Proceedings of Machine Learning Research vol 75:1–15, 2018 31st Annual Conference on Learning Theory

The Externalities of Exploration and How Data Diversity Helps Exploitation*

Manish Raghavan† MANISH@CS.CORNELL.EDU
Cornell University

Aleksandrs Slivkins SLIVKINS@MICROSOFT.COM
Microsoft Research

Jennifer Wortman Vaughan JENN@MICROSOFT.COM
Microsoft Research

Zhiwei Steven Wu ZSW@UMN.EDU
Microsoft Research

Editors: Sebastien Bubeck, Vianney Perchet and Philippe Rigollet

Abstract

Online learning algorithms, widely used to power search and content optimization on the web, must balance exploration and exploitation, potentially sacrificing the experience of current users in order to gain information that will lead to better decisions in the future. Recently, concerns have been raised about whether the process of exploration could be viewed as unfair, placing too much burden on certain individuals or groups. Motivated by these concerns, we initiate the study of the externalities of exploration—the undesirable side effects that the presence of one party may impose on another—under the linear contextual bandits model. We introduce the notion of a group externality, measuring the extent to which the presence of one population of users (the majority) impacts the rewards of another (the minority). We show that this impact can, in some cases, be negative, and that, in a certain sense, no algorithm can avoid it. We then move on to study externalities at the individual level, interpreting the act of exploration as an externality imposed on the current user of a system by future users. This drives us to ask under what conditions inherent diversity in the data makes explicit exploration unnecessary. We build on a recent line of work on the smoothed analysis of the greedy algorithm that always chooses the action that currently looks optimal. We improve on prior results to show that a greedy approach almost matches the best possible Bayesian regret rate of any other algorithm on the same problem instance whenever the diversity conditions hold, and that this regret is at most $\tilde{O}(T^{1/3})$. Returning to group-level effects, we show that under the same conditions, negative group externalities essentially vanish if one runs the greedy algorithm. Together, our results uncover a sharp contrast between the high externalities that exist in the worst case, and the ability to remove all externalities if the data is sufficiently diverse.

1. Introduction

Online learning algorithms are a key tool in web search and content optimization, adaptively learning what users want to see. In a typical application, each time a user arrives, the algorithm chooses among various content presentation options (e.g., news articles to display), the chosen content is

* Extended abstract. Full version appears as CoRR arXiv:1806.00543 v#1.

† MR is supported by an NSF Graduate Research Fellowship (DGE-1650441). Work done while at Microsoft Research.

presented to the user, and an outcome (e.g., a click) is observed. Such algorithms must balance *exploration* (making potentially suboptimal decisions now for the sake of acquiring information that will improve decisions in the future) and *exploitation* (using information collected in the past to make better decisions now). Exploration could degrade the experience of a current user, but improves user experience in the long run. This exploration-exploitation tradeoff is commonly studied in the online learning framework of *multi-armed bandits* (Bubeck and Cesa-Bianchi, 2012).

Concerns have been raised about whether exploration in such scenarios could be unfair, in the sense that some individuals or groups may experience too much of the downside of exploration without sufficient upside (Bird et al., 2016). We formally study these concerns in the *linear contextual bandits* model (Li et al., 2010; Chu et al., 2011), a standard variant of multi-armed bandits appropriate for content personalization scenarios. We focus on *externalities* arising due to exploration, that is, undesirable side effects that the presence of one party may impose on another.

We first examine the effects of exploration at a group level. We introduce the notion of a *group externality* in an online learning system, quantifying how much the presence of one population (which we dub the majority) impacts the rewards of another (the minority). We show that this impact can be negative, and that, in a particular precise sense, no algorithm can avoid it. This cannot be explained by the absence of suitably good policies since our adoption of the linear contextual bandits framework implies the existence of a feasible policy that is simultaneously optimal for everyone. Instead, the problem is inherent to the process of exploration. We come to a surprising conclusion that more data can sometimes lead to worse outcomes for the users of an explore-exploit-based system.

We next turn to the effect of exploration at an individual level. We interpret exploration as a potential externality imposed on the current user by future users of the system. Indeed, it is only for the sake of the future users that the algorithm would forego the action that currently looks optimal. To avoid this externality, one may use the greedy algorithm that always chooses the action that appears optimal according to current estimates of the problem parameters. While this greedy algorithm performs poorly in the worst case, it tends to work well in many applications and experiments.[1]

In a new line of work, Bastani et al. (2017) and Kannan et al. (2018) analyzed conditions under which inherent diversity in the data makes explicit exploration unnecessary. Kannan et al. (2018) proved that the greedy algorithm achieves a regret rate of $\tilde{O}(\sqrt{T})$ in expectation over small perturbations of the context vectors (which ensure sufficient data diversity). This is the best rate that can be achieved in the worst case (i.e., for all problem instances, without data diversity assumptions), but it leaves open the possibilities that (i) another algorithm may perform much better than the greedy algorithm on some problem instances, or (ii) the greedy algorithm may perform much better than worst case under the diversity conditions. We expand on this line of work. We prove that under the same diversity conditions, the greedy algorithm almost matches the best possible Bayesian regret rate of *any* algorithm *on the same problem instance*. This could be as low as $\mathrm{polylog}(T)$ for some instances, and, as we prove, at most $\tilde{O}(T^{1/3})$ whenever the diversity conditions hold.

Returning to group-level effects, we show that under the same diversity conditions, the negative group externalities imposed by the majority essentially vanish if one runs the greedy algorithm. Together, our results illustrate a sharp contrast between the high individual and group externalities that exist in the worst case, and the ability to remove all externalities if the data is sufficiently diverse.

1. Both positive and negative findings are folklore. One way to precisely state the negative result is that the greedy algorithm incurs constant per-round regret with constant probability; while results of this form have likely been known for decades, Mansour et al. (2018, Corollary A.2(b)) proved this for a wide variety of scenarios. Very recently, the good empirical performance has been confirmed by state-of-art experiments in Bietti et al. (2018).

