CORe: Capitalizing On Rewards in Bandit Exploration

Nan Wang[1] **Branislav Kveton**[2] **Maryam Karimzadehgan**[2]

[1]Computer Science Dept., University of Virginia , Charlottesville, Virginia, USA
[2]Google Research , Mountain View, California, USA

Abstract

We propose a bandit algorithm that explores purely by randomizing its past observations. In particular, the sufficient optimism in the mean reward estimates is achieved by exploiting the variance in the past observed rewards. We name the algorithm **C**apitalizing **On** **Re**wards (CORe). The algorithm is general and can be easily applied to different bandit settings. The main benefit of CORe is that its exploration is fully data-dependent. It does not rely on any external noise and adapts to different problems without parameter tuning. We derive a $\tilde{O}(d\sqrt{n \log K})$ gap-free bound on the n-round regret of CORe in a stochastic linear bandit, where d is the number of features and K is the number of arms. Extensive empirical evaluation on multiple synthetic and real-world problems demonstrates the effectiveness of CORe.

1 INTRODUCTION

A *multi-armed bandit* [Lai and Robbins, 1985, Lattimore and Szepesvári, 2020] is an online sequential decision-making problem, where the learning agent chooses actions represented by arms in an n-round game. After an arm is pulled, the agent receives a *stochastic* reward generated from an unknown reward distribution associated with the arm. The goal of the agent is to maximize the expected n-round reward. As the agent needs to learn the mean rewards of the arms by pulling them, it faces the so-called exploitation-exploration dilemma: *exploit*, and pull the arm with the highest estimated mean reward thus far; or *explore*, and learn more about the arms.

A *stochastic linear bandit (or linear bandit)* [Rusmevichientong and Tsitsiklis, 2008, Abbasi-Yadkori et al., 2011] is a generalization of a multi-armed bandit where each arm is associated with a feature vector. The mean reward of an arm is the dot product of its feature vector and an unknown parameter vector, which needs to be learned by the agent. A multi-armed bandit can be considered as a special case of the linear bandit where the feature vector of each arm is a one-hot vector indicating the index of the arm, and the parameter vector is a vector of corresponding mean rewards.

Arguably, the most popular and well-studied exploration strategies for solving bandit problems are *Thompson sampling* (TS) [Thompson, 1933, Agrawal and Goyal, 2013] and *Optimism in the Face of Uncertainty (OFU)* [Auer et al., 2002]. TS maintains a posterior distribution over each arm's mean reward and samples from it to explore. This is efficient and has strong empirical performance when the posterior has a closed form [Chapelle and Li, 2011]. However, if the posterior does not have a closed form, as in many non-linear problems [McCullagh, 1984, Filippi et al., 2010], it needs to be approximated, which is typically computationally expensive and limits the applicability of TS [Gopalan et al., 2013, Abeille and Lazaric, 2016, Riquelme et al., 2018]. On the other hand, OFU-based algorithms [Auer et al., 2002] depend on the construction of high-probability confidence sets. They are theoretically near-optimal in multi-armed and linear bandits. However, as the confidence sets are often constructed for worst-case scenarios, they are empirically less competitive. In addition, in some problems, such as generalized linear bandits or neural network bandits [Zhou et al., 2020], it is only possible to approximate the confidence sets [Filippi et al., 2010, Zhang et al., 2016, Li et al., 2017]. These approximations affect the statistical efficiency of the algorithms and often perform poorly.

To design general algorithms that do not rely on problem-specific confidence sets or posteriors, recent works proposed randomized exploration [Baransi et al., 2014a, Osband and Roy, 2015, Kveton et al., 2019b,a, Vaswani et al., 2020]. The key idea is to randomize the reward history of the bandit algorithms before estimating the mean rewards. The randomization strategy is sufficiently general to apply to challenging problems, such as generalized linear bandits or neural network bandits. Bootstrapping [Eckles and Kaptein,

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

2014, Osband and Roy, 2015, Tang et al., 2015, Vaswani et al., 2018] is one of the randomization strategies, which uses the resampled reward history for mean reward estimation. However, exploration by bootstrapping has been poorly understood theoretically. Kveton et al. [2019b] showed that bootstrapping can suffer from linear regret in certain bandit instances and proposed to add pseudo rewards to each arm's reward history before bootstrapping. They proved that the pseudo rewards provide sufficient variance for exploration. Kim and Tewari [2019], Kveton et al. [2019a] further showed that the sufficient variance can be induced by other randomization schemes, which they analyzed. Unfortunately, all the analyses rely on the right amount of external noise or pseudo rewards that match the problem instances. In real-world problems, however, we often do not have prior knowledge of the variance of the reward distributions. Thus the external noise and pseudo rewards are hard to design.

