enough that one could expect a bandit algorithm based on a misspecified linear model to do well.

Figure 1 shows the behavior of average regret for a realizability-based algorithm FALCON+ (Simchi-Levi & Xu, 2020), under the (incorrect) assumption that the underlying model is linear. The details of this algorithm are not particularly important. It suffices to understand that at the end of approximately every 2^m rounds (we call these intervals "epochs" and index them by m), the algorithm computed an estimate $\hat{f}(x, a)$ of (1), assuming a linear model and based on data from the previous epoch. Then, for every round in this epoch, it selects arms based on a probabilistic model where arms with high $\hat{f}(x, \cdot)$ have higher probability. A full description of this example is given in the appendix.

We notice two phenomena. First, the spaghetti plot reveals that average per-epoch reward has an oscillatory behavior, switching between very low and very high levels. This can be explained as follows. In some of the later epochs, the algorithm estimates a model that is close to the best linear approximation of (1), and thus it almost exclusively selects actions optimally (i.e., $a = 1$ when $x_t > .5$, $a = 2$ otherwise). In such epochs average rewards are high. However, in doing so, it collects data that is so skewed that it adversely affects the model estimated in the next epoch, causing the algorithm to make many mistakes and driving average rewards down again. However, these mistakes in turn allow the algorithm to get less skewed data for the misspecified arm, leading to estimate a good linear approximation to (1) again, and the cycle repeats.

More importantly, as a consequence of the erratic behavior just described we observe a second phenomenon: although the average regret for decreases initially, it begins to increase again after some time. The fundamental reason for this failure is that, by ignoring misspecification, the algorithm fails to accurately capture the how much uncertainty there is about the true model, in particular in regions where there is heavy extrapolation. The resulting performance is clearly suboptimal, and in this case, the model estimates do not even seem to converge. This problem is not idiosyncratic to FALCON+. For example, (Krishnamurthy et al., 2020) shows that LinUCB (Li et al., 2010) can converge to a suboptimal solution under the same example.

In this work, we propose a method to prevent the undesirable behavior described above. Our starting point is the FALCON+ algorithm. We modify it by introducing an additional step in which we test for a dip in average rewards (i.e., an increase in regret) caused by model misspecification. Upon finding evidence of this issue, we revert to a previously estimated "safe" policy and reduce further model updates. When there is no model misspecification, we attain the optimal regret guarantees inherited from FALCON+. When there is model misspecification, our regret bounds have an additional term that depends on a measure of misspecification.

Our results bound the regret overhead due to misspecification by $\mathcal{O}(\epsilon\sqrt{KT})$, where ϵ is the average misspecification error. Roughly speaking, we define the average misspecification error as a tight upper bound on the root mean squared difference between the true model and any function in the class of posited model that minimizes this squared difference (e.g., linear models). See Section 2 for a formal definition.

We now briefly describe how we get this result. Our starting point is FALCON+, which runs in epochs denoted by m. At each epoch m, the amount of exploitation is controlled by a parameter γ_m^{-1}. This parameter decreases with each epoch, as the algorithm learns about the environment and exploitation increases. We observe that while γ_m^{-1} is larger than the average misspecification ϵ – which is true for the initial epochs – cumulative regret decreases at the same rate as when realizability holds. When the average misspecification is comparable to this measure of exploration, we get that the expected instantaneous regret can be bounded by $\mathcal{O}(\epsilon\sqrt{K})$. Once γ_m^{-1} becomes less than the average misspecification, the expected instantaneous regret may increase. In fact, this behaviour is observed in Figure 1.

Our algorithm works by continuously testing for an unexpected change in cumulative rewards, and when that is detected the algorithm reverts to a good historically "safe". Our algorithm achieves the required regret bound because the expected instantaneous regret of this "safe" policy is bounded by $\mathcal{O}(\epsilon\sqrt{K})$ with high probability. Finally, to ensure that the regret guarantees in the initial epochs continue

to hold with the high-probability parameter of our choice, our algorithm uses a γ_m parameter that is about $\sqrt{2}$ times smaller compared to the one used in FALCON+.

Our algorithm is computationally efficient and flexible, as all that is needed is an offline regression oracle (i.e., the ability to fit a predictive model), thereby extending the reduction from contextual bandits to offline regression oracles (Simchi-Levi & Xu, 2020) to scenarios where realizability may not hold. Further our algorithm does not require knowledge of the average misspecification error, nor does it require the use of master algorithms.

Not relying on master algorithms to adapt to unknown misspecification allows us enjoy additional computational and statistical benefits. The computational benefit comes from not requiring to maintain and update $\lfloor \log(T) \rfloor$ base bandit algorithms[1]. We also inherit the optimal realizability-based regret bound from FALCON+ when the assumption holds, and save an additional $\mathcal{O}(\sqrt{\log(T)})$ factor that would have been incurred had we relied on a master with $\lfloor \log(T) \rfloor$ base algorithms.

Finally, our bounds on regret overhead due to misspecification are in terms of the average misspecification error and match the best known bounds from prior work (Foster et al., 2020a).

Our upper bound results are summarized in Section 2.4. To see that these upper bounds are optimal for contextual bandit algorithms that are based on regression oracles, up to constant factors, we also obtain matching lower bounds for the regret overhead due to misspecification (see Section 2.5). These results can also be interpreted as a quantification of the bias-variance trade-off for contextual bandits with regression oracles, see Section 2.4 for a more detailed discussion.

