Practical Contextual Bandits with Regression Oracles

Dylan J. Foster [1] **Alekh Agarwal** [2] **Miroslav Dudík** [2] **Haipeng Luo** [3] **Robert E. Schapire** [2]

Abstract

A major challenge in contextual bandits is to design general-purpose algorithms that are both practically useful and theoretically well-founded. We present a new technique that has the empirical and computational advantages of realizability-based approaches combined with the flexibility of agnostic methods. Our algorithms leverage the availability of a regression oracle for the value-function class, a more realistic and reasonable oracle than the classification oracles over policies typically assumed by agnostic methods. Our approach generalizes both UCB and LinUCB to far more expressive possible model classes and achieves low regret under certain distributional assumptions. In an extensive empirical evaluation, we find that our approach typically matches or outperforms both realizability-based and agnostic baselines.

1. Introduction

We study the design of practically useful, theoretically well-founded, general-purpose algorithms for the contextual bandits (CBs) problem. In this setting, the learner repeatedly receives *context*, then selects an *action*, resulting in a received *reward*. The aim is to learn a *policy*, a mapping from contexts to actions, to maximize the long-term cumulative reward. For instance, a news portal must repeatedly choose articles to present to each user to maximize clicks. Here, the context is information about the user, the actions are the articles, and the reward might be indicator of a click. We refer the reader to an ICML 2017 tutorial (`http://hunch.net/~rwil/`) for further examples.

CB algorithms can be put into two groups. Some methods (Langford & Zhang, 2008; Agarwal et al., 2014) are *agnostic* in the sense that they are provably effective for *any* given policy class and data distribution. In contrast, *realizability-based* approaches such as LinUCB and variants (Chu et al., 2011; Li et al., 2017; Filippi et al., 2010) or Thompson sampling (Thompson, 1933) assume the data is generated from a particular parametrized family of models. Computationally tractable realizability-based algorithms are only known for specific model families, such as when the conditional reward distributions come from a generalized linear model.

The two groups of approaches seem to have different advantages and disadvantages. Empirically, in the contextual semibandit setting, Krishnamurthy et al. (2016) found that the realizability-based LinUCB approach outperforms all agnostic baselines using a linear policy class. However, the agnostic approaches were able to overcome this shortcoming by using a more powerful policy class. Computationally, previous realizability-based approaches have been limited by their reliance on either closed-form confidence bounds (as in LinUCB variants), or the ability to efficiently sample from and frequently update the posterior (as in Thompson sampling). Agnostic approaches, on the other hand, typically assume an oracle for cost-sensitive classification, which is computationally intractable in the worst case, but often practically feasible for many natural policy classes.

In this paper, we aim to develop techniques that combine the best of both of these approaches. To this end, in Section 3, we propose computationally efficient and practical realizability-based algorithms for arbitrary model classes. As is often done in agnostic approaches, we assume the availability of an oracle which reduces to a standard learning setting and knows how to efficiently leverage the structure of the model class. Specifically, we require access to a least-squares regression oracle over the model class that we use for predicting rewards given contexts. Since regression can often be solved efficiently, the availability of such an oracle is a far more reasonable assumption than the cost-sensitive classification oracle usually assumed, which typically must solve NP-hard problems. In fact, for this reason, even the classification oracles are typically approximated by regression oracles in practice (see, e.g., Beygelzimer & Langford, 2009). Our main algorithmic components here are motivated by and adapted from a recent work of Krishnamurthy et al. (2017) on cost-sensitive active learning.

[1]Cornell University. Work performed while the author was an intern at Microsoft Research. [2]Microsoft Research [3]University of Southern California. Correspondence to: Dylan J. Foster <djf244@cornell.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

In Section 4, we prove that our algorithms are effective in achieving low regret under certain distributional assumptions. Specifically, we show that our methods enjoy low regret so long as certain quantities like the *disagreement coefficient* (Hanneke, 2014; Krishnamurthy et al., 2017) are bounded, or when some other distributional coefficients inspired by Bastani & Bayati (2015) are well-behaved. As a special consequence, we obtain nearly dimension-free results for sparse linear bandits in high dimensions.

Finally, in Section 5, we conduct a very extensive empirical evaluation of our algorithms on a number of datasets and against both realizability-based and agnostic baselines. In this test of practical effectiveness, we find that our approach gives comparable or superior results in nearly all cases, and we also validate the distributional assumptions required for low-regret guarantees on these datasets.

2. Preliminaries

We consider the following contextual bandit protocol. Contexts are drawn from an arbitrary space, $x \in \mathcal{X}$, actions are from a finite set, $a \in \mathcal{A} := \{1, \dots, K\}$, for some fixed K, and reward vectors are from a bounded set, $r \in [0,1]^K$, with component $r(a)$ denoting the reward for action $a \in \mathcal{A}$.

We consider an i.i.d. setting where there is a fixed and unknown distribution D over the context-reward pairs (x, r), with $D_{\mathcal{X}}$ denoting its marginal over $\mathcal{X}$. At each round $t = 1, 2, \dots, T$, nature samples (x_t, r_t) according to D and reveals x_t to the learner. The learner chooses an action $a_t \in \mathcal{A}$ and observes the reward $r_t(a_t)$. The learner aims to maximize its reward and compete with strategies that model the expected reward $\mathbb{E}[r(a) \mid x, a]$ via functions $f : \mathcal{X} \times \mathcal{A} \to [0,1]$. We consider mappings f from a given class $\mathcal{F}$, such as linear predictors or regression trees.

The main assumption this paper follows is that the class $\mathcal{F}$ is rich enough to contain a predictor that perfectly predicts the expected reward of any action under any context, that is:

Assumption 1 (Realizability)**.** There is a predictor $f^\star \in \mathcal{F}$ such that $\mathbb{E}[r(a) \mid x, a] = f^\star(x, a) \;\; \forall x \in \mathcal{X},\; a \in \mathcal{A}$.

This assumption is used by essentially all regression-based contextual bandit algorithms (Chu et al., 2011; Filippi et al., 2010; Russo & Van Roy, 2013; Li et al., 2017).