In this work, we propose a general randomized exploration strategy without adding any external noise or pseudo rewards. Specifically, we take advantage of the randomness in the agent's past observed rewards from all arms for exploration. In each round, the learning agent adds to each arm's history the rewards sampled from past observations of all the arms, and pulls the arm with the highest estimated mean reward based on the perturbed histories. We call the resulting algorithm `CORe`, meaning **C**apitalizing **O**n **Re**wards. As `CORe` only relies on past observed rewards, its exploration is *data-dependent*. With a well designed sampling strategy, the observed rewards from all arms provide enough variance for exploration, without the need of knowing the actual reward distributions of the arms. Thus the exploration adapts to different problems without parameter tuning. This is a significant advantage in real-world applications, where we often have no knowledge of the actual reward distributions.

We make the following contributions. First, we propose a randomized exploration strategy that does not rely on any external noise. We show that the new algorithm `CORe` ensures proper variance for exploration by sampling from the past observed rewards, agnostic to the variance of reward distributions. Second, we analyze `CORe` in a linear Gaussian bandit and derive a $\tilde{O}(d\sqrt{n \log K})$ gap-free bound on its n-round regret, where d is the dimension of feature vectors and K is the number of arms. Although we assume Gaussian noise in the analysis, we observe empirically that `CORe` works well when the reward noise is not Gaussian and varies significantly across the arms. Finally, we conduct comprehensive experiments on both synthetic and real-world problems that demonstrate the effectiveness of `CORe`.

2 SETTING

We use the following notation throughout the paper. The set $\{1, 2, ..., n\}$ is denoted by $[n]$. We denote by $u \oplus v$ the concatenation of vectors u and v. We use I_d to denote a $d \times d$ identity matrix, and write $\tilde{O}$ for the big-O notation up to polylogarithmic factors in n.

A *stochastic linear bandit* [Rusmevichientong and Tsitsiklis, 2008, Abbasi-Yadkori et al., 2011] is an online learning problem where the learning agent sequentially pulls K arms in n-rounds and each arm is associated with a d-dimensional feature vector. We denote by $x_i \in \mathbb{R}^d$ the feature vector of arm $i \in [K]$ and $\theta_* \in \mathbb{R}^d$ is the unknown parameter vector. The *reward* of arm i in round $t \in [n]$, $Y_{i,t}$, is drawn i.i.d. from the reward distribution of arm i, P_i, with mean $\mu_i = x_i^\top \theta_*$. In round t, the learning agent pulls arm $I_t \in [K]$ and receives the reward $Y_{I_t,t}$. To have a more compact notation, we denote by $X_t = x_{I_t}$ and $Y_t = Y_{I_t,t}$ the feature vector of the pulled arm in round t and its observed reward, respectively. The agent does not know the mean rewards or the parameter vector in advance and learns them by pulling the arms. The goal of the agent is to maximize its *expected cumulative reward* in n rounds. In particular, when x_i is a K-dimensional one-hot vector with $x_i = e_i$, $i \in [K]$, and θ_* is a vector of K mean rewards, the linear bandit reduces to a *multi-armed bandit* [Lai and Robbins, 1985, Lattimore and Szepesvári, 2020].

