Linear Bandits with Feature Feedback

Urvashi Oswal,[1] **Aniruddha Bhargava,**[*] **Robert Nowak**[2]
[1]Oracle Labs, Redwood Shores, CA
[2]University of Wisconsin-Madison, Madison, WI
urvashi.oswal@oracle.com, aniruddha.j@gmail.com, rdnowak@wisc.edu

Abstract

This paper explores a new form of the linear bandit problem in which the algorithm receives the usual stochastic rewards as well as stochastic feedback about which features are relevant to the rewards, the latter feedback being the novel aspect. The focus of this paper is the development of new theory and algorithms for linear bandits with feature feedback which can achieve regret over time horizon T that scales like $k\sqrt{T}$, without prior knowledge of which features are relevant nor the number k of relevant features. In comparison, the regret of traditional linear bandits is $d\sqrt{T}$, where d is the total number of (relevant and irrelevant) features, so the improvement can be dramatic if $k \ll d$. The computational complexity of the algorithm is proportional to k rather than d, making it much more suitable for real-world applications compared to traditional linear bandits. We demonstrate the performance of the algorithm with synthetic and real human-labeled data.

1 Introduction

Linear stochastic bandit algorithms are used to sequentially select actions to maximize rewards. For instance, (Deshpande and Montanari 2012) propose to model recommendation systems that help users navigate through a large collection of items (products, videos, documents) using linearly parameterized multi-armed bandits. This model strikes a balance by allowing the user to explore the space of available items and probing the user's preferences. The linear bandit model assumes that the expected reward of each action is an (unknown) linear function of a (known) finite-dimensional feature associated with the action. Mathematically, if $\mathbf{x}_t \in \mathbf{R}^d$ is the feature associated with the action chosen at time t, then the stochastic reward is

$$y_t = \mathbf{x}_t^\top \boldsymbol{\theta}_* + \eta_t, \tag{1}$$

where $\boldsymbol{\theta}_*$ is the unknown linear functional (representing the user's preferences) and η_t is a zero mean random variable. The goal is to adaptively select actions to maximize the rewards (corresponding to user's assessment of the chosen item's value). This involves (approximately) learning $\boldsymbol{\theta}_*$ and exploiting this knowledge. Linear bandit algorithms that exploit this special structure have been extensively studied and applied (Rusmevichientong and Tsitsiklis 2010; Abbasi-Yadkori, Pal, and Szepesvari 2011).

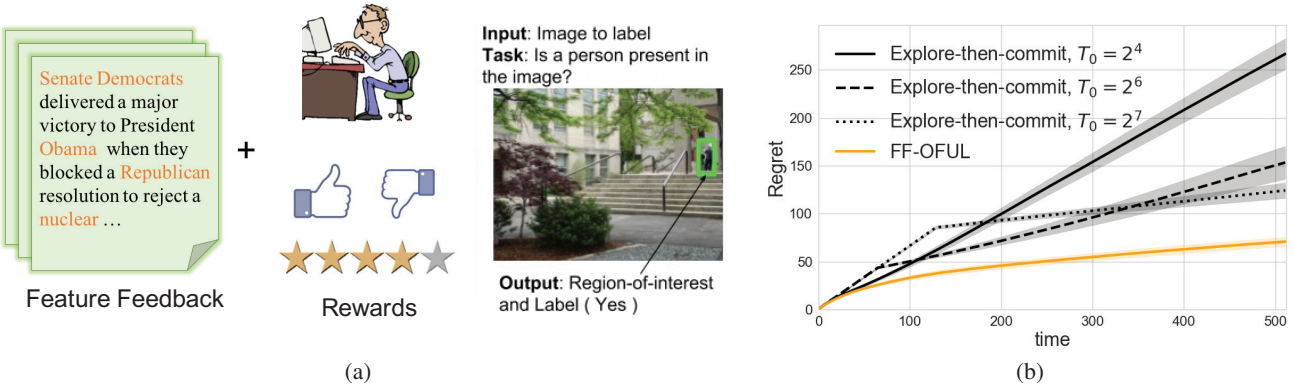
Unfortunately, standard linear bandit algorithms suffer from the curse of dimensionality. The regret grows linearly with the feature dimension d. The dimension d may be quite large in modern applications (*e.g.*, 1000s of features in NLP or image/vision applications). In the recommendations setting, the existing bounds easily require $T > 100$s of ratings to be meaningful, which is not realistic. The high-dimensionality also makes it challenging to employ state-of-the-art algorithms since it involves maintaining and updating a $d \times d$ matrix at every stage. However, in many cases the linear function may only involve a sparse subset of the $k < d$ features, and this can be exploited to partially reduce dependence on d. In such cases, the regret of sparse linear bandit algorithms scales like $\sqrt{dk}$ (Abbasi-Yadkori, Pal, and Szepesvari 2012; Lattimore and Szepesvári 2018).

We tackle the problem of linear bandits from a new perspective that incorporates feature feedback in addition to reward feedback, mitigating the curse of dimensionality. Specifically, we consider situations in which the algorithm receives a stochastic reward *and* stochastic feedback indicating which, if any, feature-dimensions were relevant to the reward value. For example, consider a situation in which users rate recommended text documents and additionally highlight keywords or phrases that influenced their ratings. Figure 1(a) illustrates the idea. Obviously, the additional "feature feedback" may significantly improve an algorithm's ability to home-in on the relevant features. The focus of this paper is the development of new theory and algorithms for linear bandits with feature feedback. We show that the regret of linear bandits with feature feedback scales linearly in k, the number of relevant features, without prior knowledge of which features are relevant nor the value of k. This leads to large improvements in theory and practice.