Given a predictor $f \in \mathcal{F}$, the associated optimal strategy $\pi_f : \mathcal{X} \to \mathcal{A}$, called a *policy*, picks the action with the highest predicted reward, i.e., $\pi_f(x) := \arg\max_{a \in \mathcal{A}} f(x, a)$ (ties broken arbitrarily). Using $\pi^\star := \pi_{f^\star}$ to denote an optimal policy, the learner aims to minimize its regret

$$\mathrm{Reg}_T = \textstyle\sum_{t=1}^T r_t(\pi^\star(x_t)) - \sum_{t=1}^T r_t(a_t),$$

which compares the accumulated rewards between the optimal policy and the learner's strategy. The classic Exp4 algorithm (Auer et al., 2002b) achieves an optimal regret bound of order $O(\sqrt{TK \ln |\mathcal{F}|})$ (for any finite $\mathcal{F}$), but the computational complexity is unfortunately linear in $|\mathcal{F}|$.

Regression Oracle To overcome the computational obstacle, our algorithms reduce the contextual bandit problem to weighted least-squares regression. Abstracting the computational complexity, we assume access to a *weighted least-squares regression oracle* over the predictor class $\mathcal{F}$, which takes any set H of weighted examples $(w, x, a, y) \in \mathbb{R}_+ \times \mathcal{X} \times \mathcal{A} \times [0,1]$ as input, and outputs the predictor with the smallest weighted squared loss:

$$\textsc{Oracle}(H) = \arg\min_{f \in \mathcal{F}} \textstyle\sum_{(w,x,a,y) \in H} w(f(x,a) - y)^2.$$

As mentioned, such regression tasks are very common in machine learning practice and the availability of such oracle is thus a very mild assumption.

3. Algorithms

The high-level idea of our algorithms is the following. As data is collected, we maintain a subset of $\mathcal{F}$, referred to as the *version space*, that only contains $f \in \mathcal{F}$ with small squared loss on observed data. For a new example, we construct a confidence interval for the expected reward of each action based on this version space. Finally, with these confidence intervals, we either optimistically pick the action with the highest upper bound, similar to UCB and LinUCB, or randomize among all actions that are potentially the best.

The challenge here is to maintain such version spaces and compute upper and lower confidence bounds efficiently, and we show that this can be done using a binary search together with a small number of regression oracle calls.

More formally, we define the upper and lower bounds on the expected reward with respect to a subset $\mathcal{F}' \subseteq \mathcal{F}$ as

$$\textsc{High}_{\mathcal{F}'}(x,a) = \max_{f \in \mathcal{F}'} f(x,a), \;\; \textsc{Low}_{\mathcal{F}'}(x,a) = \min_{f \in \mathcal{F}'} f(x,a).$$

Our algorithms will induce the confidence bounds by instantiating these quantities using the version space as $\mathcal{F}'$. To reduce computation, our algorithms update on a doubling epoch schedule. There are $M = O(\log T)$ epochs and each epoch m begins at time $\tau_m = 2^{m-1}$. At epoch m our algorithms (implicitly) construct a version space $\mathcal{F}_m \subseteq \mathcal{F}$, and then select an action based on the reward ranges defined by $\textsc{High}_{\mathcal{F}_m}(x,a)$ and $\textsc{Low}_{\mathcal{F}_m}(x,a)$ for each time t that falls into epoch m. Specifically, we consider two algorithm variants: the first one uniformly at random picks from actions that are plausible to be the best (see lines 6-7 in Algorithm 1); the second one simply behaves optimistically and picks the action with the highest upper bound (see line 9 in Algorithm 2). For technical reasons, the optimistic variant also performs pure exploration in the first few epochs to warm-start the algorithm.

Algorithm 1 REGCB.ELIMINATION

Input: square-loss tolerance β_m
for epoch $m = 1, \dots, M$ **do**
  $\mathcal{F}_m \leftarrow \begin{cases} \prod_{a \in \mathcal{A}} \widehat{\mathcal{G}}_m(\beta_m, a) & \text{(OPTION I)} \\ \widehat{\mathcal{F}}_m(\beta_m) & \text{(OPTION II)} \end{cases}$
  for time $t = \tau_m, \dots, \tau_{m+1} - 1$ **do**
    Receive x_t and define A_t as:
    $\{a : \textsc{High}_{\mathcal{F}_m}(x_t, a) \geq \max_{a' \in \mathcal{A}} \textsc{Low}_{\mathcal{F}_m}(x_t, a')\}$.
    Sample $a_t \sim \textsc{Unif}(A_t)$ and receive $r_t(a_t)$.
  end for
end for

To construct these version spaces, we further introduce the following least-squares notation for any $m \geq 2$:

- $\widehat{R}_m(f) = \frac{1}{\tau_m - 1} \sum_{s < \tau_m} \big(f(x_s, a_s) - r_s(a_s)\big)^2$,
- $\widehat{\mathcal{F}}_m(\beta) = \big\{f \in \mathcal{F} \,\big|\, \widehat{R}_m(f) - \min_{f \in \mathcal{F}} \widehat{R}_m(f) \leq \beta\big\}$,

and also let $\widehat{\mathcal{F}}_1(\beta) = \mathcal{F}$ for any β. With this notation $\mathcal{F}_m$ is simply set to $\widehat{\mathcal{F}}(\beta_m)$ for some β_m, and $\textsc{High}_{\mathcal{F}_m}$ and $\textsc{Low}_{\mathcal{F}_m}$ recover the confidence bounds in UCB (Auer et al., 2002a) and LinUCB (Chu et al., 2011) for appropriate β_m.

Product Classes Sometimes it is desirable to have a product predictor class, that is, $\mathcal{F} = \mathcal{G}^{\mathcal{A}}$, where $\mathcal{G} : \mathcal{X} \to [0, 1]$ is a "base class" and each $f \in \mathcal{F}$, described by a K-tuple $(g_a)_{a \in \mathcal{A}}$ where $g_a \in \mathcal{G}$, predicts according to $f(x, a) = g_a(x)$. Similar to the general case, we define:

- $\widehat{\mathcal{R}}_m(g, a) = \frac{1}{\tau_m - 1} \sum_{s < \tau_m} (g(x_s) - r_s(a_s))^2 \mathbf{1}\{a_s = a\}$,
- $\widehat{\mathcal{G}}_m(\beta, a) = \big\{g \in \mathcal{G} \mid \widehat{\mathcal{R}}_m(g, a) - \min_{g \in \mathcal{G}} \widehat{\mathcal{R}}_m(g, a) \leq \beta\big\}$,

and let $\widehat{\mathcal{G}}_1(\beta, a) = \mathcal{G}$ for any β. In this case we construct $\mathcal{F}_m$ as $\prod_{a \in \mathcal{A}} \widehat{\mathcal{G}}_m(\beta_m, a)$ for some tolerance parameter β_m.

Our two procedures are described in Algorithms 1 and 2.