Without loss of generality, we assume that arm 1 is optimal, meaning $\mu_1 > \max_{i>1} \mu_i$. We denote by $\Delta_i = \mu_1 - \mu_i$ the *gap* of arm i, which is the difference between the mean rewards of arms 1 and i. Maximizing the expected n-round reward is equivalent to minimizing the *expected n-round regret*, which is defined as

$$R(n) = \sum_{i=2}^{K} \Delta_i \mathbb{E}\left[\sum_{t=1}^{n} \mathbb{1}\{I_t = i\}\right].$$

We make the following standard assumptions in this setting. First, the mean reward $\mu_i = x_i^\top \theta_*$ for any arm $i \in [K]$ is bounded, and without loss of generality, we assume that it is in $[0, 1]$. Second, the feature vectors of the first d arms are a basis in $\mathbb{R}^d$. This is without loss of generality for the following reason. Since the feature vectors of arms are known in advance, a basis of size d can be found by standard methods from linear algebra, as long as it exists. Then we can exchange it for the first d feature vectors. If it does not exist, because the feature space is $d' < d$ dimensional, we can use SVD to reduce the feature space to d' dimensions where a basis of size d' exists. Note that this reduction will not change the linearity of the expected reward in features and the optimal arm.

3 CAPITALIZING ON REWARDS IN BANDIT EXPLORATION

In this section, we introduce the new algorithm Capitalizing On Rewards (`CORe`). We first illustrate key ideas of `CORe` and discuss how it works in Section 3.1. In Section 3.2, we

instantiate the algorithm in a stochastic linear bandit. To be more specific, in the rest of the paper, we use CORe to refer to the algorithm applied in a multi-armed bandit, and use LinCORe to represent the algorithm in a linear bandit.

3.1 KEY IDEAS AND INFORMAL JUSTIFICATION OF CORe

The principle of CORe is to utilize the variance in the past observed rewards to incentivize exploration. We first discuss CORe in a simple multi-armed bandit to illustrate how it works. Specifically, when estimating the mean reward of arm i in round t, CORe first perturbs each reward of arm i with a reward sampled from all observed rewards in the past $t-1$ rounds. Then the mean of arm i is estimated based on its perturbed rewards. Thus if there is sufficient variance in the past observed rewards, CORe is able to overestimate the mean rewards of arms to achieve optimism.

To be more concrete, we make an analogy between CORe and TS. For example, in a *Gaussian bandit*, adding additive noise to the mean reward estimate is equivalent to posterior sampling. In particular, fix arm i and let the number of its pulls be s. Let $\mu_i \sim \mathcal{N}(\mu_0, \sigma^2)$ be the mean reward of arm i, where $\mathcal{N}(\mu_0, \sigma^2)$ is the Gaussian prior in TS and σ^2 is the variance of the arm's reward distribution. Let $(Y_{i,\ell})_{\ell=1}^s \sim \mathcal{N}(\mu_i, \sigma^2)$ be s i.i.d. noisy observations of μ_i. Then the posterior distribution of μ_i conditioned on $(Y_{i,\ell})_{\ell=1}^s$ is

$$\mathcal{N}\left(\frac{\mu_0 + \sum_{\ell=1}^s Y_{i,\ell}}{s+1}, \frac{\sigma^2}{s+1}\right). \tag{1}$$

It is well known that sampling from this distribution in TS leads to near-optimal regret [Agrawal and Goyal, 2013]. From another perspective, sampling a mean reward of arm i as above is equivalent to adding i.i.d. Gaussian noise to μ_0 and each reward in $(Y_{i,\ell})_{\ell=1}^s$, and then taking the average [Kveton et al., 2019a]. Specifically,

$$\frac{\mu_0 + Z_0 + \sum_{\ell=1}^s (Y_{i,\ell} + Z_\ell)}{s+1}$$

is a sample from distribution (1) for $(Z_\ell)_{\ell=0}^s \sim \mathcal{N}(0, \sigma^2)$.

However, in practice, σ^2 depends on the specific problem instance and is unknown. Thus the variance of $(Z_\ell)_{\ell=0}^s$ needs to be carefully tuned to match σ^2. The key insight in CORe is that the exact value of σ^2 does not have to be known. Instead of sampling the noise from a given distribution, CORe samples $(Z_l)_{\ell=0}^s$ from a reward pool, which is composed of previously observed rewards of all arms. Then CORe adds sampled rewards to rewards of arm i for mean reward estimate. As we show in Section 4, after an initialization period of $\frac{4 \log n}{z - 1 - \log z} + 1$ rounds, for any $z \in (0, 1)$, the empirical variance of the observed rewards so far is at least $z\sigma^2/2$ with a high probability. Thus, after scaling the rewards by α to construct the reward pool, the variance of an i.i.d. sampled reward is greater than $\alpha^2 z \sigma^2 / 2$. This is at least σ^2 for $\alpha^2 z > 2$, and can be achieved without knowing σ^2.