Additional motivation. Whether and when explicit exploration is necessary is an important concern in the study of the exploration-exploitation tradeoff. Fairness considerations aside, explicit exploration is expensive. It is wasteful and risky in the short term, it adds a layer of complexity to algorithm design (Langford and Zhang, 2007; Agarwal et al., 2014), and its adoption at scale tends to require substantial systems support and buy-in from management (Agarwal et al., 2016, 2017). A system based on the greedy algorithm would typically be cheaper to design and deploy.

Further, explicit exploration can run into incentive issues in applications such as recommender systems. Essentially, when it is up to the users which products or experiences to choose and the algorithm can only issue recommendations and ratings, an explore-exploit algorithm needs to provide incentives to explore for the sake of the future users (Kremer et al., 2014; Frazier et al., 2014; Che and Hörner, 2015; Mansour et al., 2015; Bimpikis et al., 2017). Such incentive guarantees tend to come at the cost of decreased performance, and rely on assumptions about human behavior. The greedy algorithm avoids this problem as it is inherently consistent with the users' incentives.

Additional related work. Our research draws inspiration from the growing body of work on fairness in machine learning (e.g., Dwork et al., 2012; Hardt et al., 2016; Kleinberg et al., 2017; Chouldechova, 2017). Several other authors have studied fairness in the context of the contextual bandits framework. Our work differs from the line of research on meritocratic fairness in online learning (Kearns et al., 2017; Liu et al., 2017; Joseph et al., 2016), which considers the allocation of limited resources such as bank loans and requires that nobody should be passed over in favor of a less qualified applicant. We study a fundamentally different scenario in which there are no allocation constraints and we would like to serve each user the best content possible. Our work also differs from that of Celis and Vishnoi (2017), who studied an alternative notion of fairness in the context of news recommendations. According to this notion, all users should have approximately the same probability of seeing a particular type of content (e.g., Republican-leaning articles), regardless of their individual preferences, in order to mitigate the possibility of discriminatory personalization.

The data diversity conditions in Kannan et al. (2018) and this paper are inspired by the smoothed analysis framework of Spielman and Teng (2004), who proved that the expected running time of the simplex algorithm is polynomial for perturbations of any initial problem instance (whereas the worst-case running time has long been known to be exponential). Such disparity implies that very bad problem instances are brittle. We find a similar disparity for the greedy algorithm in our setting.

Our results on group externalities. A typical goal in online learning is to minimize *regret*, the (expected) difference between the cumulative reward that would have been obtained had the optimal policy been followed at every round and the cumulative reward obtained by the algorithm. We define a corresponding notion of *minority regret*, the portion of the regret experienced by the minority. Since online learning algorithms update their behavior based on the history of their observations, minority regret is influenced by the entire population on which an algorithm is run. If the minority regret is much higher when a particular algorithm is run on the full population than it is when the same algorithm is run on the minority alone, we can view the majority as imposing a negative externality on the minority; the minority population would achieve a higher cumulative reward if the majority were not present. Asking whether this can ever happen amounts to asking whether access to more data points can ever lead an explore-exploit algorithm to make inferior decisions. One might think that more data should always lead to better decisions and therefore better outcomes for the users. Surprisingly, we show that this is not the case, even with a standard algorithm.

Consider LinUCB (Li et al., 2010; Chu et al., 2011; Abbasi-Yadkori et al., 2011), a standard algorithm for linear contextual bandits that is based on the principle of "optimism under uncertainty." We provide a specific problem instance on which, after observing T users, LinUCB would have a minority regret of $\Omega(\sqrt{T})$ if run on the full population, but only constant minority regret if run on the minority alone. While stylized, this example is motivated by the problem of providing driving directions to different populations of users, about which fairness concerns have been raised (Bird et al., 2016). Further, the situation is reversed on a slight variation of this example: LinUCB obtains constant minority regret when run on the full population and $\Omega(\sqrt{T})$ on the minority alone. That is, group externalities can be large and positive in some cases, and large and negative in others.

Although these regret rates are specific to LinUCB, we show that this phenomenon is, in some sense, unavoidable. Consider the minority regret of LinUCB when run on the full population and the minority regret that LinUCB would incur if run on the minority alone. We know that one may be much smaller or larger than the other. We ask whether any algorithm could achieve the minimum of the two on every problem instance. Using a variation of the same problem instance, we prove that this is impossible; in fact, no algorithm could simultaneously approximate both up to any $o(\sqrt{T})$ factor. In other words, an externality-free algorithm would sometimes "leave money on the table."

In terms of techniques, we rely on the special structure of our example, which can be viewed as an instance of the sleeping bandits problem (Kleinberg et al., 2010). This simplifies the behavior and analysis of LinUCB, allowing us to obtain the $O(1)$ upper bounds. The lower bounds are obtained using KL-divergence techniques to show that the two variants of our example are essentially indistinguishable, and an algorithm that performs well on one must obtain $\Omega(\sqrt{T})$ regret on the other.

Our results on the greedy algorithm. We consider a version of linear contextual bandits in which the latent weight vector θ is drawn from a known prior. In each round, an algorithm is presented several actions to choose from, each represented by a *context vector*. The expected reward of an action is a linear product of θ and the corresponding context vector. The tuple of context vectors is drawn independently from a fixed distribution. In the spirit of smoothed analysis, we assume that this distribution has a small amount of jitter. Formally, the tuple of context vectors is drawn from some fixed distribution, and then a small *perturbation*—small-variance Gaussian noise—is added independently to each coordinate of each context vector. This ensures arriving contexts are diverse. We are interested in Bayesian regret, i.e., regret in expectation over the Bayesian prior. Following the literature, we are primarily interested in the dependence on the time horizon T.

We focus on a batched version of the greedy algorithm, in which new data arrives to the algorithm's optimization routine in small batches, rather than every round. This is well-motivated from a practical perspective—in high-volume applications data usually arrives to the "learner" only after a substantial delay (Agarwal et al., 2016, 2017)—and is essential for our analysis.