1.1. Related work

Over the last couple of decades, contextual bandit algorithms have been extensively studied (Lattimore & Szepesvári, 2020). However, the performance of many algorithms rely on the "realizability" assumption, which requires the analyst to know the form of the conditionally expected reward model. Moreover, the theoretical analysis of these algorithms that bounded cumulative regret often break down even under mild violations of the assumption. Since the realizability assumption may not be realistic, bandit algorithms that are robust to model misspecification have become a subject of intense recent interest.

In particular, recent works study algorithms that bound the regret overhead due to misspecification. When the true reward function is linear up to an additive error uniformly bounded by ϵ and contexts are d-dimensional, (Neu & Olkhovskaya, 2020) provide an algorithm that bound the regret overhead due to misspecification by $\mathcal{O}(\epsilon\sqrt{d}T)$. This paper assumes perfect knowledge of the covariance matrix for the distribution of contexts, but the algorithm does not require knowledge of ϵ.

This result is improved[2] and generalized by (Foster & Rakhlin, 2020). Given an online regression oracle for a class of value functions $\mathcal{F}$ and suppose the true reward function can be approximated by a function in $\mathcal{F}$ up to an additive error uniformly bounded by ϵ, (Foster & Rakhlin, 2020) provide an algorithm that achieves a regret overhead bound of $\mathcal{O}(\epsilon\sqrt{KT})$, where K is the number of arms. The algorithms proposed in this paper uses ϵ as an input parameter.

In many scenarios, one would expect a uniform bound on the additive error to be rather stringent. Concurrent work bound the regret overhead in terms of the "average misspecification error" (Foster et al., 2020a; Krishnamurthy et al., 2020), the notions of misspecification used in these papers are not the same but are similar. Roughly speaking, the average misspecification error is averaged over contexts but are uniformly bounded over policies. If ϵ is the average misspecification error, (Foster et al., 2020a) and (Krishnamurthy et al., 2020) achieve regret overhead bounds of $\mathcal{O}(\epsilon\sqrt{KT})$ and $\mathcal{O}(K^{2/5}\epsilon^{4/5}T)$. [3] The algorithms used in these papers assume access to robust regression oracles, like online regression oracles (Foster et al., 2020a) and offline constrained regression oracles (Krishnamurthy et al., 2020). Furthermore, (Krishnamurthy et al., 2020) assume that the set $\mathcal{F}$ is convex and require knowledge of ϵ (up to a constant factor) as an input parameter. In contrast, (Foster et al., 2020a) can adapt to the unknown misspecification without requiring any information about the misspecification parameter (ϵ).

To start with, (Foster et al., 2020a) provide a base algorithm that requires knowledge of ϵ (up to a constant factor) to achieve the regret overhead bound of $\mathcal{O}(\epsilon\sqrt{KT})$. They then consider $\lfloor \log(T) \rfloor$ base algorithms with different guesses for the average misspecification error (ϵ). Finally, they show that a master algorithm can be used to select the best performing base algorithm while continuing to achieve an overall regret overhead bound of $\mathcal{O}(\epsilon\sqrt{KT})$.

The idea of using master algorithms to adapt to unknown misspecification has also been used earlier in the context of misspecified linear bandits (Pacchiano et al., 2020). The

[1] These base algorithms make different guesses for the misspecification measure, and the master algorithm chooses the best performing base algorithm.

[2] Assuming $K < d$, which is often true.

[3] These bounds are non-trivial only if $\epsilon\sqrt{K} < 1$, clearly under this setting the bounds guaranteed by (Krishnamurthy et al., 2020) are weaker. This is because, their algorithm performs uniform sampling for a fraction of the time-steps.

misspecified linear bandit setup has also been studied in (Ghosh et al., 2017) and (Lattimore et al., 2020). These papers bound regret overhead in terms of the uniform misspecification error.

Oracle-based agnostic contextual bandit algorithms do not assume realizability and hence directly adapt to unknown misspecification[4] (Dudik et al., 2011; Agarwal et al., 2014). Unfortunately, these approaches suffer from computational issues that limit their implementability.

Within the broader literature of contextual bandits, we build on the recent line of work that provide reductions to offline/online squared loss regression (Foster et al., 2018; Foster & Rakhlin, 2020; Xu & Zeevi, 2020; Foster et al., 2020b). In particular, our work can be viewed as an extension of the analysis of (Simchi-Levi & Xu, 2020) to general scenarios that do not assume realizability.

2. Theory

To formalize the problem and discuss the properties of our algorithm, let's first establish some basic notation. Other symbols will be introduced later as appropriate.

Basic notation We let $\mathcal{A}$ denote the finite set of actions, K denote the number of arms (i.e. $K := |\mathcal{A}|$), and $\mathcal{X}$ denote the set of contexts. We let the notation $[n]$ denote the set $\{1, ..., n\}$. The (possibly unknown) number of rounds is denoted by T. Our algorithm will work in epochs indexed by m; the final round of each epoch is denoted by τ_m. We let $m(t)$ denote the epoch containing round t – that is, $m(t) := \min\{m|t \leq \tau_m\}$.

At every time-step $t \in [T]$, the environment draws a context $x_t \in \mathcal{X}$ and reward vector $r_t \in [0,1]^K$ from a fixed but unknown distribution D. Unless stated otherwise, all expectations are with respect to this distribution. Using potential outcome notation, we let $r_t(a)$ denote the reward that associated with arm a at time t.