The simple feedback model, where the user directly selects a subset of the relevant features, can be generalized by allowing for an indirect form of feedback. For example, the user can select a region of an image instead of a subset of the standard deep neural network features. This form of

[*]This research was performed when U.O. and A.B. were at University of Wisonsin-Madison.

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.



(a) (b)

Figure 1: (a) (Left) Highlighted words for text-based applications and (Right) Region-of-interest feature feedback for image-based applications. (b) Comparison of the explore-then-commit strategy for different values of T_0 and our new FF-OFUL algorithm which combines exploration and exploitation steps (details of data generation in Section 5.1).

feedback could be used in different ways. For instance, we could use methods to map deep image features to image regions. However, in this paper we focus on the processes and benefits of incorporating direct feature feedback, and defer the development of indirect feedback models to future work.

Perhaps the most natural and simple way to leverage the feature feedback is an explore-then-commit strategy. In the first T_0 steps the algorithm selects actions at random and receives rewards and feature feedback. If T_0 is sufficiently large, then the algorithm will have learned all or most of the relevant features and it can then switch to a standard linear bandit algorithm operating in the lower-dimensional subspace defined by those features. There are two major problems with such an approach:

1. The correct choice of T_0 depends on the prevalence of relevant features in randomly selected actions, which generally is unknown. If T_0 is too small, then many relevant features will be missed and the long-run regret will scale linearly with the time horizon. If T_0 is too large, then the initial exploration period will suffer excess regret. This is depicted in Figure 1(b).
2. Regardless of the choice of T_0, the regret will grow linearly for $t < T_0$. The new FF-OFUL algorithm that we propose combines exploration and exploitation from the start and can lead to smaller regret initially and asymptotically as shown in Figure 1(b).

These observations motivate our proposed approach that dynamically adjusts the trade-off between exploration and exploitation. A key aspect of the approach is that it is automatically adaptive to the unknown number of relevant features k. Our theoretical analysis shows that its regret scales like $k\sqrt{T}$. Experimentally, we show the algorithm generally outperforms traditional linear bandits and the explore-then-commit strategy. This is due to the fact that the dynamic algorithm exploits knowledge of relevant features as soon as they are identified, rather than waiting until all or most are found. A key consequence is that our proposed algorithm yields significantly better rewards at early stages of the process, as shown in Figure 1(b) and in more comprehensive experiments later in the paper. The intuition for this is that estimating $\boldsymbol{\theta}_*$ on a fraction of the relevant coordinates can be exploited to recover a fraction of the optimal reward. Similar ideas are explored in linear bandits (without feature feedback) in (Deshpande and Montanari 2012).

1.1 Definitions

For round, t, let $\mathcal{X}_t \subseteq \mathbb{R}^d$ be the set of actions/items provided to the learner. We assume the standard linear model for rewards with a hidden weight vector $\boldsymbol{\theta}_* \in \mathbb{R}^d$. If the learner selects an action, $\mathbf{x}_t \in \mathcal{X}_t$, it receives reward, y_t, defined in (1) where η_t is noise with a sub-Gaussian random distribution with parameter R. For the set of actions $\mathcal{X}_t$, the optimal action is given by, $\mathbf{x}_t^* := \text{argmax}_{\mathbf{x} \in \mathcal{X}_t} \mathbf{x}^\top \boldsymbol{\theta}_*$, which is unknown. We define regret as,

$$R_T = \sum_{t=1}^{T} \left(\mathbf{x}_t^{*\top} \boldsymbol{\theta}_* - \mathbf{x}_t^\top \boldsymbol{\theta}_* \right). \tag{2}$$

This is also called cumulative regret but, unless stated otherwise, we will refer to it as regret. We refer to the quantity $\mathbf{x}_t^{*\top} \boldsymbol{\theta}_* - \mathbf{x}_t^\top \boldsymbol{\theta}_*$ as the instantaneous regret which is the difference between the optimal reward and the reward received at that instant. We make the standard assumption that the algorithm is provided with an enormous action set which is only changing slowly over time, for instance, from sampling the actions without replacement ($\mathcal{X}_{t+1} = \mathcal{X}_t \backslash \mathbf{x}_t$).

1.2 Related Work

The area of contextual bandits was introduced by (Ginebra and Clayton 1995). The first algorithms for linear bandits appeared in (Abe and Long 1999) followed by those using the optimism in the face of uncertainty principle, (Auer and Long 2002), (Dani, Hayes, and Kakade 2008). (Rusmevichientong and Tsitsiklis 2010) showed matching upper and lower bounds when the action (feature) set is a unit hypersphere. Finally, (Abbasi-Yadkori, Pal, and Szepesvari

Algorithm 1 OFUL

1: **for** $t = 1, 2, \ldots, T-1$ **do**
2: $(\mathbf{x}_t, \tilde{\boldsymbol{\theta}}_t) = \text{argmax}_{(\mathbf{x},\boldsymbol{\theta}) \in \mathcal{X}_t \times \mathcal{C}_{t-1}} \langle \mathbf{x}, \boldsymbol{\theta} \rangle$
3: Select action $\mathbf{x}_t$ and receive reward y_t.
4: Update $\overline{\mathbf{V}}_t = (\mathbf{X}_t^T \mathbf{X}_t + \lambda \mathbf{I})$ and $\widehat{\theta}_t = \overline{\mathbf{V}}_t^{-1} \mathbf{X}_t^T \mathbf{y}_t$
5: Update ellipsoidal confidence set $\mathcal{C}_t$ as
$\mathcal{C}_t = \Big\{ \boldsymbol{\theta} : \|\widehat{\theta}_t - \boldsymbol{\theta}\|_{\overline{\mathbf{V}}_t}$
$\leq R\sqrt{2\log\Big(\frac{\det(\overline{\mathbf{V}}_t)^{1/2}\det(\lambda I)^{-1/2}}{\delta}\Big)} + \lambda^{1/2} S \Big\}$
6: **end for**