3.1. Efficient Reward-Range Computation

Algorithms 1 and 2 hinge on the computation of the bounds $\textsc{High}_{\mathcal{F}_m}$ and $\textsc{Low}_{\mathcal{F}_m}$. This can be carried out efficiently via a small number of calls to the regression oracle.

Specifically, to calculate the confidence bounds for a given pair (x, a), we augment the data set H_m with a single example (x, a, r) with a weight w. For the upper bound $\textsc{High}_{\mathcal{F}_m}$ we use $r = 2$; for the lower bound $r = -1$ (these values are chosen as they lie outside the reward range). By changing the weight w, we trade-off the loss on this single example against that on the history H_m. The binary search over w identifies—up to a given precision—the weight w at which

Algorithm 2 REGCB.OPTIMISTIC

Input: square-loss tolerance β_m
  number of warm-start epochs M_0
for time $t = 1, \dots, \tau_{M_0} - 1$ **do**
  Receive x_t, play $a_t \sim \textsc{Unif}(\mathcal{A})$, and receive $r_t(a_t)$.
end for
for epoch $m = M_0, \dots, M$ **do**
  $\mathcal{F}_m \leftarrow \widehat{\mathcal{F}}_m(\beta_m)$.
  for time $t = \tau_m, \dots, \tau_{m+1} - 1$ **do**
    Receive x_t.
    Select $a_t = \arg\max_{a \in \mathcal{A}} \textsc{High}_{\mathcal{F}_m}(x_t, a)$.
    Receive $r_t(a_t)$.
  end for
end for

the empirical regret on H_m is exactly the desired tolerance β, with the corresponding prediction on x, a yielding $\textsc{High}_{\widehat{\mathcal{F}}_m(\beta)}(x, a)$ or $\textsc{Low}_{\widehat{\mathcal{F}}_m(\beta)}(x, a)$ (see Algorithm 3).

In Appendix A.1 we prove that this strategy works as intended and in $O(\log(1/\alpha))$ iterations computes the confidence bounds up to a precision of α.

Theorem 1. *Let $H_m = \{(x_s, a_s, r_s(a_s))\}_{s=1}^{\tau_m - 1}$. If the function class $\mathcal{F}$ is convex and closed under pointwise convergence, then the calls*

$$z_{\textsc{High}} \leftarrow \textsc{BinSearch}(\textsc{High}, (x, a), H_m, \beta, \alpha)$$
$$z_{\textsc{Low}} \leftarrow \textsc{BinSearch}(\textsc{Low}, (x, a), H_m, \beta, \alpha)$$

terminate after $O(\log(1/\alpha))$ oracle invocations and

$$\left|\textsc{High}_{\widehat{\mathcal{F}}_m(\beta)}(x, a) - z_{\textsc{High}}\right| \leq \alpha,$$
$$\left|\textsc{Low}_{\widehat{\mathcal{F}}_m(\beta)}(x, a) - z_{\textsc{Low}}\right| \leq \alpha.$$

Compared to the procedure from Krishnamurthy et al. (2017), Algorithm 3 is much simpler and achieves an exponential improvement in terms of oracle calls, namely $O(\log(1/\alpha))$ as opposed to $O(1/\alpha)$, when $\mathcal{F}$ is convex. Compared to oracles for cost-sensitive classification, convexity is not a strong assumption for regression oracles. When $\mathcal{F}$ is not convex, reward bounds can be computed in $O(1/\alpha)$ oracle calls (see Krishnamurthy et al. 2017).

4. Regret Guarantees

In this section we provide regret guarantees for RegCB (Algorithm 1 and Algorithm 2). Note that RegCB is not minimax optimal: while it can obtain $O\big(\sqrt{KT \log|\mathcal{F}|}\big)$ regret or even logarithmic regret under certain distributional assumptions, which we describe shortly, for some instances it can make as many as $|\mathcal{F}|$ mistakes, which is suboptimal:

Proposition 1. *For every $\epsilon \in (0, 1]$ and $N \in \mathbb{N}$ there exists a class of reward predictors satisfying Assumption 1 with*

Algorithm 3 BINSEARCH

Input: bound type $\in \{\text{LOW}, \text{HIGH}\}$, target pair (x, a) history H, radius $\beta > 0$, precision $\alpha > 0$
Based on bound type: $r \leftarrow 2$ if HIGH and $r \leftarrow -1$ if LOW
Let $R(f) \coloneqq \sum_{(x',a',r') \in H} (f(x', a') - r')^2$
Let $\widetilde{R}(f, w) \coloneqq R(f) + \frac{w}{2}(f(x, a) - r)^2$
$w_{\text{L}} \leftarrow 0,\ w_{\text{H}} \leftarrow \beta/\alpha$
`// Invoke oracle twice`
$f_{\text{L}} \leftarrow \arg\min_{f \in \mathcal{F}} \widetilde{R}(f, w_{\text{L}}),\ z_{\text{L}} \leftarrow f_{\text{L}}(x, a)$
$f_{\text{H}} \leftarrow \arg\min_{f \in \mathcal{F}} \widetilde{R}(f, w_{\text{H}}),\ z_{\text{H}} \leftarrow f_{\text{H}}(x, a)$
$R_{\min} \leftarrow R(f_{\text{L}})$
$\Delta \leftarrow \alpha\beta/(r - z_{\text{L}})^3$
while $|z_{\text{H}} - z_{\text{L}}| > \alpha$ and $|w_{\text{H}} - w_{\text{L}}| > \Delta$ **do**
  $w \leftarrow (w_{\text{H}} + w_{\text{L}})/2$
  `// Invoke oracle.`
  $f \leftarrow \arg\min_{\tilde{f} \in \mathcal{F}} \widetilde{R}(\tilde{f}, w),\ z \leftarrow f(x, a)$
  if $R(f) \geq R_{\min} + \beta$ **then**
    $w_{\text{H}} \leftarrow w,\ z_{\text{H}} \leftarrow z$
  else
    $w_{\text{L}} \leftarrow w,\ z_{\text{L}} \leftarrow z$
  end if
end while
return z_{H}.

$|\mathcal{F}| = N + 1$ and a distribution for which both Algorithms 1 and 2 have regret lower bounded by $(1-\epsilon) \cdot \min\{N, \widetilde{\Omega}(T)\}$.