Our main result is that the greedy algorithm matches the Bayesian regret of any algorithm up to polylogarithmic factors, for each problem instance, fixing the Bayesian prior and the context distribution. We also prove that LinUCB achieves regret $\tilde{O}(T^{1/3})$ for each realization of θ. This implies a worst-case Bayesian regret of $\tilde{O}(T^{1/3})$ for the greedy algorithm under the perturbation assumption.

Our results hold for both natural versions of the batched greedy algorithm, Bayesian and frequentist, henceforth called BatchBayesGreedy and BatchFreqGreedy. In BatchBayesGreedy, the chosen action maximizes expected reward according to the Bayesian posterior. BatchFreqGreedy estimates θ using ordinary least squares regression and chooses the best action according to this estimate. The results for BatchFreqGreedy come with additive polylogarithmic factors, but are

stronger in that the algorithm does not need to know the prior. Further, the $\tilde{O}(T^{1/3})$ regret bound for BatchFreqGreedy is approximately prior-independent, in the sense that it applies even to very concentrated priors such as independent Gaussians with standard deviation on the order of $T^{-2/3}$.

The key insight in our analysis of BatchBayesGreedy is that any (perturbed) data can be used to simulate any other data, with some discount factor. The analysis of BatchFreqGreedy requires an additional layer of complexity. We consider a hypothetical algorithm that receives the same data as BatchFreqGreedy, but chooses actions based on the Bayesian-greedy selection rule. We analyze this hypothetical algorithm using the same technique as BatchBayesGreedy, and then upper bound the difference in Bayesian regret between the hypothetical algorithm and BatchFreqGreedy.

Our analyses extend to group externalities and (Bayesian) minority regret. In particular, we circumvent the impossibility result mentioned above. We prove that both BatchBayesGreedy and BatchFreqGreedy match the Bayesian minority regret of any algorithm run on either the full population or the minority alone, up to polylogarithmic factors

Detailed comparison with prior work. We substantially improve over the $\tilde{O}(\sqrt{T})$ worst-case regret bound from Kannan et al. (2018), at the cost of some additional assumptions. First, we consider Bayesian regret, whereas their regret bound is for each realization of θ.[2] Second, they allow the context vectors to be chosen by an adversary before the perturbation is applied. Third, they extend their analysis to a somewhat more general model, in which there is a separate latent weight vector for every action (which amounts to a more restrictive model of perturbations). However, this extension relies on the greedy algorithm being initialized with a substantial amount of data. The results of Kannan et al. (2018) do not appear to have implications on group externalities.

Bastani et al. (2017) show that the greedy algorithm achieves logarithmic regret in an alternative linear contextual bandits setting that is incomparable to ours in several important ways. They consider two-action instances where the actions share a common context vector in each round, but are parameterized by different latent vectors. They ensure data diversity via a strong assumption on the context distribution. This assumption does not follow from our perturbation conditions; among other things, it implies that each action is the best action in a constant fraction of rounds. Further, they assume a version of Tsybakov's *margin condition*, which is known to substantially reduce regret rates in bandit problems (e.g., see Rigollet and Zeevi, 2010).

2. Preliminaries

We consider the model of *linear contextual bandits* (Li et al., 2010; Chu et al., 2011). Formally, there is a learner who serves a sequence of users over T rounds, where T is the (known) time horizon. For the user who arrives in round t there are (at most) K actions available, with each action $a \in \{1, \ldots, K\}$ associated with a *context vector* $x_{a,t} \in \mathbb{R}^d$. Each context vector may contain a mix of features of the action, features of the user, and features of both. We assume that the tuple of context vectors for each round t is drawn independently from a fixed distribution. The learner observes the set of contexts and selects an action a_t for the user. The user then experiences a reward r_t which is visible to the learner. We assume that the expected reward is linear in the chosen context vector. More precisely, we let $r_{a,t}$ be the reward of action a if this action is chosen in round t (so that $r_t = r_{a_t,t}$), and assume that there exists an unknown vector $\theta \in \mathbb{R}^d$ such that $\mathbb{E}\left[r_{a,t} \mid x_{a,t}\right] = \theta^\top x_{a,t}$ for any round t and action a. Throughout most of the paper, the realized rewards are either in

2. Equivalently, they allow point priors, whereas our priors must have variance $T^{-O(1)}$.

$\{0, 1\}$ or are the expectation plus independent Gaussian noise of variance at most 1. We sometimes consider a Bayesian version, in which the latent vector θ is initially drawn from some known prior $\mathcal{P}$.

A standard goal for the learner is to maximize the expected total reward over T rounds, or $\sum_{t=1}^{T} \theta^\top x_{a,t}$. This is equivalent to minimizing the learner's *regret*, defined as

$$\text{Regret}(T) = \sum_{t=1}^{T} \theta^\top x_t^* - \theta^\top x_{a_t,t} \tag{1}$$

where $x_t^* = \arg\max_{x \in \{x_{1,t}, \cdots, x_{K,t}\}} \theta^\top x$ denotes the context vector associated with the best action at round t. We are mainly interested in *expected regret*, where the expectation is taken over the context vectors, the rewards, and the algorithm's random seed, and *Bayesian regret*, where the expectation is taken over all of the above and the prior over θ.

We introduce some notation in order to describe and analyze algorithms in this model. We write x_t for $x_{a_t,t}$, the context vector chosen at time t. Let $X_t \in \mathbb{R}^{t \times d}$ be the *context matrix* at time t, a matrix whose rows are vectors $x_1, \ldots, x_t \in \mathbb{R}^d$. A $d \times d$ matrix $Z_t := \sum_{\tau=1}^{t} x_\tau x_\tau^\top = X_t^\top X_t$, called the *empirical covariance* matrix, is an important concept in some of the prior work on linear contextual bandits (e.g., Abbasi-Yadkori et al., 2011; Kannan et al., 2018), as well as in this paper.