2011) gave a tight regret bound using new martingale techniques. We use their algorithm, OFUL, as a subroutine in our work. In the area of sparse linear bandits, regret bounds are known to scale like $\sqrt{kdT}$, (Abbasi-Yadkori, Pal, and Szepesvari 2012),(Lattimore and Szepesvári 2018), when operating in a d dimensional feature space with k relevant features. The strong dependence on the ambient dimension d is unavoidable without further (often strong and unrealistic) assumptions. For instance, if the distribution of feature vectors is isotropic or otherwise favorably distributed, then the regret may scale like $k\log(d)\sqrt{T}$, e.g., by using incoherence based techniques from compressed sensing (Carpentier and Munos 2012). These results also assume knowledge of sparsity parameter k and without it no algorithm can satisfy these regret bounds for all k simultaneously. In contrast, we propose a new algorithm that automatically adapts to the unknown sparsity level k and removes the dependence of regret on d by exploiting additional feature feedback. In terms of feature feedback in text-based applications, (Croft and Das 1989) have proposed a method to reorder documents based on the relative importance of words using feedback from users. (Poulis and Dasgupta 2017) consider a similar problem but for learning a linear classifier. We use a similar feedback model but focus on the bandit setting where such feedback can be naturally collected along with rewards. The idea of allowing user's to provide richer forms of feedback has been studied in the active learning literature (Raghavan, Madani, and Jones 2006), (Druck, Settles, and McCallum 2009) and also been considered in other (interactive) learning tasks, such as cognitive science (Roads, Mozer, and Busey 2016), machine teaching (Chen et al. 2018), and NLP tasks (Yessenalina, Yue, and Cardie 2010).

2 Model for Feature Feedback

The algorithm presents the user with an item (*e.g.*, document) and the user provides feedback in terms of whether they like the item or not (logistic model) or how much they like it (inner product model). The user also selects a few features (*e.g.*, words), if they can find them, to help orient the search. We make the following assumptions.

Assumption 1 (Sparsity)**.** *The hidden weight vector $\boldsymbol{\theta}_* \in \mathbb{R}^d$ is k-sparse and k is unknown. In other words, $\boldsymbol{\theta}_*$ has at most k non-zero entries or if supp$(\boldsymbol{\theta}_*) = \{i | \boldsymbol{\theta}_{*i} \neq 0\}$ then $|supp(\boldsymbol{\theta}_*)| = k \leq d$.*

Assumption 2 (Discoverability)**.** *For an action $\mathbf{x} \in \mathcal{X}$ selected uniformly at random, the probability that a relevant feature is present and is selected is at least $p > 0$ (unknown).*

Assumption 3 (Noise)**.** *Users may report irrelevant features. The number of reported irrelevant features (denoted by $0 \leq k' \leq d - k$) is unknown in advance.*

Assumption 1 ensures that there are at most k relevant features, however we stress that the value of k is unknown (it is possible that all d features are relevant). Assumption 2 ensures that while every item may not have relevant features, we are able to find them with a non-zero probability when searching through items at random. This assumption can be viewed as a (possibly pessimistic) lower bound on the rate at which relevant features are discovered. For example, it is possible that exploitative actions may yield relevant features at a higher rate (e.g., relevant features may be correlated with higher rewards). We do not attempt to model such possibilities since this would involve making additional assumptions that may not hold in practice. Assumption 3 accounts for ambiguous features that are irrelevant but users erring on the side of marking as relevant.

The set up is as follows: we have a set of items, $\mathcal{X} \subseteq \mathbb{R}^d$ that we can propose to the users. There is a hidden weight vector $\boldsymbol{\theta}_* \in \mathbb{R}^d$ that is k-sparse. We will further assume that $\|\boldsymbol{\theta}_*\| \leq S$ and the action vectors are bounded in norm: $\forall \mathbf{x} \in \mathcal{X}, \|\mathbf{x}\| \leq L$. Besides the reward y_t, (1), at each time-step the learner gets $\mathcal{I}_t \subseteq [d]$ which is the relevance feedback information. The model further specifies that $\forall j \in \text{supp}(\boldsymbol{\theta}_*), \Pr(j \in \mathcal{I}_t) \geq p$. That is, the probability a relevant feature is selected at random is at least p. We need this assumption to make sure that we can find all the relevant features.

3 Algorithm

In this section, we introduce an algorithm that uses feature relevance feedback by starting with a small feature space and gradually increasing the space over time without knowledge of k. We use the OFUL algorithm (stated as Algo. 1) based on the principle of optimism in face of uncertainty as a subroutine. The algorithm constructs ellipsoidal confidence sets centered around the ridge regression estimate, using observed data such that the sets contain the unknown θ_* with high probability, and selects the action/item that maximizes the inner product with any θ from the confidence set.

All updates are made only in the dimensions that have been marked as relevant and the space is dynamically increased as new relevant features are revealed. If nothing is marked as relevant, then by default the actions are selected at random, potentially suffering the worst possible reward but, at the same time, increasing our chances of getting relevance feedback leading to a trade-off. Note that the algorithm is adaptive to the unknown number of relevant features k. If k were known, we could stop looking for features when all relevant ones have been selected. We find that in practice, this algorithm has an additional benefit of being more robust to changes in the ridge parameter (λ) due to its intrinsic regularization of restricting the parameter space.