Proposition 1 is proved in Appendix A.2. The proof builds on a well-known albeit rather pathological instance. In contrast, our strong empirical results in the following section show that such instances are not encountered in practice. In order to understand the typical behavior of such algorithms, prior works have considered structural assumptions such as finite eluder dimension (Russo & Van Roy, 2013) or disagreement coefficients (Hanneke, 2014; Krishnamurthy et al., 2017). In the next two subsections, we use similar ideas to analyze the regret incurred by our algorithm. We assume that $\text{HIGH}_{\mathcal{F}_m}$ and $\text{LOW}_{\mathcal{F}_m}$ are computed exactly, but extension to the approximate case is straightforward.

4.1. Disagreement-based Analysis

Disagreement coefficients come from the active learning literature (Hanneke, 2014), and roughly assume that given a set of functions which fit the historical data well, the probability that these functions make differing predictions on a new example is small. This rules out the bad case of Proposition 1, where a near-optimal predictor significantly disagrees from the others on each context. Our development in this subsection largely follows Krishnamurthy et al. (2017), with appropriate modifications to translate from active learning to contextual bandits. We begin with a formal definition of the disagreement coefficient.

Definition 1 (Disagreement Coefficient)**.** *The disagreement coefficient for $\mathcal{F}$ (with respect to $D_{\mathcal{X}}$) is defined as*

$$\theta_0 \coloneqq \sup_{\delta > 0, \varepsilon > 0} \frac{\delta}{\varepsilon} \Pr_{D_{\mathcal{X}}}\Big[x \in \text{Dis}(\mathcal{F}(\varepsilon)) \text{ and } \exists a \in A_{\mathcal{F}(\varepsilon)}(x) : W_{\mathcal{F}(\varepsilon)}(x, a) > \delta\Big].$$

Here $\mathcal{F}(\varepsilon)$ is the set of all predictors f whose greedy policies have regret at most ε, $\text{Dis}(\mathcal{F}(\varepsilon))$ is the set of x's where the greedy policies of at least two functions in $\mathcal{F}(\varepsilon)$ choose different actions, $A_{\mathcal{F}}(x) = \bigcup_{f \in \mathcal{F}} \big(\arg\max_{a \in \mathcal{A}} f(x, a)\big)$, and $W_{\mathcal{F}}(x, a)$ is the difference between the upper and lower bounds $\text{HIGH}_{\mathcal{F}}(x, a) - \text{LOW}_{\mathcal{F}}(x, a)$. Formal definitions of these quantities can be found in Appendix A.3. Informally, the disagreement coefficient is small if on most contexts either all $f \in \mathcal{F}(\varepsilon)$ choose the same action according to their greedy policies or all actions chosen by those policies have a low range of predicted rewards.

The following theorem provides regret bounds in terms of the disagreement coefficient. In all theorems we use $\widetilde{O}$ to suppress polynomial terms in $\log T$, $\log K$, and $\log(1/\delta)$, where δ is the failure probability. Moreover, all results can be improved to be logarithmic (in T) under the standard Massart noise condition (see the appendix for the details).

Theorem 2. *With* $\beta_m = \frac{(M-m+1)C_\delta}{\tau_m - 1}$ *and* $C_\delta = 16 \log\left(\frac{2|\mathcal{G}|KT^2}{\delta}\right)$*, Algorithm 1 with Option I incurs* $\text{Reg}_T = \widetilde{O}\left(T^{\frac{3}{4}} (\log|\mathcal{G}|)^{\frac{1}{4}} \sqrt{\theta_0 K}\right)$ *with probability at least* $1 - \delta$.

See Theorem 5 in Appendix A.3 for the full version of this theorem, which applies to infinite classes and additionally obtains faster rates under the Massart noise condition.

Discussion Theorem 2 critically uses the product class structure, specifically the fact that the set A_t computed by the algorithm coincides with the disagreement set $A_{\mathcal{F}_m}(x_t)$ for $t \in \{\tau_m, \ldots, \tau_{m+1} - 1\}$. This is true for product classes, but not necessarily for general (non-product) predictor classes. Computing the disagreement set efficiently for non-product classes is a challenge for future work.

While bounding the disagreement coefficients *a priori* often requires strong assumptions on the model class and the distribution, the size of disagreement set can be easily checked empirically under the product class assumption, and we include this diagnostic in our experimental results.

Finally, while the disagreement coefficient enables the analysis of Algorithm 1, it is not obvious how to use it to analyze Algorithm 2. Our analysis crucially requires that any plausibly optimal action a be chosen with a reasonable probability, something which the optimistic algorithm fails to ensure.

4.2. Moment-based Analysis

The disagreement-based analysis of Theorem 2 is not entirely satisfying, because even for linear predictors (e.g.,

as in LinUCB, Chu et al. 2011), fairly strong assumptions on $D_{\mathcal{X}}$ (e.g., log-concavity) are required to bound the disagreement coefficient θ_0 (Hanneke, 2014). To generalize the analysis to richer than linear classes without distributional assumptions on the contexts, prior work has used the notion of eluder dimension (Russo & Van Roy, 2013). It remains challenging, however, to show examples with a small eluder dimension beyond linearly parameterized functions. In addition, taking the worst-case over all histories, as in eluder dimension, is overly pessimistic for i.i.d. contextual bandits.

To address the shortcomings of both the disagreement-based analysis as well as eluder dimension for i.i.d. settings, we define two new distributional properties which we will use to analyze the regret of both of our algorithms.

Definition 2 (Surprise bound)**.** *The surprise bound $L_1 > 0$ is the smallest constant such that for all $f \in \mathcal{F}$, $x \in \mathcal{X}$, and $a \in \mathcal{A}$, the gap $(f(x,a) - f^\star(x,a))^2$ is at most*

$$L_1 \cdot \mathbb{E}_{x' \sim D_{\mathcal{X}}} \mathbb{E}_{a' \sim \mathrm{Unif}(\mathcal{A})} \Big[\big(f(x',a') - f^\star(x',a') \big)^2 \Big] \ .$$

The surprise bound is small if functions with a small expected squared error relative to $f^\star$ (under a uniform choice of actions) do not encounter a much larger squared error on any single context-action pair.

The second quantity, which we call the *implicit exploration coefficient* (or IEC) relates the expected regression error under actions chosen by the optimal policy to the worst-case error on any other context-action pair. For $\lambda \in (0,1]$ define $U_\lambda(a) = \{x \mid f^\star(x,a) \geq f^\star(x,a') + \lambda \ \forall a' \neq a\}$.