Optimism under uncertainty. Optimism under uncertainty is a common paradigm in problems with an explore-exploit tradeoff (Bubeck and Cesa-Bianchi, 2012). The idea is to evaluate each action "optimistically"—assuming the best-case scenario for this action—and then choose an action with the best optimistic evaluation. When applied to the basic multi-armed bandit setting, it leads to a well-known algorithm called UCB1 (Auer et al., 2002), which chooses the action with the highest upper confidence bound (henceforth, UCB) on its mean reward. The UCB is computed as the sample average of the reward for this action plus a term which captures the amount of uncertainty.

Optimism under uncertainty has been extended to linear contextual bandits in the LinUCB algorithm (Chu et al., 2011; Abbasi-Yadkori et al., 2011). The high-level idea is to compute a confidence region $\Theta_t \subset \mathbb{R}^d$ in each round t such that $\theta \in \Theta_t$ with high probability, and choose an action a which maximizes the optimistic reward estimate $\sup_{\theta \in \Theta_t} x_{a,t}^\top \theta$. Concretely, one uses regression to form an empirical estimate $\hat{\theta}_t$ for θ. Concentration techniques lead to high-probability bounds of the form $|x^\top(\theta - \hat{\theta}_t)| \leq f(t)\sqrt{x^\top Z_t^{-1} x}$, where the *interval width function* $f(t)$ may depend on hyperparameters and features of the instance. LinUCB simply chooses an action

$$a_t^{LinUCB} := \arg\max_a x_{a,t}^\top \hat{\theta}_t + f(t)\sqrt{x_{a,t}^\top Z_t^{-1} x_{a,t}}. \tag{2}$$

Among other results, Abbasi-Yadkori et al. (2011) use

$$f(t) = S + \sqrt{d c_0 \log(T + tTL^2)}, \tag{3}$$

where L and S are known upper bounds on $\|x_{a,t}\|_2$ and $\|\theta\|_2$, respectively, and c_0 is a parameter. For any $c_0 \geq 1$, they obtain regret $\tilde{O}(dS\sqrt{c_0 K T})$, with only a polylog dependence on TL/d.

3. Group Externality of Exploration

In this section, we study the externalities of exploration at a group level, quantifying how much the presence of one population impacts the rewards of another in an online learning system. We consider linear contextual bandits in a setting in which there are two underlying user populations,

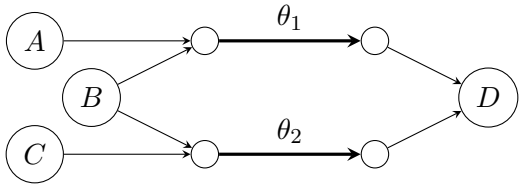

Figure 1: Visual illustration of the two-bridge instance.

called the *majority* and the *minority*. The user who arrives at round t is assumed to come from the majority population with some fixed probability and the minority population otherwise, and the population from which the user comes is known to the learner. The tuple of context vectors at time t is then drawn independently from a fixed group-specific distribution.

We assume there is a single hidden vector θ, and that the distribution of rewards conditioned on the chosen context vector is the same for both groups. Only the distribution over tuples of available context vectors differs between groups. This implies that externalities cannot be explained by the absence of a good policy, since there always exists a policy that is simultaneously optimal for everyone. This allows us to focus only on externalities inherent to the process of exploration.

We define the *minority regret* to be the regret experienced by the minority. The *group externality* imposed on the minority by the majority is then the difference between the minority regret of an algorithm run on the minority alone and the minority regret of the same algorithm run on the full population. A negative group externality implies that the minority is worse off due to the presence of the majority. It is generally more meaningful to bound the multiplicative difference between the minority regret obtained with and without the majority present. Several of our results have this form.

We first ask whether large group externalities can exist. We show that on a simple toy example, a large negative group externality arises under LinUCB, while a slight variant of this example leads to a large positive externality. Put another way, more available data can lead to either better or worse outcomes for the users of a system. We show that this general phenomenon is unavoidable. That is, no algorithm can simultaneously approximate the minority regret of LinUCB run on the full population and LinUCB run on the minority alone, up to any $o(\sqrt{T})$ multiplicative factor.

3.1. Two-Bridge Instance

We consider a toy example, motivated by a scenario in which a learner is choosing driving routes for two groups of users. Each user starts at point A, B, or C, and wants to get to the same destination, point D, which requires taking one of two bridges, as shown in Figure 1. The travel costs for each of the two bridges are unknown. For simplicity, assume all other edges are known to have 0 cost.

Suppose that 95% of users are in the majority group. All of these users start at point A and have access only to the top bridge. The other 5% are in the minority. Of these users, 95% start at point C, from which they have access only to the bottom bridge. The remaining 5% of the minority users start at point B, and have access to both bridges.

Consider the behavior of an algorithm that follows the principle of optimism under uncertainty. If run on the full user population, it will quickly collect many observations of the commute time for the top bridge since all users in the majority group must travel over the top bridge. It will collect relatively fewer observations of the commute time over the bottom bridge. Therefore, when the

algorithm is faced with a member of the minority population who starts at point B, the algorithm will likely send this user over the bottom bridge in order to collect more data and improve its estimate.

If the same algorithm is instead run on the minority alone, it will quickly collect many more observations of the commute time for the bottom bridge relative to the top. Now when the algorithm is faced with a user who starts at point B, it will likely send her over the top bridge.

Which is better depends on which bridge has the longer commute time. If the top bridge is the better option, then the presence of the majority imposes a negative externality on the minority. If not, then the presence of the majority helps. These two scenarios may be difficult to distinguish.

This toy example can be formalized in the linear contextual bandits framework. There are two underlying actions (the two bridges), but these actions are not always available. To capture this, we define a parameter vector θ in $[0,1]^2$, with the two coordinates θ_1 and θ_2 representing the expected rewards for taking the top and bottom bridge respectively. (Though we motivated the example in terms of costs, it can be expressed equivalently in terms of rewards.) There are two possible context vectors: $[1\ 0]^\top$ and $[0\ 1]^\top$. A user has available an action with context vector $[1\ 0]^\top$ if and only if she has access to the top bridge. Similarly, she has available an action with context vector $[0\ 1]^\top$ if and only if she has access to the bottom bridge. The instance can then be formalized as follows.