We use p to denote probability kernels from $\mathcal{A} \times \mathcal{X}$ to $[0,1]$, and let $D(p)$ be the induced distribution over $\mathcal{X} \times \mathcal{A} \times [0,1]$, where sampling $(x, a, r(a)) \sim D(p)$ is equivalent to sampling $(x, r) \sim D$ and then sampling $a \sim p(\cdot|x)$.

The true conditional expectation function of rewards is denoted by $f^* : \mathcal{X} \times \mathcal{A} \to [0,1]$; i.e. $f^*(x,a) := \mathbb{E}[r_t(a)|x_t = x]$. We also let $D_{\mathcal{X}}$ denote the marginal distribution of D on the set of contexts $\mathcal{X}$. A model $f \in \mathcal{F}$ is any map from $\mathcal{X} \times \mathcal{A}$ to $[0,1]$. With a slight abuse of notation, for any model $f \in \mathcal{F}$ and context $x \in \mathcal{X}$, we let $f(x)$ denote the vector $(f(x,a))_{a \in \mathcal{A}}$ that lies in $[0,1]^K$.

[4]Here misspecification would correspond to the optimal policy not lying in the set of policies being explored.

By a policy we mean is a deterministic function from $\pi : \mathcal{X} \to \mathcal{A}$. The policy that maximizes the conditional mean of rewards is denoted by $\pi^*(x)$; i.e., $\pi^*(x) = \arg\max_a f^*(x,a)$. We also let π_f denote the policy induced by model f, which is given by $\pi_f(x) := \arg\max_a f(x,a)$ for every x. The goal of a contextual bandit algorithm is to bound cumulative regret:

$$R_T := \sum_{t=1}^{T} [r_t(\pi^*(x_t)) - r_t(a_t)]. \tag{2}$$

Misspecification Since our algorithm does not require that the model be well-specified, the class $\mathcal{F}$ may not contain the true model f^*. We denote by $\sqrt{B}$ the *average misspecification* error relative to $\mathcal{F}$,[5] where

$$B := \max_{p \in \mathcal{K}(\mathcal{F})} \min_{f \in \mathcal{F}} \mathop{\mathbb{E}}_{x \sim D_{\mathcal{X}}} \mathop{\mathbb{E}}_{a \sim p(\cdot|x)} [(f(x,a) - f^*(x,a))^2]. \tag{3}$$

Where $\mathcal{K}(\mathcal{F})$ is the set of probability kernels induced by $\mathcal{F}$,

$$\mathcal{K}(\mathcal{F}) := \Big\{ p : \mathcal{A} \times \mathcal{X} \to [0,1] \text{ is a probability kernel} | \\ \exists f_p \in \mathcal{F}, \text{ probability kernel } g_p : \mathcal{A} \times [0,1]^K \to [0,1], \\ \text{such that } \forall\ (x,a) \in \mathcal{X} \times \mathcal{A},\ p(a|x) = g_p(a|f_p(x)) \Big\}. \tag{4}$$

The set $\mathcal{K}(\mathcal{F})$ contains all probability kernels p from $\mathcal{A} \times \mathcal{X}$ to $[0,1]$ that can be represented by some pair (f_p, g_p), where f_p is a function in $\mathcal{F}$ and g_p is a probability kernel from $\mathcal{A} \times [0,1]^K$ to $[0,1]$ such that $p(a|x) = g_p(a|f_p(x))$ for all actions a and contexts x. That is, $\mathcal{K}(\mathcal{F})$ considers only those probability kernels that depend on the context x through some some model in $\mathcal{F}$. We discuss this in more detail in Section 2.5.

The average misspecification need not be known, though our regret bounds stated in Section 2.4 will depend on it.

2.1. Regression Oracle

We use a regression algorithm as a subroutine on the class of outcome models $\mathcal{F}$, and our exploration depends on the estimation rates of this subroutine. Suppose $\hat{f}$ is the output of the regression algorithm fitted on n independently and identically drawn samples from $D(p)$, it is then reasonable to expect that for any $\zeta \in (0,1)$, the following holds with probability $1 - \zeta$:

$$\mathop{\mathbb{E}}_{x \sim D_{\mathcal{X}}} \mathop{\mathbb{E}}_{a \sim p(\cdot|x)} [(\hat{f}(x,a) - f^*(x,a))^2] \\ \leq \min_{f \in \mathcal{F}} \mathop{\mathbb{E}}_{x \sim D_{\mathcal{X}}} \mathop{\mathbb{E}}_{a \sim p(\cdot|x)} [(f(x,a) - f^*(x,a))^2] + \xi(n, \zeta). \tag{5}$$

[5]Similar measures of misspecification are denoted by ϵ in other papers.

We call $\xi(\cdot,\cdot)$ the estimation rate of the regression algorithm and assume that it is known. We also require it to satisfy two benign conditions, and say it is a "valid" estimation rate if it satisfies these conditions. First, we require ξ to be a decreasing function of n. In particular, we require:[6]

$$\begin{aligned}&\text{For all } \delta \in (0,1),\\ &\xi(n, \delta/\ln(n)) \text{ is non-increasing in } n.\end{aligned} \tag{6}$$

The second condition is that this estimation rate is lower bounded by the rate for estimating the mean of a one-dimensional bounded random variable:[7]

$$\begin{aligned}&\text{For all } \zeta \in (0,1) \text{ and } n \in \mathbb{N},\\ &\xi(n,\zeta) \geq \ln(1/\zeta)/n.\end{aligned} \tag{7}$$