Definition 3 (Implicit exploration coefficient—IEC)**.** *For any $\lambda \in (0,1]$, the implicit exploration coefficient $L_{2,\lambda} > 0$ is the smallest constant such that for all $f \in \mathcal{F}$, $x \in \mathcal{X}$, and $a \in \mathcal{A}$, the gap $(f(x,a) - f^\star(x,a))^2$ is at most*

$$L_{2,\lambda} \, \mathbb{E}_{x' \sim D_{\mathcal{X}}} \mathbb{E}_{a' \sim \mathrm{Unif}(\mathcal{A})} \Big[\mathbf{1}\{x' \in U_\lambda(a')\} \cdot \big(f(x',a') - f^\star(x',a') \big)^2 \Big]. \tag{1}$$

We now make two remarks about these definitions and their impact on the performance of Algorithms 1 and 2.

- By definition, $L_{2,\lambda}$ is non-decreasing in λ. For Algorithm 1 we can simply use $\lambda = 0$, for which $L_{2,0}$ is defined by replacing the right-hand side of (1) with $\frac{L_{2,0}}{K} \mathbb{E}_{x \sim D_{\mathcal{X}}} [(f(x,\pi^\star(x)) - f^\star(x,\pi^\star(x)))^2]$. The analysis of Algorithm 2 requires $\lambda > 0$, and this λ must be used to tune the algorithm's warm-start period.
- We always have $L_1 \leq L_{2,0}$, but L_1 may be much smaller. L_1 appears in the regret bound of Algorithm 2, but not Algorithm 1.

We now state the regret bound for Algorithm 1 with a general class $\mathcal{F}$, and employ the shorthand $C'_\delta = 16 \log\left(\frac{2|\mathcal{F}|T^2}{\delta}\right)$.

Theorem 3. *With $\beta_m = \frac{(M-m+1)C'_\delta}{\tau_m - 1}$, Algorithm 1 with Option II incurs $\mathrm{Reg}_T = \widetilde{O}\left(\sqrt{TL_{2,0}\log|\mathcal{F}|}\right)$ with probability at least $1-\delta$.*

We now move on to describe the performance guarantee for Algorithm 2. Because this optimistic strategy does not explore as readily as the elimination-based strategy of Algorithm 1, the analysis requires both that (i) the IEC $L_{2,\lambda}$ be invoked for some $\lambda > 0$ and (ii) that the algorithm use a warm-start period whose size grows as $1/\lambda^2$.

Theorem 4. *With $\beta_m = \frac{(M-m+1)C'_\delta}{\tau_m - 1}$ and $M_0 = 2 + \left\lfloor \log_2\left(1 + \frac{(2M+3)L_1 C'_\delta}{\lambda^2}\right)\right\rfloor$ for any $\lambda \in (0,1)$, Algorithm 2 incurs $\mathrm{Reg}_T = \widetilde{O}\left(\frac{L_1 \log|\mathcal{F}|}{\lambda^2} + \sqrt{TL_{2,\lambda}\log|\mathcal{F}|}\right)$ with probability at least $1-\delta$.*

As Algorithm 2 requires a warm start, the regret bounds of Theorem 4 are always worse than those of Theorem 3. Appendix A.4 contains full versions of these theorems, Theorem 6 and Theorem 7, which obtain faster rates under the Massart noise condition and apply to infinite classes.

Linear classes For concreteness, let us discuss the regret of both algorithms in a linear setting with a fixed feature map $\phi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^d$ and $\mathcal{F} = \{(x,a) \mapsto w^\top \phi(x,a) \mid w \in \mathcal{W}\}$ for some $\mathcal{W} \subseteq \mathbb{R}^d$ (e.g., as in LinUCB). In the basic ℓ_2-bounded case, L_1 and $L_{2,\lambda}$ can be bounded in terms of the minimum eigenvalues of $\mathbb{E}_x[\phi(x,a)\phi(x,a)^\top]$ and $\mathbb{E}_x\big[\mathbf{1}\{x \in U_\lambda(a)\}\,\phi(x,a)\phi(x,a)^\top\big]$, respectively. When predictors are s-sparse we can instead obtain bounds in terms of $\psi(A) \coloneqq \min_{w \neq 0:\, \|w\|_0 \leq 2s} w^\top A w \,/\, w^\top w$, the minimum restricted eigenvalue for $2s$-sparse predictors (Raskutti et al., 2010). For Algorithm 1 this yields a near dimension-independent bound on Reg_T of

$$\widetilde{O}\left(s\sqrt{KT\log d \,/\, \psi\big(\mathbb{E}_x[\phi(x,\pi^\star(x))\phi(x,\pi^\star(x))^\top]\big)}\right).^{1}$$

This improves upon the moment matrix conditions of Bastani & Bayati (2015), although our algorithm requires non-convex optimization oracles.[2] Note that without the scaling with K as in our result, a $\sqrt{d}$ dependence is unavoidable (Abbasi-Yadkori et al., 2012). The result highlights the strengths of our analysis in the best case compared with eluder dimension, which does not adapt to sparsity structures. On the other hand, for the standard LinUCB setting, our result is inferior by at least a factor of K.

Discussion Our analysis is influenced by the results of Bastani & Bayati (2015) for the (high-dimensional) linear setting, but extends to general classes $\mathcal{F}$, and when applied to Algorithm 1 with linear classes, the assumed bound on

[1] See Proposition 3, Lemma 9, and Theorem 3 in the appendix.

[2] Also, since the class $\mathcal{F}$ is non-convex, this requires the slower binary search algorithm of Krishnamurthy et al. (2017).

$L_{2,\lambda}$ is weaker than their "diversity condition". Similar assumptions have been used to analyze purely greedy linear contextual bandits (Bastani et al., 2017; Kannan et al., 2018); our assumptions are strictly weaker.

5. Experiments

We compared our new algorithms with existing oracle-based alternatives. In addition to showing that RegCB[3] has strong empirical performance, our experiments provide a more extensive empirical study of oracle-based contextual bandit algorithms than any past works (e.g., Agarwal et al., 2014, Krishnamurthy et al., 2016). Description of the datasets, benchmark algorithms, and oracle configurations, as well as further experimental results are included in Appendix B.

Datasets We begin with 10 datasets with full reward information and simulate bandit feedback by withholding the rewards for actions not selected by the algorithm. We use two large-scale learning-to-rank datasets, Microsoft MSLR-WEB30k (`mslr`) (Qin & Liu, 2010) and Yahoo! Learning to Rank Challenge V2.0 (`yahoo`) (Chapelle & Chang, 2011), which were previously used to evaluate contextual semibandits (Krishnamurthy et al., 2016). We also use eight classification datasets from the UCI repository (Lichman, 2013), summarized in Table 1 of Appendix B.1.