3.2 CAPITALIZING ON REWARDS IN A STOCHASTIC LINEAR BANDIT

We present the algorithm in a stochastic linear bandit (LinCORe) in Algorithm 1, as it is a more general setting than a multi-armed bandit. In round t, LinCORe first constructs a *reward pool* $\mathcal{R}_t$ from all the past $t-1$ observed rewards. To achieve optimism in the mean reward estimation in round t, each reward Y_ℓ observed from a pulled arm with feature vector X_ℓ is perturbed by a randomly sampled reward Z_ℓ from $\mathcal{R}_t$. Then we fit a linear model to the perturbed rewards (line 11),

$$\tilde{\theta}_t \leftarrow G_t^{-1} \sum_{\ell=1}^{t-1} X_\ell \Big[Y_\ell + Z_\ell \Big], \tag{2}$$

where

$$G_t \leftarrow \sum_{\ell=1}^{t-1} X_\ell X_\ell^\top + \lambda I_d \tag{3}$$

is the sample covariance matrix up to round t and $\lambda > 0$ is the regularization parameter. $(Z_\ell)_{\ell=1}^{t-1}$ are i.i.d. rewards freshly sampled in each round from $\mathcal{R}_t$. The estimate of the mean reward of arm i is $x_i^\top \tilde{\theta}_t$. The arm with the highest reward estimate is pulled. This is similar to Thompson sampling [Thompson, 1933, Agrawal and Goyal, 2012, Abeille and Lazaric, 2016] and perturbed history exploration [Kveton et al., 2019a, 2020] in linear bandits, which add noise to the parameter estimate. However, LinCORe does not depend on any posterior variance or external pseudo rewards for exploration, and instead only relies on randomness in the agent's own reward history.

Specifically, in lines 1-3 of Algorithm 1, we initialize LinCORe by pulling arms sequentially for the first $\max\{d, \frac{4 \log n}{z - 1 - \log z} + 1\}$ rounds, where $z \in (0, 1)$ is a tunable parameter that determines the initial variance in the reward pool. Each arm in the d-dimensional basis is initially pulled and this guarantees sufficient diversity of feature vectors in the reward pool. This is needed because our exploration is purely data-driven. After initialization, in each round t, LinCORe processes the past $t-1$ rewards and creates a new reward pool $\mathcal{R}_t$ in lines 5-8. LinCORe scales the rewards by α to guarantee sufficient variance in $\mathcal{R}_t$ for exploration, as suggested in Section 3.1. Besides, the processed rewards in $\mathcal{R}_t$ are centered to have zero mean and each reward y has its symmetric reward $-y$ around zero in the pool. This additional processing is only to simplify the theoretical analysis in Section 4. It does not change the variance of samples from the reward pool and LinCORe performs in practice similarly without it. LinCORe then samples $t-1$ i.i.d. rewards from

Algorithm 1 Capitalizing on Rewards in a stochastic linear bandit (`LinCORe`)

Input: Initial variance ratio $z \in (0, 1)$, sample scale ratio $\alpha \in \mathbb{R}^+$, number of rounds n

1: **for** $t = 1, 2, ..., n$ **do**
2: **if** $t \leq \max\left\{d, \frac{4 \log n}{(z - 1 - \log z)} + 1\right\}$ **then**
3: $I_t \leftarrow t \bmod K$
4: **else**
5: $\mathcal{R}_t \leftarrow ()$
6: $\mu(\mathcal{R}_t) = \frac{1}{t-1} \sum_{\ell=1}^{t-1} Y_\ell$
7: **for** $\ell = 1, \ldots, t-1$ **do**
8: $\mathcal{R}_t \leftarrow \mathcal{R}_t \oplus (\alpha(Y_\ell - \mu(\mathcal{R}_t)),\ \alpha(\mu(\mathcal{R}_t) - Y_\ell))$
9: $(Z_\ell)_{\ell=1}^{t-1} \leftarrow$ Sample $t-1$ i.i.d. rewards from $\mathcal{R}_t$
10: $G_t \leftarrow \sum_{\ell=1}^{t-1} X_\ell X_\ell^\top + \lambda I_d$
11: $\tilde{\theta}_t \leftarrow G_t^{-1} \sum_{\ell=1}^{t-1} X_\ell \Big[Y_\ell + Z_\ell \Big]$
12: $I_t \leftarrow \arg\max_{i \in [K]} x_i^\top \tilde{\theta}_t$
13: Pull arm I_t and get reward Y_t