Algorithm 2 Feature Feedback OFUL (FF-OFUL)

Let the set of relevant indices, $\mathcal{R}_0, \mathcal{I}_0 = \{\}$.
while $\mathcal{I}_0$ is empty **do**
  Select action at random, $\mathcal{I}_0 = \{$ indices revealed $\}$
end while
$\mathcal{R}_1 = \mathcal{R}_0 \bigcup \mathcal{I}_0$
Initialize $\mathcal{C}_0$ using actions sampled.
for $t = 1, 2, \ldots, T$ **do**
  Let $\boldsymbol{X}_t$ be the feature matrix restricted to $\mathcal{R}_t$.
  Set $\epsilon_t = 1/\sqrt{t}$. Draw b_t from bernoulli(ϵ_t)
  if $b_t = 1$ **then**
    Pick an action $\mathbf{x}_t$ uniformly at random from $\mathcal{X}_t$,
  else
    Pick $(\mathbf{x}_t, \tilde{\boldsymbol{\theta}}_t) = \text{argmax}_{(\mathbf{x},\boldsymbol{\theta}) \in \mathcal{X}_t \times \mathcal{C}_{t-1}} \langle \mathbf{x}, \boldsymbol{\theta} \rangle$
  end if
  With action $\mathbf{x}_t$ observe reward y_t and indices, $\mathcal{I}_t$.
  Update $\mathcal{R}_t = \mathcal{R}_t \bigcup \mathcal{I}_t$
  if $\mathcal{I}_t$ is empty **then**
    Rank one update to $\overline{\mathbf{V}}_t, \widehat{\theta}_t, \mathcal{C}_t$ (see Algo. 1) using $(y_t, \mathbf{x}_t)$
  else
    Update $\boldsymbol{X}_t$ with features in $\mathcal{R}_t$.
    Recompute $\overline{\mathbf{V}}_t, \widehat{\theta}_t, \mathcal{C}_t$ with new feature set $\boldsymbol{X}_t$.
  end if
end for

(Abbasi-Yadkori, Pal, and Szepesvari 2011) provide a $\tilde{O}(d\sqrt{t})$ bound on the regret of OFUL stated as Algorithm 1 by ignoring constants and logarithmic terms. We prove a similar result but reduce the dependence on the dimension from d to k. In order to do so, we must discover the support of $\boldsymbol{\theta}_*$. The idea being that we apportion a set of actions, with a form of ϵ-greedy algorithm due to (Sutton and Barto 1998), to random plays in order to guarantee that we find all the relevant features, otherwise we run OFUL on the identified relevant dimensions. Reducing the proportion of random actions over time guarantees that the regret remains sub-linear in time. We propose Algorithm 2 to exploit feature feedback. Here, at each time t, with probability proportional to $1/\sqrt{t}$, the algorithm selects an action/item to present at random, otherwise it selects the item recommended by feature-restricted-OFUL.

4 Regret Analysis

In this section, we state regret bounds for the FF-OFUL algorithm along with a sketch of the proof and discuss approaches to improve the bounds while deferring proof details to supplementary material.

4.1 Regret Bound for Algorithm 2 (FF-OFUL)

Recall that $\forall \mathbf{x} \in \mathcal{X}, \|\mathbf{x}\| \leq L$ and $\|\boldsymbol{\theta}_*\| \leq S$. Therefore, for any action, the worst-case instantaneous regret can be derived using Cauchy-Schwarz as follows:

$$\begin{aligned} |\langle \mathbf{x}^*, \boldsymbol{\theta}_* \rangle - \langle \mathbf{x}, \boldsymbol{\theta}_* \rangle| &\leq |\langle \mathbf{x}^*, \boldsymbol{\theta}_* \rangle| + |\langle \mathbf{x}, \boldsymbol{\theta}_* \rangle| \\ &\leq \|\mathbf{x}^*\| \, \|\boldsymbol{\theta}_*\| + \|\mathbf{x}\| \, \|\boldsymbol{\theta}_*\| \\ &\leq 2SL \end{aligned}$$

We provide the main result that bounds the regret (2) of Algorithm 2 in the following theorem.

Theorem 1. *Assume that $\forall t > 0$ and $\mathbf{x} \in \mathcal{X}_t$, $\langle \mathbf{x}, \boldsymbol{\theta}_* \rangle \in [-1, 1]$, with the additional assumptions 1, 2 and 3 (k' = 0). Then with probability $\geq 1 - \delta$, the cumulative regret after T steps for Algorithm 2,*

$$\begin{aligned} R_T \leq \; & \frac{8SL}{\log 6M/\delta} \left(\frac{\log 3k/\delta}{\log 1/(1-p)} \right)^2 \\ & + \log_2 \frac{T}{2} \left(3SL \sqrt{T \log \frac{6M}{\delta}} \right) \\ & + 4 \log_2 \frac{T}{2} \sqrt{\frac{T}{2} k \log(\lambda + nL/k)} \left(\lambda^{1/2} S \right. \\ & \left. + \; R \sqrt{2 \log(3M/\delta) + k \log(1 + TL/(2\lambda k))} \right). \end{aligned}$$

where $M = \log_2 \frac{T}{2}$, $\lambda > 0$ is the ridge regression parameter and k is the (unknown) number of relevant features.

In other words, with high probability, the regret of Algorithm 2 (FF-OFUL) scales like $\tilde{O}(k\sqrt{T} + \frac{1}{p^2})$, by ignoring constants and logarithmic terms and using the taylor series expansion of $-\log(1-p)$, over time horizon T where k is the number of relevant features and p is the probability with which a relevant feature is marked in an action selected uniformly at random.

Remark. The values of k and p are unknown and the algorithm implicitly adapts to these problem-dependent parameters. Since the regret of any algorithm is trivially bounded by O(T) (for bounded rewards), our new regret bound is non-trivial for $T > \max(k^2, p^{-2})$. In comparison, linear bandits without feature feedback have an O($d\sqrt{T}$) regret, which is non-trivial only when $T > d^2$. So, our new algorithm enjoys a better regret bound if $p > d^{-1}$, which is a reasonable condition in high-dimensional settings (*e.g.*, $d = 10^4$).