The ranking datasets have natural rewards (relevances), but the rewards for the classification datasets always have multiclass structure (1 for the correct action and 0 for all others). To ensure that we evaluate the full generality of the CB setting, we create eight "noisy" UCI datasets by sampling new rewards for the datasets according to a noisy reward matrix model described in Appendix B. This yields additional 8 datasets for a total of 18. On each dataset we consider several replicates obtained by randomly permuting examples and, on noisy UCI, also randomly generating rewards. All the methods are evaluated on the same set of replicates.

Algorithms We evaluate Algorithms 1 and 2 against three baselines, all based on various optimization-oracle assumptions. First two are agnostic baselines, ϵ-Greedy (Langford & Zhang, 2008) and the minimax-optimal ILOVETOCONBANDITS (ILTCB) strategy of Agarwal et al. (2014).[4]

ϵ-Greedy and ILTCB both assume cost-sensitive classification oracles and come with theoretical guarantees. The third baseline is a bootstrapping-based exploration strategy of Dimakopoulou et al. (2017) (Bootstrap-TS), which uses bootstrapping to estimate confidence intervals and then performs Thompson sampling to select an action based on the intervals. This algorithm represents a computationally tractable alternative to Thompson sampling as it works in the regression-oracle model we consider here, but it does not have a theoretical analysis.[5]

Note that the LinUCB algorithm (Chu et al., 2011; Abbasi-Yadkori et al., 2011), which is a natural baseline as well, coincides with our Algorithm 2 (with a linear oracle), so we only plot the performance of RegCB with a linear oracle.

All of the algorithms update on an epoch schedule with epoch lengths of $2^{i/2}$, which is a theoretically rigorous choice for each algorithm.

Oracles We consider two baseline predictor classes $\mathcal{F}$: ℓ_2-regularized linear functions (Linear) and gradient-boosted depth-5 regression trees (GB5) (Friedman, 2001). For the regularized linear class, Algorithm 2 is equivalent to LinUCB on an epoch schedule.[6] See Appendix B.3 for details.

When running both RegCB variants with the GB5 oracle, we use a simple heuristic to substantially speed up the computation. At the beginning of each epoch m, we find the best regression-tree ensemble on the dataset so far (i.e., with respect to $\widehat{R}_m$). Throughout the epoch, we keep the structure of the ensemble fixed and in each call to $\textsc{Oracle}(H)$ we only re-optimize the predictions in leaves. This can be solved in closed form, similar to LinUCB, so the full binary search procedure (Algorithm 3) does not need to be run.

Parameter Tuning We evaluate each algorithm for eight exponentially spaced parameter values across five replicates. For ϵ-Greedy we tune the constant ϵ, and for ILTCB we tune a certain smoothing parameter (see Appendix B). For Algorithms 1 and 2 we set $\beta_m = \beta$ for all m and tune β. For Algorithm 2 we use a warm start of 0. We tune a confidence parameter similar to β for Bootstrap-TS.

Evaluation Each dataset is split into "training data", for which algorithm receives one example at a time and must predict online, and a holdout validation set. Validation is performed by simulating the algorithm's predictions on examples from the holdout set without allowing the algorithm to incorporate these examples. We also plot the validation reward of a "supervised" baseline obtained by training the oracle (either Linear or GB5) on the entire training set at once (including rewards for all actions).

For Algorithms 1 and 2 we show average reward at various numbers of training examples for the best fixed parameter value in each dataset. For the baselines, we take the *pointwise maximum of the average reward across all parameter values* for each number of examples. Thus,

[3] RegCB refers collectively to both Algorithms 1 and 2.

[4] We use an implementation available at `https://github.com/akshaykr/oracle_cb`, which was also used by Krishnamurthy et al. (2016).

[5] It is not known how to implement the standard formulation of Thompson sampling for contextual bandits (e.g., Russo & Van Roy 2013) with optimization oracles.

[6] More precisely, it is equivalent to the well-known OFUL variant of LinUCB (Abbasi-Yadkori et al., 2011).

the curves for our methods correspond to an actual run of the algorithm, while the baselines are an upper envelope aggregating multiple parameter values.

Results: Performance Figure 1 shows average reward of each algorithm on a holdout validation set for three representative datasets, `letter` from UCI, `letter-noise` (the variant with simulated rewards), and `yahoo`.

RegCB (both Algorithms 1 and 2) outperforms all baselines on the unmodified UCI datasets (e.g., `letter` in Figure 1). On the noisy variants (e.g., `letter+N` in Figure 1), the performance of the ILTCB and Bootstrap-TS benchmarks improves significantly, with Bootstrap-TS slightly edging out the rest of the algorithms. On the `yahoo` ranking dataset (Figure 1, right), the ordering of the algorithms in performance is similar to noisy UCI datasets.

Validation performance plots for all datasets are in Appendix B. Overall, RegCB methods and Bootstrap-TS generally dominate the field. While Bootstrap-TS can outperform RegCB methods when using GB5 models, the gap is typically quite small. For linear models, RegCB methods generally outperform Bootstrap-TS, hinting that the approximate binary search might be hurting RegCB with GB5 models. We also observe that when RegCB methods outperform Bootstrap-TS, the gap is often quite large. We will see further evidence of this behavior in the next set of results.

Results: Aggregate Performance To rigorously draw conclusions about overall performance, Figure 2 aggregates performance across all datasets. We compute "normalized relative loss" for each algorithm by rescaling the validation reward (computed as in Figure 1) so that, at each round, the best performing algorithm has loss 0 and the worst has loss 1. In each plot of Figure 2 we consider normalized relative losses at a specific cutoff time (1000 examples in the left plot, and all examples in the center and right), and for each method we plot the number of datasets where it achieves loss below a threshold, as a function of the threshold. Thus, curves towards top left corner correspond to methods that achieve lower relative loss on more datasets. The intercept at loss 0 is the number of datasets where an algorithm is the best, and the intercept at 0.99 is the number of datasets where the it is not the worst (so the distance from top is the number of datasets where it is the worst). Solid lines are runs with GB5 and dashed lines are with the Linear oracle.

The aggregate performance with the GB5 oracle across all datasets can be briefly summarized as follows: RegCB always beats ϵ-Greedy and ILTCB, but sometimes loses out to Bootstrap-TS, and Bootstrap-TS itself sometimes underperforms relative to the other baselines, especially on the UCI datasets. Even when RegCB is not the best, it is almost always within 20% of the best. The elimination and optimistic variants of RegCB have comparable performance, with elimination performing slightly better in aggregate.