We restate these general requirements of the regression algorithm as Assumption 1 in Section 2.4. Finally, for concreteness, note that any estimation rate of the following form is valid:

$$\xi(n,\zeta) = \begin{cases} \frac{C \ln^{\rho'}(n) \ln(1/\zeta) \mathbf{comp}(\mathcal{F})}{n^{\rho}}, & \text{for } n \geq n_0. \\ 1. & \text{otherwise.} \end{cases} \tag{8}$$

where $C > 0$, $\rho \in (0,1]$, $\rho' \in [0,\infty)$, $\mathbf{comp}(\mathcal{F})$ is an appropriate measure of the complexity of the outcome model class $\mathcal{F}$, and $n_0 \in \mathbb{N}$ is an appropriately chosen constant that ensures (6) holds. Many statistical rates have this form (see e.g., Koltchinskii, 2011), indicating that our conditions on the regression algorithm are relatively benign.

2.2. Algorithm

In this section we outline the Safe-FALCON algorithm. A formal description is deferred to Algorithm 1 below.

As the name suggests, our method is based on the FALCON+ algorithm in (Simchi-Levi & Xu, 2020, Algorithm 2). FALCON+ is computationally tractable and, when the model is well-specified (i.e., when $f^* \in \mathcal{F}$), attains optimal regret bounds on cumulative regret (2). However, as we saw in the introduction, under misspecification its behavior can be erratic.

Safe-FALCON is implemented in epochs indexed by m. Where epoch m starts at round $\tau_{m-1}+1$ and ends at round τ_m, $\tau_{m+1} = 2\tau_m$ for all $m \geq 1$, $\tau_0 = 0$, and $\tau_1 \geq 2$ is an input to the algorithm. Each round t starts with a status that is called "safe" or "not safe", depending on whether the algorithm has detected evidence of model misspecification using a test that will be described shortly. The algorithm's behavior depends on this status, and once it switches to "not safe" status, it never returns to "safe". Let's describe each of these behaviors.

Status-dependent behavior At the beginning of any epoch m that starts on a "safe" round, the algorithm uses an estimate of the reward model $\hat{f}_m$, obtained using data from epoch $m-1$ from an offline regression oracle. As long as the "safe" status is maintained, at each round in this epoch it assigns actions by drawing from the following distribution, named the *action selection kernel*:

$$p_m(a|x) := \begin{cases} \frac{1}{K+\gamma_m\left(\hat{f}_m(x,\hat{a}) - \hat{f}_m(x,a)\right)} & \text{for } a \neq \hat{a}, \\ 1 - \sum_{a' \neq \hat{a}} p(a'|x) & \text{for } a = \hat{a}. \end{cases} \tag{9}$$

where $\hat{a} = \max_a \hat{f}_m(a,x)$ is the predicted best action. The parameter $\gamma_m > 0$ governs how much the algorithm exploits and explores: assignment probabilities concentrate on the predicted best policy $\hat{a}$ when γ_m is large, and are more spread out when γ_m is small. During "safe" epochs, the speed at which γ_m increases is inversely proportional to the square-root of the estimation rate of the regression algorithm:

$$\gamma_m := \sqrt{1/8}\sqrt{K/\xi(\tau_{m-1}-\tau_{m-2}, \delta'/m^2)} \tag{10}$$

where ξ is the estimation rate of the regression oracle defined in (5), $\delta > 0$ is a confidence parameter, $\delta' = C\delta$ for a universal constant $C > 0$, the quantity $\tau_{m-1} - \tau_{m-2}$ is the size of the previous batch, which was used to estimate the model $\hat{f}_m$. Definition (10) implies that small classes such as linear models allow for a quickly increasing γ_m and therefore more exploitation, while large classes require more exploration and therefore γ_m increases more slowly. Finally, we let S_m denote the data collected in epoch m, and note that $\hat{f}_{m+1}$ is the output of the regression oracle with S_m as input.

Now suppose misspecification is detected at round t and the algorithm enters "not safe" status. For all epochs $m < m(t)$, we compute a high-probability lower bound l'_m around the expected reward of the policy that selects actions according to the kernel p_m, and select the action selection kernel associated with the epoch corresponding to the highest lower bound $\hat{m} = \arg\max_{m \leq m(t)} l'_m$, where

$$l'_m := \frac{1}{|S_m|} \sum_{(x,a,r) \in S_m} r - \sqrt{\frac{1}{2|S_m|} \ln\left(\frac{m^2}{\delta'}\right)}. \tag{11}$$

Thereafter, all actions will be selected according the the action selection kernel $p_{\hat{m}}(x|a)$.

Testing for misspecification Denoting the beginning of the m-th epoch by $\tau_{m-1}+1$, the algorithm tests for misspecification at the end of round $\tau_{m-1}+1$, $\tau_{m-1}+2$, $\tau_{m-1}+4$,

[6] We require the first condition to ensure that γ_m is a increasing function of m, see Section 2.2.

[7] The second condition is more for notational convenience as (5) will always hold with a larger ξ. Further, in most scenarios, one would not expect a rate smaller than the one for estimating the mean of a one-dimensional bounded random variable.

and so on, up to and including τ_m. Suppose round t is one of these time-steps in epoch m where the algorithm tests for misspecification. The test starts by constructing a loose high-probability lower bound on the expected reward of the optimal policy, $l_{m-1} = \max_{m' \le m-1} l'_{m'}$.[8] The test consists of checking whether cumulative rewards $\sum_{i=1}^{t} r_i(a_i)$ remain above some lower bound L_t, defined as

$$L_t := t \cdot l_{m-1} - \tau_1 - \sqrt{2t \ln\left(\frac{\lceil m + \log_2(\tau_1)\rceil^3}{\delta'}\right)} - 20.3\sqrt{K}\sum_{i=\tau_2}^{t}\sqrt{\xi\left(\tau_{m(i)-1} - \tau_{m(i)-2}, \frac{\delta'}{(m(i))^2}\right)}. \tag{12}$$

Once we detect that cumulative reward dip below this bound, the algorithm switches to "not safe" status forever.