Definition 1 (Two-Bridge Instance) *The* two-bridge instance *is an instance of linear contextual bandits. On each round t, the user who arrives is from the majority population with probability 0.95, in which case $x_{1,t} = x_{2,t} = [1\ 0]^\top$. Otherwise, the user is in the minority. In this case, with probability 0.95, $x_{1,t} = x_{2,t} = [0\ 1]^\top$ (based on Figure 1, we call these C rounds), while with probability 0.05, $x_{1,t} = [1\ 0]^\top$ and $x_{2,t} = [0\ 1]^\top$ (B rounds). We consider two values for the hidden parameter vector θ, $\theta^{(0)} = [1/2\ \ 1/2 - \varepsilon]^\top$ and $\theta^{(1)} = [1/2 - \varepsilon\ \ 1/2]^\top$ where $\varepsilon = 1/\sqrt{T}$.*

3.2. Performance of LinUCB

We start by analyzing the performance of LinUCB on the two-bridge instance. Our main result formalizes the intuition above, showing that when $\theta = \theta^{(0)}$ (that is, the top bridge is better) the majority imposes a large negative group externality on the minority, while the majority imposes a large positive externality when $\theta = \theta^{(1)}$. We assume rewards are 1-subgaussian.[3]

Theorem 2 *Consider LinUCB with any interval width function f satisfying $f(t) \geq 2\sqrt{\log(T)}$.[4] On the two-bridge instance, assuming 1-subgaussian noise on the rewards, when $\theta = \theta^{(0)}$, LinUCB achieves expected minority regret $O(1)$ when run on the minority alone, but $\Omega(\sqrt{T})$ when run on the full population. In contrast, when $\theta = \theta^{(1)}$, LinUCB achieves expected minority regret $O(1)$ when run on the full population, but $\Omega(\sqrt{T})$ when run on the minority alone.*

We omit the proofs of the $\Omega(\sqrt{T})$ lower bounds, which both follow a similar structure to the one used in the proof of the general impossibility result in Section 3.3; in fact, both of these lower bounds could be stated as an immediate corollary of Theorem 3. Essentially, an argument based on KL-divergence shows that it is difficult to distinguish between the case in which $\theta = \theta^{(0)}$ and the case in which $\theta = \theta^{(1)}$, and therefore LinUCB must choose similar actions in these two cases.

3. A random variable X is called σ-subgaussian if $E[e^{\sigma X^2}] < \infty$. A special case is Gaussians with variance σ^2.
4. For instance, the interval width function in Equation (3) satisfies this condition whenever $dc_0 \geq 4$, so one can either set $c_0 \geq 2$, or add two more dimensions to the problem instance (and set $\theta_3 = \theta_4 = 0$).

To prove the $O(1)$ upper bounds, we make heavy use of the special structure of the two-bridge instance, which significantly simplifies the analysis of LinUCB. We exploit the fact that the only context vectors available to the learner are the basis vectors $[1\ 0]^\top$ and $[0\ 1]^\top$, which essentially makes this an instance of sleeping bandits (Kleinberg et al., 2010). In this special case, the covariance matrix Z_t is always diagonal, which simplifies Equation (2) and leads to LinUCB choosing the ith basis, where i maximizes $(\hat{\theta}_t)_i + f(t)/\sqrt{(Z_t)_{ii}}$ and $(Z_t)_{ii}$ is simply the number of times that this basis vector was already chosen. Additionally, in this setting $(\hat{\theta}_t)_i$ is just the average reward observed for the ith basis vector, allowing us to bound the difference between each $(\hat{\theta}_t)_i$ and θ_i using standard concentration techniques. Using this, we show that with high probability, after a logarithmic number of rounds—during which the learner can amass at most $O(1)$ regret since the worst-case regret on any round is $\varepsilon = 1/\sqrt{T}$—the probability that LinUCB chooses the wrong action on a B round is small ($O(1/\sqrt{T})$). This leads to constant regret on expectation.

3.3. An Impossibility Result

It is natural to ask whether it is possible to design an algorithm that can distinguish between the two scenarios analyzed above, obtaining minority regret that is close to the best of LinUCB run on the minority alone and LinUCB run on the full population on any problem instance. In this section, we show that the answer is no. In particular, we prove that on the two-bridge instance, if $\Pr[\theta = \theta^{(0)}] = \Pr[\theta = \theta^{(1)}] = 1/2$, then any algorithm must suffer $\Omega(\sqrt{T})$ regret on expectation (and therefore $\Omega(\sqrt{T})$ minority regret, since all regret is incurred by minority users).

To prove this result, we begin by formalizing the idea that it is hard to distinguish between the case in which $\theta = \theta^{(0)}$ and the case in which $\theta = \theta^{(1)}$. To do so, we bound the KL-divergence between the joint distributions over the sequences of context vectors, actions taken by the given algorithm, and the given algorithm's rewards that are induced by the two choices of θ. By applying the high-probability Pinsker lemma (Tsybakov, 2009), we show that a low KL-divergence between these distributions implies that the algorithm must be likely either to choose the top bridge on B rounds more than half the time when the bottom bridge is better or to choose the bottom bridge on B rounds more than half the time when the top bridge is better, either of which would lead to high ($\Omega(\sqrt{T})$) regret as long as the number of B rounds is sufficiently large. To finish the proof, we use a simple Chernoff bound to show that the number of B rounds is large with high probability.

To derive the KL-divergence bound, we make use of the assumption that the realized rewards r_t at each round are either 0 or 1. This assumption is not strictly necessary. An analogous argument could be made, for instance, for real-valued rewards with Gaussian noise.

Theorem 3 *On the two-bridge instance with realized rewards $r_t \in \{0, 1\}$, any algorithm must incur $\Omega(\sqrt{T})$ minority regret in expectation when $\Pr[\theta = \theta^{(0)}] = \Pr[\theta = \theta^{(1)}] = \frac{1}{2}$.*

Note that "any algorithm" here includes algorithms run on the minority alone, essentially ignoring data from the majority. Theorems 2 and 3 immediately imply the following corollary.