The three terms in the total regret come from the following events. Regret due to: (1) exploration to guarantee observing all the relevant features (with high probability), (2) exploration after observing all relevant features (due to lack of knowledge of p or k), and (3) exploitation and exploration running OFUL (after having observed all relevant features).

In practice, feature feedback may be noisy. Sometimes, features that are irrelevant may be marked as relevant. To account for this, we can relax our assumption to allow for subset of k' irrelevant features that are mistakenly marked as relevant. Including these features will increase the regret but the theory goes through without much difficulty as stated in the following corollary.

Corollary 1. *With the same assumptions as Theorem 1, if a fixed set of k' irrelevant features were indicated by the user (Assumption 3), then the regret of Algorithm 2 (FF-OFUL) scales like $\tilde{O}((k + k')\sqrt{T} + \frac{1}{p^2})$.*

The corollary follows since exploration is not affected by this noise and the regret of exploitation on the vector restricted to $k+k'$ dimensions scales like $(k+k')\sqrt{T}$. This accounts for having some features being ambiguous and users erring on the side of marking them as relevant. This only results in slightly higher regret so long as $k+k'$ is still smaller than d. One could improve this regret by making assumptions on the probabilities of feature selection to weed out the irrelevant features.

Proof Sketch of Main Result We provide a sketch of the proof here and defer details to supplementary material. Recall, the cumulative regret is summed over the instantaneous regrets for $t = 1, \ldots, T$. We divide the cumulative regret across epochs $s = 0, \ldots, M$ of doubling size $T_s = 2^s$ for $M = \log_2 \frac{T}{2}$.

This ensures that the last epoch dominates the regret and it allows for the evolving feature space. For each epoch, we bound the regret under two events, all relevant features have been identified (via user feedback) up to that epoch or not. First, we bound the regret conditioned on the event that all the relevant features have been identified in Lemma 3. This is further, in expectation, broken down into the ϵ_s portion of random actions for pure exploration (Lemma 1) and $1-\epsilon_s$ modified OFUL actions on the k-dimensional feature space for exploitation-exploration (Lemma 2). For pure exploration, we use the worst case regret bound but since ϵ_s is decreasing this does not dominate the OFUL term. Second, we bound the probability that some of the relevant features are not identified so far (Proposition 3), which is a constant depending on k and p since it becomes zero after enough epochs have passed. Pure exploration ensures the probability that some features are not identified decreases with each passing epoch. An issue of bounding regret of the actions selected by OFUL subroutine in each epoch is that, unlike OFUL, the confidence sets in our algorithm are constructed using additional actions from exploration rounds and past epochs. To accommodate this we prove a regret bound for this variation in Lemma 2. Putting all this together gives us the final result.

Lower bound. The arguments from (Dani, Hayes, and Kakade 2008),(Rusmevichientong and Tsitsiklis 2010) can be used get a lower bound of $O(k\sqrt{T})$. Suppose we knew the support, then any linear bandit algorithm that is run on that support must incur an order $k\sqrt{T}$ regret. We don't know the support but we estimate it with high probability and therefore the lower bound also applies here. Our algorithm is optimal up to log factors in terms of the dimension.

4.2 Better Early-Regret Bounds

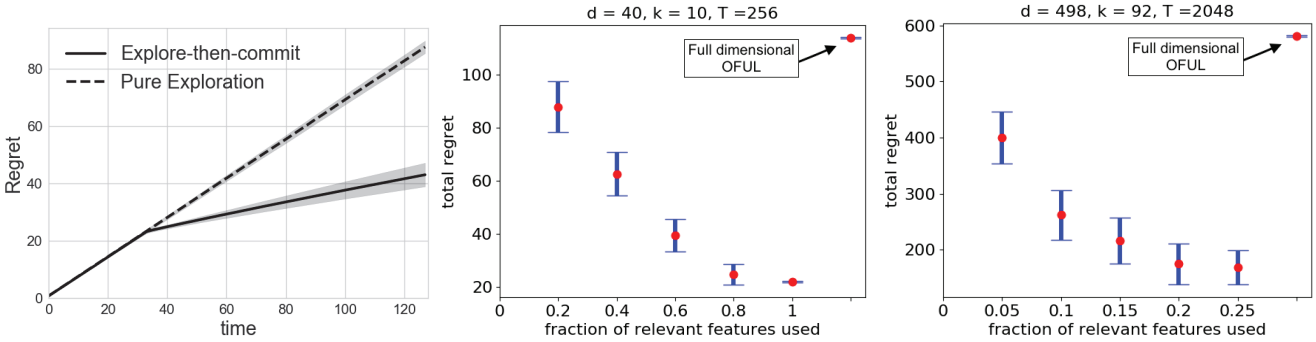
Our analysis bounds the regret of early rounds, before observing all relevant features, with the worst case regret which may be too pessimistic in practice. We present **results to support the idea of restricting the feature space in the short-term horizon and growing the feature space over time**. The results also suggest that an additional assumption on the behavior of early-regret could lead to better constants in our bounds. Any linear bandit algorithm restricted to the support of $\boldsymbol{\theta}_*$ must incur an order $k\sqrt{T}$ regret so one can only hope to improve the constants of the bound.