Algorithm 1 Safe-FALCON

input: Initial epoch length $\tau_1 \ge 2$, confidence parameter $\delta \in (0,1)$.

Set $\tau_0 = 0$, and $\tau_{m+1} = 2\tau_m$ for all $m \ge 1$.
Let $\hat{f}_1 \equiv 0$, $l_0 = 0$, $\text{Crwd}_0 = 0$, **safe** = **True**, and $\hat{m} = 0$.
for epoch $m = 1, 2, \dots$ **do**
  Let γ_m be given by (10). (for epoch 1, $\gamma_1 = 1$.)
  for round $t = \tau_{m-1} + 1, \dots, \tau_m$ **do**
    Observe context x_t.
    if safe then
      Let p_m be given by (9).
      Sample $a_t \sim p_m(\cdot|x_t)$, and observe $r_t(a_t)$.
      Let $\text{Crwd}_t \leftarrow \text{Crwd}_{t-1} + r_t(a_t)$.
      if $m \ge 2$ **then**
        safe $\leftarrow$ Check-is-safe$(m, t, l_{m-1}, \text{Crwd}_t)$.
      end if
    else
      Sample $a_t \sim p_{\hat{m}}(\cdot|x_t)$.
    end if
  end for
  if safe then
    Let $S_m := \{(x_t, a_t, r_t(a_t))\}_{t=\tau_{m-1}+1}^{\tau_m}$, be the data collected in epoch m.
    Let $(l_m, \hat{m}) \leftarrow$ Choose-safe(m, S_m, l_{m-1}).
    Let $\hat{f}_{m+1}$ be the output of the regression algorithm with S_m as input.
  end if
end for

[8] By construction, l'_m is a high probability lower bound on the expected reward of the randomized policy used in epoch m. Hence, $l_m = \max_{m' \le m} l'_{m'}$ is a high probability lower bound on the expected reward of some (possibly randomized) policy. Therefore, l_m is also a high probability lower bound on the expected reward of the optimal policy.

Algorithm 2 Check-is-safe

input: Epoch m, time-step t, lower bound l_{m-1}, and Crwd_t.

Let L_t be given by (12).
if $\log_2(t - \tau_{m-1}) \in \{0, 1, 2, \cdots\}$ or $t = \tau_m$ **then**
  if $\text{Crwd}_t \ge L_t$ **then**
    safe $\leftarrow$ **True**.
  else
    safe $\leftarrow$ **False**.
  end if
end if
Return **safe**.

Algorithm 3 Choose-safe

input: Epoch m, lower bound l_{m-1}, and data collected in the m-th epoch S_m.

Let

$$l'_m = \frac{1}{|S_m|}\sum_{(x,a,r)\in S_m} r - \sqrt{\frac{1}{2|S_m|}\ln\left(\frac{m^2}{\delta}\right)}.$$

Let $l_m = \max(l_{m-1}, l'_m)$.
if $l_m \ne l_{m-1}$ **then**
  Update $\hat{m} \leftarrow m$.
end if
Return $(l_m, \hat{m})$.

2.3. Understanding Safe-FALCON

In this section we try to understand Safe-FALCON and simultaneously sketch a proof for our main cumulative regret bound (Theorem 1). Theorem 1 provides the following bound on cumulative regret:

$$R_T \le \mathcal{O}\left(\sqrt{KB}\,T + \sqrt{K}\sum_{t=\tau_1+1}^{T}\sqrt{\xi\left(\tau_{m(t)-1} - \tau_{m(t)-2}, \frac{\delta}{(m(t))^2}\right)}\right). \tag{13}$$

The first term in (13) is the regret overhead due to misspecification, and the second term is the regret bound for FALCON+ assuming realizability holds. We also briefly note that all expectations in this section are taken over the randomness in the environment and algorithm being used.

To understand the intuition behind the algorithm, consider the following epoch,

$$m^* := \max\left\{m \mid B \le \xi\left(\tau_m - \tau_{m-1}, \frac{\delta'}{m^2}\right)\right\}. \tag{14}$$

We show that, with high probability, the status at the end of

epoch m^*+1 is "safe". Moreover, up to the end of epoch m^*+1, our upper bound on the expected instantaneous regret decreases at the same rate as when realizability holds. The proof of this fact follows by making a "simple" observation that allows us to extend the analysis of (Simchi-Levi & Xu, 2020) to bound cumulative regret in these early epochs. In particular, if $m(t) \leq m^*+1$, we get that with high probability the expected cumulative regret up to time t is upper bounded by:

$$\mathbb{E}\left[\sum_{i=1}^{t}(r_i(\pi^*(x_i)) - r_i(a_i))\right] \leq \tau_1 + 20.3\sqrt{K}\sum_{i=\tau_2}^{t}\sqrt{\xi\Big(\tau_{m(i)-1}-\tau_{m(i)-2}, \frac{\delta'}{(m(i))^2}\Big)}, \tag{15}$$

which are the bounded attained by FALCON+ under realizability. After epoch m^*+1, the expected instantaneous regret may increase. However, we show that the lower bound we construct for the expected reward of the policy that selects according to the action selection kernel p_{m^*+1} is sufficiently close to the expected reward of the optimal policy:

$$\mathbb{E}[r_t(\pi^*(x_t))] - l'_{m^*+1} \leq \mathcal{O}(\sqrt{KB}). \tag{16}$$