The RegCB algorithms with the GB5 oracle also dominate the ϵ-Greedy, ILTCB, and Bootstrap-TS baselines when they are equipped with Linear oracles (the dashed lines in Figure 2). When the RegCB algorithms use the Linear oracle they also dominate the baselines with the Linear oracle across all datasets, including Bootstrap-TS.[7] This suggests that the gap between RegCB and Bootstrap-TS for GB5 may be due to the approximation of fixing the ensemble structure in each epoch, as noted earlier.

Results: Confidence Width The analysis of RegCB relies on assumptions on D (disagreement coefficient or moment parameters) that are not easy to verify. The main role of these parameters is to control the rate at which confidence width $W_{\mathcal{F}_m}(x_t, a) = \textsc{High}_{\mathcal{F}_m}(x_t, a) - \textsc{Low}_{\mathcal{F}_m}(x_t, a)$ used in RegCB shrinks, since small widths imply that the algorithm makes good decisions and thus has low regret.

To investigate whether the width indeed shrinks empirically, we compute $W_{\mathcal{F}_m}(x_t, a)$ on each dataset for Algorithm 2 and Bootstrap-TS, where a is the "optimistic" action with highest upper confidence bound under each algorithm. Finally for both Algorithm 2 and Bootstrap-TS we compute the size of the "disagreement set" A_t, defined in Algorithm 1, which measures how many actions the algorithm thinks are plausibly best.[8]

Figure 3 shows width and disagreement for a representative sample of datasets under the GB5 oracle; the remaining datasets are in Appendix B. The figure suggests that our distributional assumptions are reasonable for real-world datasets. In particular, for our algorithm, the width decays roughly as $T^{-1/3}$ for `letter` and $T^{-1/2}$ for `letter+N` and `yahoo`. Interestingly, the best hyper-parameter setting for Bootstrap-TS on `letter` yields low but essentially constant (i.e., not shrinking) width, and obtains a poor validation reward in Figure 1 (left). This suggests that while the Bootstrap-TS confidence intervals are small, they may not be faithful in the sense of containing $f^\star(x, a)$.

6. Conclusion and Discussion

This work serves as a starting point for what we hope will be a fruitful line of research on oracle-efficient contextual bandit algorithms in realizability-based settings. We have shown that the RegCB family of algorithms have strong empirical performance and enjoy nice theoretical properties.

[7] The aggregate plots for RegCB with the Linear oracle can be found in Appendix B along with additional aggregate plots.

[8] This set is well-defined for both RegCB-Opt and Bootstrap-TS even through neither algorithm instantiates it explicitly. For the `yahoo` and `mslr` datasets this $|A_t|$ is technically a lower bound on the true disagreement set size $|A_{\mathcal{F}_m}(x_t)|$ because our classes $\mathcal{F}$ do not have product structure on these datasets—see Section 4.1.

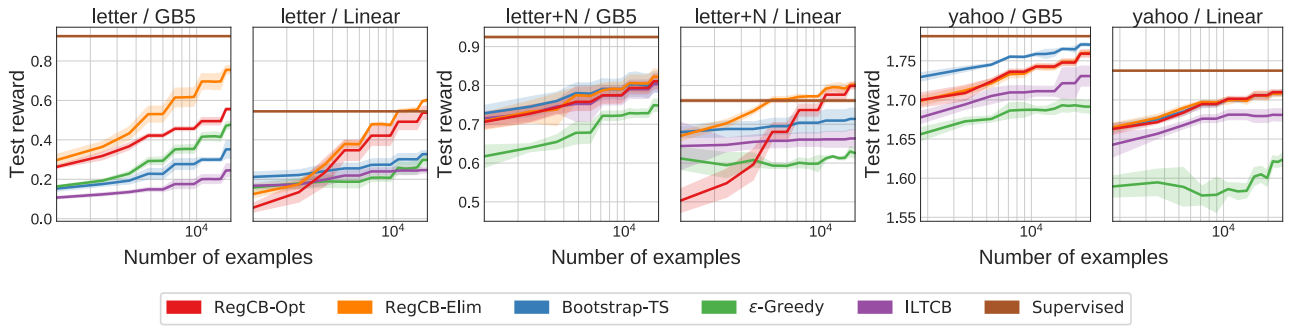

Figure 1. Validation performance for three representative datasets as a function of the number of rounds t of interaction. The number of rounds t is on a log scale. For each dataset, we show separately the performance with the GB5 oracle and the Linear oracle.

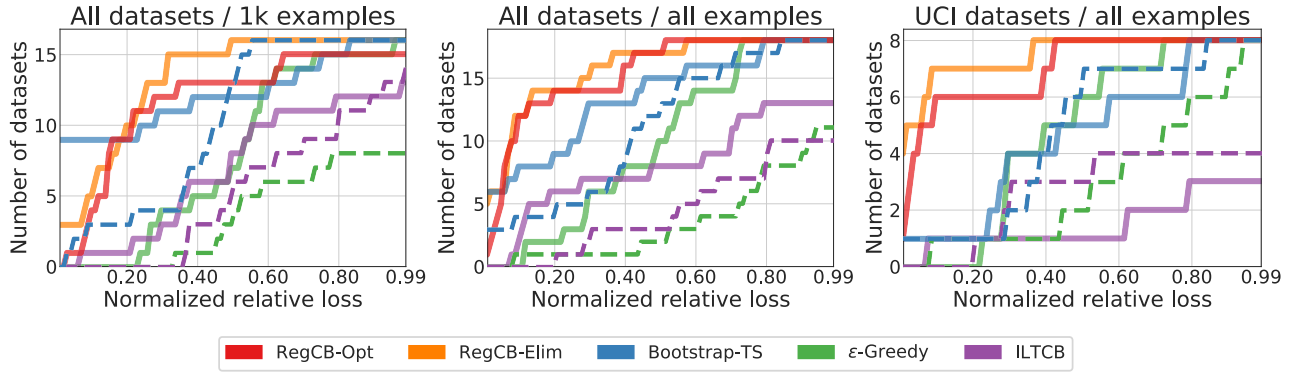

Figure 2. Aggregate performance across all datasets, at various sample sizes; solid lines — GB5 oracle; dashed lines — Linear oracle. Left: All datasets (the UCI datasets, their noisy variants, and the Microsoft and Yahoo ranking datasets) at 1 000 examples (datasets with fewer examples dropped). Center: All datasets at their final round. Right: Unmodified UCI at their final round.