In Figure 2(a) it can be seen that the average linear regret of pure exploration has a slope that is worse than OFUL restricted to a subset of the relevant features. The $N = 1000$ actions were randomly sampled from the unit sphere in $d = 40$ dimensions and θ_* was generated with $k = 5$ sparsity. For a pure exploration algorithm that picks actions uniformly at random, independent of the problem instance, the regret can only be bound by $2SLT$ or O(T). Let $R^{\mathcal{K}}_{alg}$ be the expected regret of algorithm alg run on a subset of relevant features $\mathcal{K} \subseteq \{1, \ldots, k\}, |\mathcal{K}| = j \leq k$. For example, alg could be the OFUL algorithm. Then $R^{\mathcal{K}}_{alg}$ represents the expected regret of OFUL restricted to features in $\mathcal{K}$. To discover relevant features ($\mathcal{K}$) we can employ an explore-then-commit strategy which first explores for $\sim \sqrt{T}$ time followed by an exploitation stage such as OFUL restricted to features in $\mathcal{K}$. The rewards in the latter exploitation stage can be divided in two parts,

$$\langle \mathbf{x}, \boldsymbol{\theta}_* \rangle = \left\langle \mathbf{x}^{\mathcal{K}}, \boldsymbol{\theta}^{\mathcal{K}}_* \right\rangle + \left\langle \mathbf{x}^{\mathcal{K}^c}, \boldsymbol{\theta}^{\mathcal{K}^c}_* \right\rangle,$$

where $\mathbf{x}^{\mathcal{K}}$ is the portion of $\mathbf{x}$ restricted to $\mathcal{K}$ and $\mathcal{K} \cup \mathcal{K}^c = [d]$. Similarly, the regret $R^{\mathcal{K}}_{alg}$ can be divided in two parts. Roughly the regret on $\mathcal{K}$ can be bounded by $j\sqrt{T}$ under certain conditions using the OFUL regret bound. For the regret on $\mathcal{K}^c$, suppose each relevant component of $\boldsymbol{\theta}_*$ has a mean square value of S^2/k (this can be achieved with a sparse gaussian model such as those described in (Deshpande and Montanari 2012)). This yields $\mathbf{E}\|\boldsymbol{\theta}^{\mathcal{K}^c}_*\|^2 \approx \frac{k-j}{k} S^2$ where $j = |\mathcal{K}|$. The worst-case instantaneous regret bound on $\mathcal{K}^c$ becomes $2\sqrt{(k-j)/k}SL$ leading to an improvement in the linear regret slope by a factor of $\sqrt{(k-j)/k}$ over pure exploration (see Figure 2).

Figure 2(b) shows average regret of OFUL restricted to feature subsets of different sizes with synthetic data ($N = 1000$ actions, $d = 40$ and $k = 10$). For $j \in \{2, 4, \ldots, 10\}$, we randomly picked 100 subsets of size j from the support of $\boldsymbol{\theta}_*$. We report the average regret of OFUL for a short horizon, $T = 2^8$, restricted to the 100 random subsets. We also plot average regret of OFUL on the full $d = 40$ dimensional data. Figure 2(c) depicts the same with real data from (Poulis and Dasgupta 2017) with $d = 498$ and sparsity, $k = 92$, we choose 100 random subsets of size $j \in \{5, 10, \ldots, 25\}$ from the set of relevant features marked by users (see Section 5 for details) and report the average regret of OFUL restricted to the feature subsets for a relatively short time horizon, $T = 2^{11}$. The plots show that, in the short horizon, it may be more beneficial to use a subset of the relevant features than using the total feature set which may include many irrelevant features. The intuition is that when OFUL has not seen many samples, it does not have enough information to separate the irrelevant dimensions from relevant ones. As time goes on (i.e., for longer horizons) OFUL's relative performance improves since it enjoys sublinear regret but would ultimately be a factor of d/k worse than that of the low-dimensional model that includes all k relevant features.



(a) Synthetic data (b) Synthetic data (c) Data from (Poulis and Dasgupta 2017)

Figure 2: (a) Regret of pure exploration versus explore-then-commit strategy (b,c) Average regret of OFUL restricted to feature subsets (red dots) with 95% confidence regions (blue). The short time horizon was chosen to make the case for restricting the feature space in early rounds. In the long horizon, with more information, the relative performance of OFUL improves, but would ultimately be a factor of d/k worse than that of the low-dimensional model that includes all k relevant features.

5 Experiments

5.1 Results with Synthetic Data

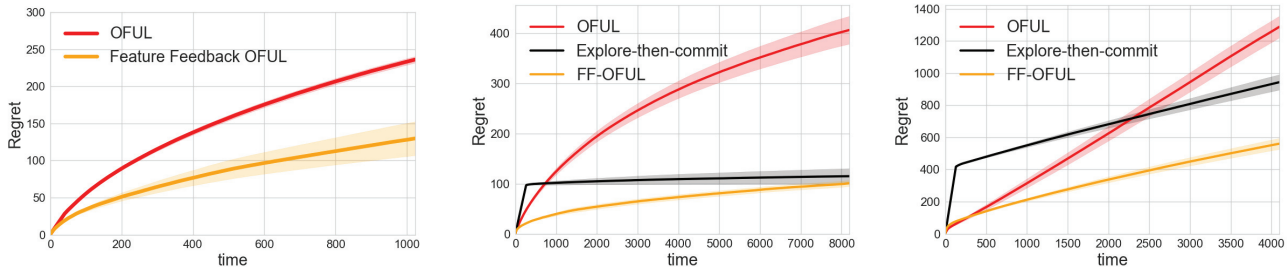
For synthetic data, we simulate a text categorization dataset as follows. Each action corresponds to an article. Generally an article contains only a small subset of the words from the dictionary. Therefore, to simulate documents we generate 1000 sparse actions in 40 dimensions. A 5-sparse reward generating vector, $\boldsymbol{\theta}_*$, is chosen at random. This represents the fact that in reality a document category probably contains only a few relevant words. The features represent word counts and hence are always positive. Here we have access to $\boldsymbol{\theta}_*$ therefore for any action $\mathbf{x}$, we use the standard linear model (1) for the reward y_t with $\eta_t \sim \mathcal{N}(0, R^2)$. The support of $\boldsymbol{\theta}_*$ is taken as the set of oracle relevant words. For every round, each word from the intersection of the support of the action and oracle relevant words is marked as relevant with probability $(p' = 0.1)$. Figure 3(a) shows the results averaged over 100 random trials for sparse $\boldsymbol{\theta}_*$ with $k = 5, d = 40$, and 1000 actions. As expected, the FF-OFUL algorithm outperforms standard OFUL significantly. Figure 6(b) in supplement also shows that the feedback does not hurt the performance much for non-sparse $\boldsymbol{\theta}_*$ with $k = d = 40$. Figure 1(b) compares the performance of FF-OFUL with an explore-then-commit strategy.