Hence, if we knew m^*, by switching the algorithm's status to "not safe" at the end of epoch m^*+1 would give us the required bounds on cumulative regret (Theorem 1). Unfortunately, we do not know the value of m^*. So, we try to detect if our current epoch m is larger than m^*+1 by looking for unexpected jumps in cumulative regret.[9] Recall that from the construction of l_{m-1} described in Section 2.2, we have have that, l_{m-1} is a weak high-probability lower bound on the expected reward of the optimal policy ($\mathbb{E}[r_t(\pi^*(x_t))]$). Therefore, from (15) we get that when $m(t) \leq m^*+1$, the expected cumulative reward up to time t should be lower bounded by:

$$\mathbb{E}\left[\sum_{i=1}^{t} r_i(a_i)\right] \geq t\cdot l_{m-1} - \tau_1 - 20.3\sqrt{K}\sum_{i=\tau_2}^{t}\sqrt{\xi\Big(\tau_{m(i)-1}-\tau_{m(i)-2}, \frac{\delta'}{(m(i))^2}\Big)}. \tag{17}$$

Now, from standard concentration arguments and (17), we get that with high probability, the cumulative reward up to time t must be lower bounded by L_t if $m(t) \leq m^*+1$. That is, with high-probability, our test claims that $m(t) > m^*+1$ only if it is true. This completes the proof sketch for the validity of the misspecification test. Hence, by design, we get that the status at the end of epoch m^*+1 is safe with high probability.

Finally consider the case when $m(t) > m^*+1$, but the misspecification test was not violated. That is $m(t) > m^*+1$, but the cumulative reward up to time t is lower bounded by L_t. By our algorithm design, since $m(t) > m^*+1$, we get that:

$$l_{m(t)-1} \geq l_{m^*+1} \geq l'_{m^*+1}. \tag{18}$$

Combining (16) and (18), gives us that with high-probability, the lower bound $l_{m(t)-1}$ is close to the expected reward of the optimal policy:

$$\mathbb{E}[r_t(\pi^*(x_t))] - l_{m(t)-1} \leq \mathcal{O}(\sqrt{KB}). \tag{19}$$

Combining (19) and the fact that cumulative reward up to time t is lower bounded by L_t, we get the required bound on cumulative regret (13).[10]

As we argued earlier, with high-probability, the algorithm's status switch only happens after epoch m^*+1. From (16), we get that the instantaneous regret after the status switch is sufficiently small to give us the required bound on the cumulative regret (13). This completes the proof sketch for Theorem 1 and also explains our algorithmic choices in Safe-FALCON.

2.4. Main result

The performance of our algorithm will depend on known estimation rates of the regression algorithm. As discussed in Section 2.1, we require the regression algorithm used in Safe-FALCON to satisfy Assumption 1 described below.

Assumption 1. *Suppose that the regression algorithm used on the class of outcome model $\mathcal{F}$ satisfies the following property. For any probability kernel $p \in \mathcal{K}(\mathcal{F})$, any natural number n, and any $\zeta \in (0,1)$, the following holds with probability at least $1-\zeta$:*

$$\mathop{\mathbb{E}}_{x\sim D_{\mathcal{X}}}\mathop{\mathbb{E}}_{a\sim p(\cdot|x)}[(\hat{f}(x,a) - f^*(x,a))^2] \leq B + \xi(n,\zeta). \tag{20}$$

and where $\hat{f}$ is the output of the regression algorithm fitted on n independently and identically drawn samples from $D(p)$ as input. Here $B > 0$ is a (possibly unknown) constant. The function $\xi : \mathbb{N}\times[0,1] \to [0,\infty)$ is a known, "valid" rate; i.e., it satisfies (6) and (7).[11]

[9] Note that in the realizable case, m^* is ∞, therefore trying to detect if $m > m^*+1$ can also be considered as a test for misspecification as it is also testing the finiteness of m^*.

[10] We use (19) to lower bound L_t in terms of the expected optimal reward. We then use standard concentration inequalities to further lower bound this in terms of the cumulative reward of the optimal policy. Since L_t itself is a lower bound on the cumulative reward up to time t, we get the required bound on cumulative regret (13).

[11] For regression algorithms that satisfy (5), we get that the constant B used in assumption 1 is given by (3).

Theorem 1 (Main result). *Suppose the regression algorithm used in Safe-FALCON satisfies Assumption 1. Then with probability at least $1-\delta$, Safe-FALCON attains the following regret guarantee:*

$$R_T \leq \mathcal{O}\Bigg(\sqrt{KB}\,T + \sqrt{K}\sum_{t=\tau_1+1}^{T}\sqrt{\xi\Big(\tau_{m(t)-1}-\tau_{m(t)-2},\frac{\delta}{(m(t))^2}\Big)}\Bigg). \tag{21}$$

The above regret typically has the same rate as $\mathcal{O}(\sqrt{K\xi(T,\delta/\log(T))}T+\sqrt{KBT})$. In particular, when the estimation rates in assumption 1 are of the form (8), we get the regret bound given by Corollary 1.