Corollary 4 *No algorithm can simultaneously approximate the minority regret of both LinUCB run on the minority and LinUCB run on the full population up to any $o(\sqrt{T})$ multiplicative factor.*

4. Greedy Algorithms with Perturbed Contexts

We now turn our attention to externalities at an individual level. We interpret exploration as a potential externality imposed on the current user of a system by future users, since the current user would prefer the learner to take the action that appears optimal. One could avoid such externalities by running the greedy algorithm, which does just that, but it is well known that the greedy algorithm performs poorly in the worst case. In this section, we build on a recent line of work analyzing the conditions under which inherent data diversity leads the greedy algorithm to perform well.

We analyze the expected performance of the greedy algorithm under small random perturbations of the context vectors. We focus on greedy algorithms that consume new data in batches, rather than every round. We consider both Bayesian and frequentist versions, BatchBayesGreedy and BatchFreqGreedy. Our main result is that for any specific problem instance, both algorithms match the Bayesian regret of any algorithm on that particular instance up to polylogarithmic factors. We also prove that under the same perturbation assumptions, LinUCB achieves regret $\tilde{O}(T^{1/3})$ for each realization of θ, which implies a worst-case Bayesian regret of $\tilde{O}(T^{1/3})$ for the greedy algorithms. Finally, we repurpose our analysis to derive a positive result in the group setting, implying that the impossibility result of Section 3.3 breaks down when the data is sufficiently diverse.

Setting and notation. We consider a Bayesian version of linear contextual bandits, with θ drawn from a known multivariate Gaussian prior $\mathcal{P} = \mathcal{N}(\overline{\theta}, \Sigma)$, with $\overline{\theta} \in \mathbb{R}^d$ and invertible $\Sigma \in \mathbb{R}^{d \times d}$.

To capture the idea of data diversity, we assume the context vectors on each round t are generated using the following *perturbed context generation* process: First, a tuple $(\mu_{1,t}, \ \ldots \ , \mu_{K,t})$ of *mean context vectors* is drawn independently from some fixed distribution D_μ over $(\mathbb{R} \cup \{\perp\})^K$, where $\mu_{a,t} = \perp$ means action a is not available. For each available action a, the context vector is then $x_{a,t} = \mu_{a,t} + \varepsilon_{a,t}$, where $\varepsilon_{a,t}$ is a vector of random noise. Each component of $\varepsilon_{a,t}$ is drawn independently from a zero-mean Gaussian with standard deviation ρ. We refer to ρ as the *perturbation size*. In general, our regret bounds deteriorate if ρ is very small. Together we refer to a distribution D_μ, prior $\mathcal{P}$, and perturbation size ρ as a *problem instance*.

We make several technical assumptions. First, the distribution D_μ is such that each context vector has bounded 2-norm, i.e., $\|\mu_{a,t}\|_2 \leq 1$. It can be arbitrary otherwise. Second, the perturbation size needs to be sufficiently small, $\rho \leq 1/\sqrt{d}$. Third, the realized reward $r_{a,t}$ for each action a and round t is $r_{a,t} = x_{a,t}^\top \theta + \eta_{a,t}$, the mean reward $x_{a,t}^\top \theta$ plus standard Gaussian noise $\eta_{a,t} \sim \mathcal{N}(0,1)$.[5]

The history up to round t is denoted by the tuple $h_t = ((x_1, r_1), \ \ldots \ , (x_t, r_t))$.

The greedy algorithms. For the batch version of the greedy algorithm, time is divided in batches of Y consecutive rounds each. When forming its estimate of the optimal action at round t, the algorithm may only use the history up to the last round of the previous batch, denoted t_0.

BatchBayesGreedy forms a posterior over θ using prior $\mathcal{P}$ and history h_{t_0}. In round t it chooses the action that maximizes reward in expectation over this posterior. This is equivalent to choosing

$$a_t = \arg\max_a x_{a,t}^\top \theta_t^{\text{bay}}, \quad \text{where } \theta_t^{\text{bay}} := \mathbb{E}[\theta \mid h_{t_0}] \,. \tag{4}$$

BatchFreqGreedy does not rely on any knowledge of the prior. It chooses the best action according to the least squares estimate of θ, denoted θ_t^{fre}, computed with respect to history h_{t_0}:

$$a_t = \arg\max_a x_{a,t}^\top \theta_t^{\text{fre}}, \quad \text{where } \theta_t^{\text{fre}} = \arg\min_{\theta'} \textstyle\sum_{\tau=1}^{t_0} ((\theta')^\top x_\tau - r_\tau)^2. \tag{5}$$

5. Our analysis can be easily extended to handle reward noise of fixed variance, i.e., $\eta_{a,t} \sim \mathcal{N}(0, \sigma^2)$. BatchFreqGreedy would not need to know σ. BatchBayesGreedy would need to know either Σ and σ or just Σ / σ^2.

4.1. Main Results

We first state our main results before describing the intuition behind them. We state each theorem in terms of the main relevant parameters T, K, d, Y, and ρ. First, we prove that in expectation over the random perturbations, both greedy algorithms favorably compare to any other algorithm.

Theorem 5 *With perturbed context generation, there is some $Y_0 = \text{polylog}(d, T)/\rho^2$ such that with batch duration $Y \geq Y_0$, the following holds. Fix any bandit algorithm, and let $R_0(T)$ be its Bayesian regret on a particular problem instance. Then on that same instance,*

(a) BatchBayesGreedy has Bayesian regret at most $Y \cdot R_0(T/Y) + \tilde{O}(1/T)$,

(b) BatchFreqGreedy has Bayesian regret at most $Y \cdot R_0(T/Y) + \tilde{O}(\sqrt{d}/\rho^2)$.

Our next result asserts that the Bayesian regret for LinUCB and both greedy algorithms is on the order of (at most) $T^{1/3}$. This result requires additional technical assumptions.