5.2 Results with 20Newsgroup Dataset

We use the 20Newsgroup (20NG) dataset from (Lang 1995). It has 2×10^5 documents covering 20 topics such as politics, sports. We choose a subset of 5 topics (misc.forsale, rec.autos, sci.med, comp.graphics, talk.politics.mideast) with approximately 4800 documents posted under these topics. For the word counts, we use the TF-IDF features for the documents which give us approximately $d = 47781$ features. For the sake of comparing our method with OFUL, we first report 500 and 1000 dimensional experiments and then on the full $47,781$ dimensional data. To do this, we use logistic regression to train a high accuracy sparse classifier to select 153 features. Then select an additional 847 features at random in order to simulate high dimensional features. We compared OFUL and FF-OFUL algorithms on this data. This is similar to the way (Poulis and Dasgupta 2017) ran experiments in the classification setting. We ran only our algorithm on the full 47781 dimension data since it was infeasible to run OFUL. For the reward model, we pick one of the articles from the database at random as $\boldsymbol{\theta}_*$ and the linear reward model in (1) or use the labels to generate binary, one vs many rewards to simulate search for articles from a certain category. In order to come close to simulating a noisy setting, we used the logistic model, with $q_t = 1/(1 - \exp(-\langle \mathbf{x}_t, \boldsymbol{\theta}_* \rangle), P(y_t = +1) = q_t$.

Oracle Feedback. The support of one-vs-many sparse logistic regression is used to get an "oracle set of relevant features" for each class. Each word from the intersection of the support of an action and oracle relevant words was marked as relevant with probability $p'(= 0.1)$. In our theorem statements, p is the probability that the feature is present in a random action *and* it is marked relevant. This depends on the distribution of the words, but typically $\boldsymbol{p \in (0.001, 0.01)}$ and $k \in (30, 100)$ relevant features for each category. Figure 3, compares OFUL, Explore-then-commit and FF-OFUL on the 20NG dataset with oracle feedback. In these simulations averaged over 100 random $\boldsymbol{\theta}_*$, FF-OFUL outperforms OFUL and Explore-then-commit significantly. OFUL parameter was tuned to $\lambda = 2^8$.

Human Feedback. (Poulis and Dasgupta 2017) took 50 20Newsgroup articles from 5 categories and had users annotate relevant words. These are the same categories that we used above. This is closer to simulating human feedback since we are not using sparse logistic regression to estimate the sparse vectors. We take the user indicated relevant words instead as the relevance dimensions. There were $k \in (30, 100)$ relevant features for each category. In Figure 4(a), we can see that FF-OFUL is already outperforming OFUL and Explore-then-commit. This is despite the fact that it is not a very sparse regime. Surprisingly, we found



(a) Synthetic data (b) Real data with replacement (c) Real data without replacement

Figure 3: **(a)** Synthetic data with sparse $\boldsymbol{\theta}_*$ ($d = 40$, $k = 5$), FF-OFUL outperforms OFUL significantly. See Figure 1(b) for comparison with explore-then-commit strategy. Newsgroup dataset with oracle feedback: **(b)** This plot shows that FF-OFUL outperforms OFUL and Explore-then-commit when running in $d = 1000$ dimensions, sampling actions with replacement using binary rewards model. **(c)** sampling actions without replacement and using the numerical reward model. Smallest T_0 selected such that all relevant features are marked with high probability. Note shorter time horizon for without replacement sampling since T must be less than the number of actions.

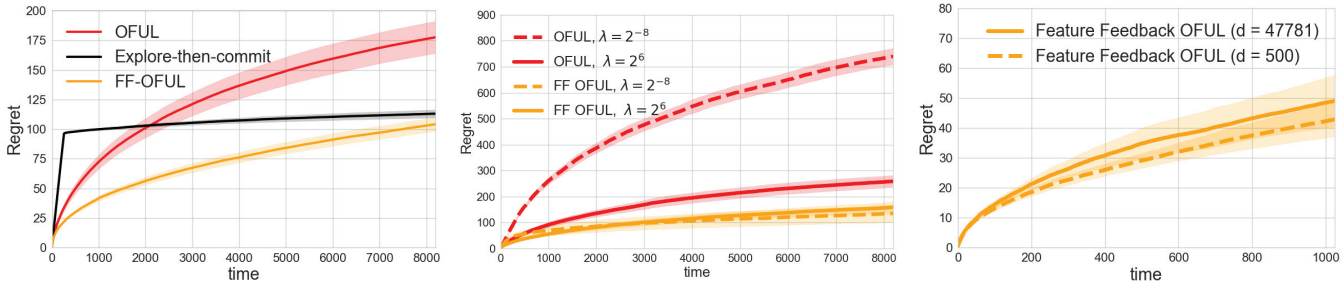


Figure 4: Newsgroup Dataset with Human Feedback: **(Left)** FF-OFUL outperforms OFUL and Explore-then-commit strategy in $d = 500$ dimensions. Both plots generated by tuning the parameter for OFUL. **(Center)** Sensitivity to tuning parameter λ seen by the drastic difference in performance of OFUL. In contrast, our FF-OFUL has a relatively modest difference in performance showing its robustness to the ridge regression parameter λ. **(Right)** Our algorithm for $d = 47781$ and $d = 500$ with ridge parameter $\lambda = 1$, showing its robustness to changes in dimensions and tuning.