Corollary 1. *Suppose the regression algorithm used in Safe-FALCON satisfies Assumption 1 with estimation rate of the form given by (8). Then with probability at least $1-\delta$, Safe-FALCON attains the following regret guarantee:*

$$R_T \leq \mathcal{O}\Bigg(\sqrt{KB}\,T + \sqrt{KT^{2-\rho}\ln^{\rho'}(T)\ln\Big(\frac{\ln(T)}{\delta}\Big)\boldsymbol{comp}(\mathcal{F})}\Bigg). \tag{22}$$

Note that Theorem 1 provides a bias-variance trade-off for contextual bandits. The first term in (21) (regret overhead due to misspecification) depends on B, which is a tight upper bound on the average squared bias for the best estimator in the model class $\mathcal{F}$ under the distribution induced by any probability kernel in $\mathcal{K}(\mathcal{F})$. The second term in (21) (regret bound under realizability) depends on the estimation rate $\xi(\cdot,\cdot)$, which captures the variance of the regression oracle estimate over the class $\mathcal{F}$. For more expressive model classes, the bias term B is small, but the variance term $\xi(\cdot,\cdot)$ is large, showing that there is a bias-variance trade-off for contextual bandits that rely on some model class $\mathcal{F}$. A better dependency on the variance term cannot be expected even when realizability holds (see e.g. Foster & Rakhlin, 2020). In Theorem 2, we show that one cannot get a better dependency on the bias term either by providing a $\Omega(\sqrt{KB})$ lower bound on the regret overhead due to misspecification for contextual bandits that use regression oracles or rely on a model class $\mathcal{F}$.

2.5. Lower bound

We prove a new lower bound on the regret overhead due to misspecification for the stochastic contextual bandit setting in terms of the average misspecification error $\sqrt{B}$, where B is defined in (3).

The issue of model misspecification is specific to contextual bandit algorithms that use regression oracles or rely on some model class $\mathcal{F}$. To state a lower bound on the regret overhead due to misspecification, it is helpful to understand the common characteristics of such algorithms. We argue that the set $\mathcal{K}(\mathcal{F})$ is central to many contextual bandit algorithms based on regression oracles. In particular, we will argue that at every time-step t such algorithms choose some probability kernel $\tilde{p}_t$ in the convex hull of $\mathcal{K}(\mathcal{F})$, receive a context x_t, and sample an action $\tilde{a}_t$ from $\tilde{p}_t(\cdot|x_t)$.

For example, at every time-step, the algorithms used in (Foster & Rakhlin, 2020), (Simchi-Levi & Xu, 2020), and this work use probability kernels of the form defined in (9), which are in $\mathcal{K}(\mathcal{F})$. Similarly, parametric Thompson Sampling algorithms (e.g., Agrawal & Goyal, 2013) select actions by following probability kernels that lie in the convex hull of $\mathcal{K}(\mathcal{F})$. This is because, in our notation, Thompson Sampling algorithms at every time-step sample a function $\tilde{f}$ from the class $\mathcal{F}$ and then follow the policy $\pi_{\tilde{f}}$, which corresponds to some kernel in $\mathcal{K}(\mathcal{F})$. The same is true for greedy and epsilon-greedy algorithms that select actions based on a regression oracle since uniform sampling does not depend on contexts and since the greedy policy π_f corresponds to some probability kernel in $\mathcal{K}(\mathcal{F})$.

While algorithms based on upper confidence bounds may not use policies that correspond to kernels in the convex hull of $\mathcal{K}(\mathcal{F})$, we informally note that these algorithms are asymptotically greedy and hence converge to policies that correspond to kernels in $\mathcal{K}(\mathcal{F})$.[12]

Theorem 2 shows that there is a family of stochastic contextual bandit instances such that, for any probability kernel in the convex hull of $\mathcal{K}(\mathcal{F})$, the expected instantaneous regret of the induced randomized policy can be lower bounded by $\Omega(\sqrt{KB})$. Hence on these instances, any algorithm that plays randomized policies induced by probability kernels in the convex hull of $\mathcal{K}(\mathcal{F})$ for at least a constant fraction of time-steps has expected cumulative regret lower bounded by $\Omega(\sqrt{KB}\cdot T)$.

Theorem 2 (Lower bound). *Consider any $K \geq 2$ and $B \in [0, 1/(2K)]$. One can construct a model class $\mathcal{F}$ and a stochastic contextual bandit instance with K arms. Such that the average misspecification error is $\sqrt{B}$. And for any probability kernel p in the convex hull of the kernel set $\mathcal{K}(\mathcal{F})$, the expected instantaneous regret of the induced randomized policy can be lower bounded by:*

$$\mathop{\mathbb{E}}_{(x,r)\sim D}\ \mathop{\mathbb{E}}_{a\sim p(\cdot|x)}\big[r(\pi^*(x))-r(a)\big] \geq \Omega(\sqrt{KB}) \tag{23}$$

An immediate implication of Theorem 2 is that the regret

[12] UCB algorithms rely on confidence estimates. It wasn't clear to us what the general form of these confidence estimates should be and how they would relate to $\mathcal{F}$.

overhead due to misspecification for most contextual bandit algorithms that use regression oracles can be lower bounded by $\Omega(\sqrt{KB}\cdot T)$. Hence showing that the regret upper bound (Theorem 1) ensured by Safe-FALCON is optimal.