Theorem 6 *Assume that the maximal eigenvalue of the covariance matrix Σ of the prior $\mathcal{P}$ is at most 1,[6] and the mean vector satisfies $\|\bar{\theta}\|_2 \geq 1 + \sqrt{3 \log T}$. With perturbed context generation,*

(a) With appropriate parameter settings, LinUCB has Bayesian regret $\tilde{O}(d^2\, K^{2/3}\, T^{1/3}/\rho^2)$.

(b) If $Y \geq Y_0$ as in Theorem 5, then both BatchBayesGreedy and BatchFreqGreedy have Bayesian regret at most $\tilde{O}(d^2\, K^{2/3}\, T^{1/3}/\rho^2)$.

The assumption $\|\bar{\theta}\|_2 \geq 1 + \sqrt{3 \log T}$ in Theorem 6 can be replaced with $d \geq \log T / \log \log T$. We use Theorem 6(b) to derive an "approximately prior-independent" result for BatchFreqGreedy. (For clarity, we state it for independent priors.) The bound in Theorem 6(b) deteriorates if $\mathcal{P}$ gets very sharp, but it suffices if $\mathcal{P}$ has standard deviation on the order of (at least) $T^{-2/3}$.

Corollary 7 *Assume that the prior $\mathcal{P}$ is independent over the components of θ, with variance $\kappa^2 \leq 1$ in each component. Suppose the mean vector satisfies $\|\bar{\theta}\|_2 \geq 1 + \sqrt{3 \log T}$. With perturbed context generation, if $Y \geq Y_0$ as in Theorem 5, then BatchFreqGreedy has Bayesian regret at most $\tilde{O}(d^2\, K^{2/3}\, T^{1/3}/\rho^2)$ as long as $\kappa \geq T^{-2/3}$.*

Finally, we derive a positive result on group externalities. We find that with perturbed context generation, the minority Bayesian regret of the greedy algorithms (i.e., the Bayesian regret incurred on minority rounds) is small compared to the minority Bayesian regret of any algorithm, whether run on the full population *or* on the minority alone. This sidesteps the impossibility result of Section 3.3.

Theorem 8 *Assume $Y \geq Y_0$ as in Theorem 5 and perturbed context generation. Fix any bandit algorithm and instance, and let $R_{\min}(T)$ be the minimum of its minority Bayesian regrets when it is only run over minority rounds or when it is run over the full population. Both greedy algorithms run on the full population achieve minority Bayesian regret at most $Y \cdot R_{\min}(T) + \tilde{O}(\sqrt{d}/\rho^2)$.*

4.2. Key Techniques

The key idea behind our approach is to show that, with perturbed context generation, BatchBayesGreedy collects data that is informative enough to "simulate" the history of contexts and rewards from the run of any other algorithm ALG over fewer rounds. This implies that it remains competitive with ALG since it has at least as much information and makes myopically optimal decisions.

6. In particular, if $\mathcal{P}$ is independent across the coordinates of θ, then the variance in each coordinate is at most 1.

We use the same technique to prove a similar simulation result for BatchFreqGreedy. To treat both algorithms at once, we define a template that unifies them. A bandit algorithm is called *batch-greedy-style* if it divides the timeline in batches of Y consecutive rounds each, in each round t chooses some estimate θ_t of θ, based only on the data from the previous batches, and then chooses the best action according to this estimate, so that $a_t = \arg\max_a \theta_t^\top x_{a,t}$. For a batch that starts at round t_0+1, the *batch history* is the tuple $((x_{t_0+\tau},\, r_{t_0+\tau}) :\ \tau \in [Y])$, and the *batch context matrix* is the matrix X whose rows are vectors $(x_{t_0+\tau} :\ \tau \in [Y])$; here $[Y] = \{1, \cdots, Y\}$. Similarly to the "empirical covariance matrix", we define the *batch covariance matrix* as $X^\top X$.

Let us formulate what we mean by "simulation". We want to use the data collected from a single batch in order to simulate the reward for any one context x. More formally, we are interested in the randomized function that takes a context x and outputs an independent random sample from $\mathcal{N}(\theta^\top x, 1)$. We denote it $\texttt{Rew}_\theta(\cdot)$; this is the realized reward for an action with context vector x.

Definition 9 *Consider batch B in the execution of a batch-greedy-style algorithm. Batch history h_B can simulate $\texttt{Rew}_\theta()$ up to radius $R > 0$ if there exists a function $g : \{\text{context vectors}\} \times \{\text{batch histories } h_B\} \to \mathbb{R}$ such that $g(x, h_B)$ is identically distributed to $\texttt{Rew}_\theta(x)$ conditional on the batch context matrix, for all θ and all context vectors $x \in \mathbb{R}^d$ with $\|x\|_2 \leq R$.*

Let us comment on how it may be possible to simulate $\texttt{Rew}_\theta(x)$. For intuition, suppose that $x = \frac{1}{2}\, x_1 + \frac{1}{2}\, x_2$. Then $(\frac{1}{2}\, r_1 + \frac{1}{2}\, r_2 + \xi)$ is distributed as $\mathcal{N}(\theta^\top x, 1)$ if ξ is drawn independently from $\mathcal{N}(0, \frac{1}{2})$. Thus, we can define $g(x, h) = \frac{1}{2}\, r_1 + \frac{1}{2}\, r_2 + \xi$ in Definition 9. We generalize this idea and show that a batch history can simulate $\texttt{Rew}_\theta$ as long as the batch covariance matrix has a sufficiently large minimum eigenvalue, which holds with high probability when the batch size is large.

Lemma 10 *With perturbed context generation, there is some $Y_0 = \text{polylog}(d, T)/\rho^2$ and $R = O(\rho\sqrt{d \log(TKd)})$ such that with probability at least $1 - T^{-2}$ any batch history from a batch-greedy-style algorithm can simulate $\texttt{Rew}_\theta()$ up to radius R, as long as $Y \geq Y_0$.*

If the batch history of an algorithm can simulate $\texttt{Rew}_\theta$, the algorithm has enough information to simulate the outcome of a fresh round of any other algorithm ALG. Eventually, this allows us to use a coupling argument in which we couple a run of BatchBayesGreedy with a slowed-down run of ALG, and prove that the former accumulates at least as much information as the latter, and therefore the Bayesian-greedy action choice is, in expectation, at least as good as that of ALG. This leads to Theorem 5(a). We extend this argument to a scenario in which both the greedy algorithm and ALG measure regret over a randomly chosen subset of the rounds, which leads to Theorem 8.