that tuning had little effect on the performance of FF-OFUL whereas it had a significant effect on OFUL (see Figure 4). This is possibly due to the implicit regularization provided by gradually growing the number of dimensions as we receive new feedback. FF-OFUL also yields significantly better rewards at early stages by exploiting knowledge of relevant features as soon as they are identified, rather than waiting until all or most are found.

Parameter Tuning. For OFUL the ridge parameter (λ) is tuned from $\{2^i\}_{i=-7}^{10}$ to pick the one with best performance. All the tuned parameters selected for OFUL were strictly inside this range (for $d = 40, k = 5$, $\lambda = 2^{-5}$ and for $d = 10^3$ (Newsgroup), $\lambda = 2^8$). Figure 4(b) demonstrates the sensitivity of OFUL to change in tuning parameter. For FF-OFUL, the remarkable feature is that it does not require parameter tuning so $\lambda = 1$ for all experiments.

Full dimension experiments. Remarkably the performance of FF-OFUL barely drops in full ($d = 47781$) feature dimensions in Figure 4(c). Even though the ridge regression parameter (λ) for all experiments was not tuned and set to $\lambda = 1$. FF-OFUL is robust to changes in the ambient dimensions and the parameter λ. Recall that we do not compare with OFUL on 47781 dimensional data since it would require storing and updating a $d \times d$ matrix at each stage.

Conclusion

In this paper we provide an algorithm that incorporates feature feedback in addition to the standard reward feedback. The framework is based on the following insight: In the short-term horizon, when the algorithm does not have enough feedback, it starts with a small subset of the dimensions and gradually grows the number of dimensions as it receives new feedback. This has three benefits: (1) it makes it possible to use the new algorithm in high-dimensional settings where conventional linear bandits are impractical, (2) it is more robust to choice of parameters because it grows the feature dimensions over time which provides implicit regularization, and (3) it leads to better early-regret. The choice of the dimensions is based on feature feedback provided by the user. This could be generalized in a number of ways, using ideas from compressed sensing and/or dimensionality

reduction, alleviating the need for feature feedback from the user in the future. The goal of this framework is to motivate this idea of growing the feature space over time.

Acknowledgements This work was partially supported by AF grant FA8750-17-2-0262 and a grant from American Family Insurance.

References

Abbasi-Yadkori, Y.; Pal, D.; and Szepesvari, C. 2011. Improved Algorithms for Linear Stochastic Bandits. *Advances in Neural Information Processing Systems (NIPS)* 1–19.

Abbasi-Yadkori, Y.; Pal, D.; and Szepesvari, C. 2012. Online-to-Confidence-Set Conversions and Application to Sparse Stochastic Bandits. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Abe, N., and Long, P. M. 1999. Associative reinforcement learning using linear probabilistic concepts. In *Proceedings of the International Conference on Machine Learning (ICML)*, 3–11.

Auer, P., and Long, M. 2002. Using Confidence Bounds for Exploitation-Exploration Trade-offs. *Journal of Machine Learning Research* 3:2002.

Carpentier, A., and Munos, R. 2012. Bandit theory meets compressed sensing for high dimensional stochastic linear bandit. In *International Conference on Artificial Intelligence and Statistics*, 190–198.

Chen, Y.; Mac Aodha, O.; Su, S.; Perona, P.; and Yue, Y. 2018. Near-optimal machine teaching via explanatory teaching sets. In *International Conference on Artificial Intelligence and Statistics*, 1970–1978.

Croft, W. B., and Das, R. 1989. Experiments with query acquisition and use in document retrieval systems. In *Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval*, 349–368. ACM.

Dani, V.; Hayes, T. P.; and Kakade, S. M. 2008. Stochastic linear optimization under bandit feedback.

Deshpande, Y., and Montanari, A. 2012. Linear bandits in high dimension and recommendation systems. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 1750–1754. IEEE.

Druck, G.; Settles, B.; and McCallum, A. 2009. Active learning by labeling features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, 81–90. Association for Computational Linguistics.

Ginebra, J., and Clayton, M. K. 1995. Response surface bandits. *Journal of the Royal Statistical Society. Series B (Methodological)* 771–784.

Lang, K. 1995. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*. Elsevier. 331–339.

Lattimore, T., and Szepesvári, C. 2018. Bandit algorithms.

Poulis, S., and Dasgupta, S. 2017. Learning with feature feedback: from theory to practice. In *Artificial Intelligence and Statistics*, 1104–1113.

Raghavan, H.; Madani, O.; and Jones, R. 2006. Active learning with feedback on features and instances. *Journal of Machine Learning Research* 7(Aug):1655–1686.

Roads, B.; Mozer, M. C.; and Busey, T. A. 2016. Using highlighting to train attentional expertise. *PloS one* 11(1):e0146266.

Rusmevichientong, P., and Tsitsiklis, J. N. 2010. Linearly Parameterized Bandits. *Math. Oper. Res.* 35(2):395–411.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Yessenalina, A.; Yue, Y.; and Cardie, C. 2010. Multi-level structured models for document-level sentiment classification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 1046–1056. Association for Computational Linguistics.