2.6. Improving Safe-FALCON and a Simulation

In Section 2.2, we discussed a misspecification test (Check-is-safe) that checks if the cumulative reward remains above a lower bound L_t. At every round where we verify this condition, we can similarly check if the average per-epoch reward remains above a lower bound (see (24)). Similar to the argument used in Section 2.3, one can show that (24) holds with high-probability if $m(t) \leq m^* + 1$. Hence adding this test to Check-is-safe can only make Safe-FALCON more robust, ensuring Theorem 1 continues to hold.

$$\begin{aligned} &\frac{1}{t-\tau_{m(t)-1}} \sum_{i=\tau_{m(t)-1}}^{t} r_i(a_i) \geq l_{m(t)-1} \\ &- 20.3\sqrt{K}\sqrt{\xi\left(\tau_{m(t)-1}-\tau_{m(t)-2}, \frac{\delta'}{(m(t))^2}\right)} \\ &- \sqrt{\frac{2}{t-\tau_{m(t)-1}} \ln\left(\frac{\lceil m(t)+\log_2(\tau_1)\rceil^3}{\delta'}\right)} \end{aligned} \tag{24}$$

Further improvements to Check-is-safe can be made by constructing better high-probability lower bounds (l_{m-1}) on the expected reward of the optimal policy. One approach to constructing such bounds would be to use offline policy evaluation methods to construct a lower bound on the expected reward of a policy that is estimated to be optimal. We do not pursue this here.

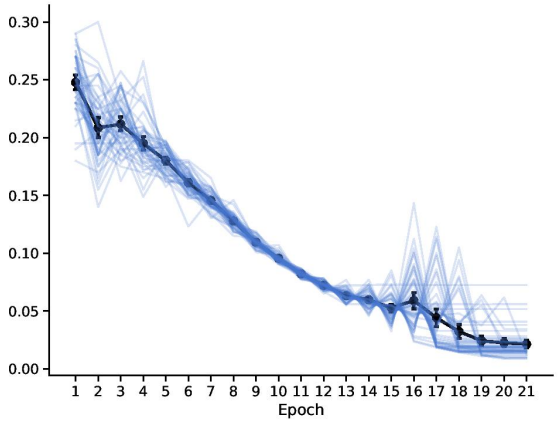

Figure 2. Illustrating that linear Safe-FALCON does not fail on Example (1). Each epoch starts at round 2^m. Vertical bars are 95% confidence intervals around the average per-epoch average regret, aggregated over 50 simulations.

To complete our discussion from Section 1, we simulate a version of linear Safe-FALCON on Example (1). In particular, we implement a version of Safe-FALCON that uses two misspecification tests, a test that checks if the cumulative reward remains above a lower bound (line 3 of Check-is-safe) and a test that checks if the average per-epoch reward remains above a lower bound (24). Other parameters are chosen as in the introduction example (see Appendix D for details). The results are shown in Figure 2.

Despite this example not being linearly realizable, in contrast to `FALCON+` (see Figure 1), average per-epoch regret under Safe-FALCON does not increase in later periods (see Figure 2). Safe-FALCON detects misspecification sometime after epoch 12 and defaults to the action selection kernel used in epoch $\hat{m}$ thereafter. For each simulation, the selected safe policy is fixed and attains constant regret, which explains the horizontal lines seen on the right side of the graph in Figure 2. An interesting direction for future improvement is to develop algorithms that continue adaptive experimentation after epoch $\hat{m}$.

3. Discussion

In this work, we presented a contextual bandit algorithm that is computationally tractable, flexible, and supports general-purpose function approximation. The ideas used here are relatively simple and allow us to provide a reduction from contextual bandits to offline regression without assuming realizability. We do this by modifying the `FALCON+` algorithm, allowing us to inherit the optimal guarantees of (Simchi-Levi & Xu, 2020) when realizability holds. When realizability doesn't hold, we get an optimal bound on the regret overhead due to misspecification in terms of the average misspecification error. We provide both upper (Theorem 1) and lower (Theorem 2) bounds on regret, allowing us to quantify the bias-variance trade-off for contextual bandit algorithms based on regression oracles.

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

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

Agrawal, S. and Goyal, N. Thompson sampling for contextual bandits with linear payoffs. In *International Confer-*

ence on Machine Learning, pp. 127–135. PMLR, 2013.

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

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

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

Foster, D. J., Gentile, C., Mohri, M., and Zimmert, J. Adapting to misspecification in contextual bandits. *Advances in Neural Information Processing Systems*, 33, 2020a.

Foster, D. J., Rakhlin, A., Simchi-Levi, D., and Xu, Y. Instance-dependent complexity of contextual bandits and reinforcement learning: A disagreement-based perspective. *arXiv preprint arXiv:2010.03104*, 2020b.

Ghosh, A., Chowdhury, S. R., and Gopalan, A. Misspecified linear bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

Koltchinskii, V. *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, 2011.

Krishnamurthy, S. K., Hadad, V., and Athey, S. Tractable contextual bandits beyond realizability. *arXiv preprint arXiv:2010.13013*, 2020.

Lattimore, T. and Szepesvári, C. *Bandit algorithms*. Cambridge University Press, 2020.

Lattimore, T., Szepesvari, C., and Weisz, G. Learning with good feature representations in bandits and in rl with a generative model. In *International Conference on Machine Learning*, pp. 5662–5670. PMLR, 2020.

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

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

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

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

Xu, Y. and Zeevi, A. Upper counterfactual confidence bounds: a new optimism principle for contextual bandits. *arXiv preprint arXiv:2007.07876*, 2020.