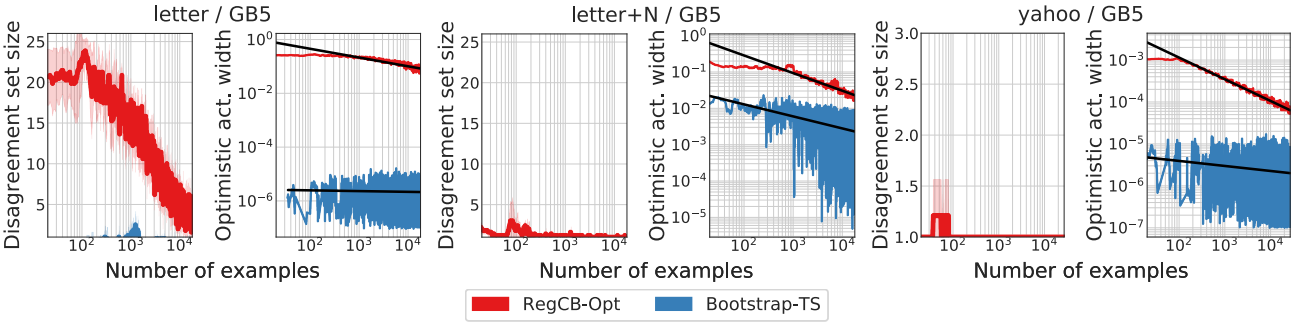

Figure 3. For each dataset, disagreement set size as a function of number of rounds t (with t on a log scale), and the log-log plot of the width of the optimistic action (the action chosen by Algorithm 2) as a function of t. Plots are averaged using a sliding window of length 20. Black lines on the width plots are best linear fits, whose slopes give the rate of the width decay as follows: letter/Bootstrap-TS: -0.05, letter/RegCB: -0.34, letter-noise/Bootstrap-TS: -0.33, letter-noise/RegCB: -0.51, yahoo/Bootstrap-TS: -0.26, yahoo/RegCB: -0.52.

These results suggest several compelling future directions.

First, is there a regression oracle–based algorithm that achieves the optimal $\widetilde{O}(\sqrt{KT\log|\mathcal{F}|})$ regret? For example, is it possible to oraclize regressor elimination of Agarwal et al. (2012)?

Second, given the competitive empirical performance of Bootstrap-TS, are there reasonable assumptions as in Section 4 under which it can be analyzed? There is recent work in this direction for linear models (Lu & Van Roy, 2017).

Finally, randomizing uniformly or putting all the mass on the optimistic choice are two extreme cases of choosing amongst the plausibly optimal actions. Are there better randomization schemes that lead to stronger regret guarantees?

Acknowledgements

We thank Akshay Krishnamurthy and Alberto Bietti for helpful discussions.

References

Abbasi-Yadkori, Y., Pál, D., and Szepesvári, C. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pp. 2312–2320, 2011.

Abbasi-Yadkori, Y., Pal, D., and Szepesvari, C. Online-to-confidence-set conversions and application to sparse stochastic bandits. In *Artificial Intelligence and Statistics*, pp. 1–9, 2012.

Agarwal, A., Dudík, M., Kale, S., Langford, J., and Schapire, R. E. Contextual bandit learning with predictable rewards. In *International Conference on Artificial Intelligence and Statistics*, pp. 19–26, 2012.

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.

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002a.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002b.

Bastani, H. and Bayati, M. Online decision-making with high-dimensional covariates. 2015.

Bastani, H., Bayati, M., and Khosravi, K. Exploiting the natural exploration in contextual bandits. *arXiv preprint arXiv:1704.09011*, 2017.

Beygelzimer, A. and Langford, J. The offset tree for learning with partial labels. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 129–138. ACM, 2009.

Chapelle, O. and Chang, Y. Yahoo! learning to rank challenge overview. In *Proceedings of the Learning to Rank Challenge*, pp. 1–24, 2011.

Chu, W., Li, L., Reyzin, L., and Schapire, R. E. Contextual bandits with linear payoff functions. In *International Conference on Artificial Intelligence and Statistics*, pp. 208–214, 2011.

Dimakopoulou, M., Athey, S., and Imbens, G. Estimation considerations in contextual bandits. *arXiv preprint arXiv:1711.07077*, 2017.

Dudík, M., Langford, J., and Li, L. Doubly robust policy evaluation and learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pp. 1097–1104. Omnipress, 2011.

Filippi, S., Cappe, O., Garivier, A., and Szepesvári, C. Parametric bandits: The generalized linear case. In *Advances in Neural Information Processing Systems*, pp. 586–594, 2010.

Friedman, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Hanneke, S. Theory of disagreement-based active learning. *Foundations and Trends® in Machine Learning*, 7(2-3): 131–309, 2014.

Kannan, S., Morgenstern, J., Roth, A., Waggoner, B., and Wu, Z. S. A Smoothed Analysis of the Greedy Algorithm for the Linear Contextual Bandit Problem. *ArXiv e-prints*, January 2018.

Krishnamurthy, A., Agarwal, A., and Dudik, M. Contextual semibandits via supervised learning oracles. In *Advances In Neural Information Processing Systems*, pp. 2388–2396, 2016.

Krishnamurthy, A., Agarwal, A., Huang, T.-K., Daume III, H., and Langford, J. Active learning for cost-sensitive classification. *International Conference on Machine Learning*, 2017.

Langford, J. and Zhang, T. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in neural information processing systems*, pp. 817–824, 2008.

Li, L., Lu, Y., and Zhou, D. Provable optimal algorithms for generalized linear contextual bandits. *International Conference on Machine Learning*, 2017.

Lichman, M. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Lu, X. and Van Roy, B. Ensemble sampling. In *Advances in Neural Information Processing Systems*, pp. 3260–3268, 2017.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

Qin, T. and Liu, T.-Y. Mslr: Microsoft learning to rank dataset. 2010. URL http://www.microsoft.com/en-us/research/project/mslr/.

Raskutti, G., Wainwright, M. J., and Yu, B. Restricted eigenvalue properties for correlated gaussian designs. *Journal of Machine Learning Research*, 11(Aug):2241–2259, 2010.

Rockafellar, R. T. *Convex analysis*. Princeton university press, 1970.

Russo, D. and Van Roy, B. Eluder dimension and the sample complexity of optimistic exploration. In *Advances in Neural Information Processing Systems*, pp. 2256–2264, 2013.

Thompson, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.