To extend these results to BatchFreqGreedy, we consider a hypothetical algorithm that receives the same data as BatchFreqGreedy, but chooses actions based on the (batched) Bayesian-greedy selection rule. We analyze this hypothetical algorithm using the same technique as above, and then argue that its Bayesian regret cannot be much smaller than that of BatchFreqGreedy. Intuitively, this is because the two algorithms form almost identical estimates of θ, differing only in the fact that the hypothetical algorithm uses the $\mathcal{P}$ as well as the data. We show that this difference amounts to effects on the order of $1/t$, which add up to a maximal difference of $O(\log T)$ in Bayesian regret.

Acknowledgments

We thank Solon Barocas, Dylan Foster, Jon Kleinberg, Aaron Roth, and Hanna Wallach for helpful discussions about these topics.

References

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

Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning (ICML)*, 2014.

Alekh Agarwal, Sarah Bird, Markus Cozowicz, Miro Dudik, John Langford, Lihong Li, Luong Hoang, Dan Melamed, Siddhartha Sen, Robert Schapire, and Alex Slivkins. Multiworld testing: A system for experimentation, learning, and decision-making. A white paper, available at https://github.com/Microsoft/mwt-ds/raw/master/images/MWT-WhitePaper.pdf, 2016.

Alekh Agarwal, Sarah Bird, Markus Cozowicz, Luong Hoang, John Langford, Stephen Lee, Jiaji Li, Dan Melamed, Gal Oshri, Oswaldo Ribas, Siddhartha Sen, and Alex Slivkins. Making contextual decisions with low technical debt. *CoRR arXiv:1606.03966*, 2017.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

Hamsa Bastani, Mohsen Bayati, and Khashayar Khosravi. Exploiting the natural exploration in contextual bandits. *CoRR arXiv:1704.09011*, 2017.

Alberto Bietti, Alekh Agarwal, and John Langford. Practical evaluation and optimization of contextual bandit algorithms. *CoRR arXiv:1802.04064*, 2018.

Kostas Bimpikis, Yiangos Papanastasiou, and Nicos Savva. Crowdsourcing exploration. *Management Science*, 2017. Forthcoming.

Sarah Bird, Solon Barocas, Kate Crawford, Fernando Diaz, and Hanna Wallach. Exploring or exploiting? Social and ethical implications of autonomous experimentation in AI. Available at SSRN: https://ssrn.com/abstract=2846909, also appeared at the Workshop on Fairness, Accountability, and Transparency in Machine Learning, 2016.

Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. *Foundations and Trends in Machine Learning*, 5(1), 2012.

L. Elisa Celis and Nisheeth K Vishnoi. Fair personalization. *CoRR arXiv:1707.02260, also appeared at the Workshop on Fairness, Accountability, and Transparency in Machine Learning*, 2017.

Venkat Chandrasekaran, Benjamin Recht, Pablo A Parrilo, and Alan S Willsky. The convex geometry of linear inverse problems. *Foundations of Computational Mathematics*, 12(6):805–849, 2012.

Yeon-Koo Che and Johannes Hörner. Optimal design for social learning. Preprint, 2015.

Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *CoRR arXiv:1703.00056*, 2017.

Wei Chu, Lihong Li, Lev Reyzin, and Robert E Schapire. Contextual bandits with linear payoff functions. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

John D Cook. Upper and lower bounds for the normal distribution function, 2009.

Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Innovations in Theoretical Computer Science (ITCS)*, 2012.

Peter Frazier, David Kempe, Jon M. Kleinberg, and Robert Kleinberg. Incentivizing exploration. In *ACM Conference on Economics and Computation (ACM EC)*, 2014.

Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Svante Janson. Tail bounds for sums of geometric and exponential variables. *CoRR arXiv:1709.08157*, 2017.

Matthew Joseph, Michael Kearns, Jamie H Morgenstern, and Aaron Roth. Fairness in learning: Classic and contextual bandits. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Sampath Kannan, Jamie Morgenstern, Aaron Roth, Bo Waggoner, and Zhiwei Steven Wu. A smoothed analysis of the greedy algorithm for the linear contextual bandit problem. *CoRR arXiv:1801.03423*, 2018.

Michael Kearns, Aaron Roth, and Zhiwei Steven Wu. Meritocratic fairness for cross-population selection. In *International Conference on Machine Learning (ICML)*, 2017.

Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. In *Innovations in Theoretical Computer Science (ITCS)*, 2017.

Robert Kleinberg, Alexandru Niculescu-Mizil, and Yogeshwer Sharma. Regret bounds for sleeping experts and bandits. *Machine Learning*, 80(2-3):245–272, 2010.

Ilan Kremer, Yishay Mansour, and Motty Perry. Implementing the wisdom of the crowd. *Journal of Political Economy*, 122:988–1012, 2014.

John Langford and Tong Zhang. The Epoch-Greedy Algorithm for Contextual Multi-armed Bandits. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *International World Wide Web Conference (WWW)*, 2010.

Yang Liu, Goran Radanovic, Christos Dimitrakakis, Debmalya Mandal, and David C Parkes. Calibrated fairness in bandits. *CoRR arXiv:1707.01875*, 2017.

Yishay Mansour, Aleksandrs Slivkins, and Vasilis Syrgkanis. Bayesian incentive-compatible bandit exploration. In *ACM Conference on Economics and Computation (ACM EC)*, 2015.

Yishay Mansour, Aleksandrs Slivkins, and Zhiwei Steven Wu. Competing bandits: Learning under competition. In *Innovations in Theoretical Computer Science (ITCS)*, 2018.

Philippe Rigollet and Assaf Zeevi. Nonparametric Bandits with Covariates. In *Conference on Learning Theory (COLT)*, 2010.

Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.

Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, 2012.

Alexandre B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer, 2009.