$\mathcal{R}_t$ (line 9). To get the parameter estimate $\tilde{\theta}_t$, `LinCORe` perturbs each observed reward by a sampled reward from $\mathcal{R}_t$ to fit a linear model (lines 10-11). Finally, `LinCORe` pulls arm I_t with highest mean reward estimate from $\tilde{\theta}_t$ and observes its reward Y_t. It is important to note that Algorithm 1 is only an instance of the proposed general randomization strategy in a linear bandit setting. The parameter estimation in lines 10-12 (Algorithm 1) can be replaced by any other estimator, such as a neural network, to get more general algorithms. For example, we only need to replace the parameter estimation in lines 10-12 with that of the neural network used for reward estimation.[1] Here we choose to show the linear case rather than a general case to be more concrete for reproducibility. Besides, when feature vectors are one-hot vectors with $x_i = e_i$, Algorithm 1 corresponds to `CORe` in a multi-armed bandit, which is essentially using the average of each arm's perturbed rewards as the mean reward estimate.

The exploration in `LinCORe` arises from the variance in $\mathcal{R}_t$. For example, if the reward distributions of all arms are Gaussian with variance of σ^2, we want a comparable variance in $\mathcal{R}_t$, so that the sampled rewards from $\mathcal{R}_t$ can offset unfavorable reward histories. To achieve this, `LinCORe` initially pulls arms sequentially $\max\{d, \frac{4 \log n}{z - 1 - \log z} + 1\}$ times, to accumulate observations. We prove in Section 4.2 that after this initialization, the empirical variance of observed rewards is at least $z\sigma^2/2$ with a high probability. However, $z\sigma^2/2$ may not be sufficient for effective exploration. Once z is fixed, the *scale ratio* α dictates the multiplicative factor of the variance of each sampled reward in the reward pool, and thus controls the trade-off between exploration and exploitation. Larger α leads to more exploration. More importantly, the variance in $\mathcal{R}_t$ is at least $\alpha^2 z/2$ of that of the reward distributions. So it is automatically adapted to the problem.

[1] Note that this implementation can be computationally costly, as it requires retraining of the neural network in each round. This can be alleviated as in Lu and Van Roy [2017].

4 REGRET ANALYSIS

We analyze `LinCORe` in the setting where the reward distribution of each arm $i \in [K]$ is $\mathcal{N}(\mu_i, \sigma^2)$ for $\mu_i \in [0, 1]$. The variance of reward distributions is σ^2, identical for all arms. In this setting, we derive the following gap-free bound on the n-round regret of `LinCORe`.

Theorem 1. *For any* $1/2 \leq z < 1$, $4\sqrt{\sigma^2 \log n} \geq 1$, *and* $n \geq 24$, *the expected* n*-round regret of* `LinCORe` *is*

$$R(n) = \tilde{O}(d\sqrt{n \log K})$$

for $\alpha = O(\sqrt{z^{-1} d \log n})$. We provide the detailed proof in Appendix A.2.

The analysis of `LinCORe` follows the general steps of analyzing `LinTS`, as presented by Kveton et al. [2019b]. Our main novelty is the analysis of concentration and anti-concentration properties of the empirical noise distribution in the reward pool, which is the key to exploration in `CORe`. This is challenging because that distribution does not have a known closed form. In Section 4.2, we show that the rewards have sufficient variance in Lemma 2. We bound their magnitude in Lemma 3. The rest of our analysis follows the outline of `LinPHE` [Kveton et al., 2020], which we generalize from Bernoulli perturbations to those in Section 4.2.

4.1 DISCUSSION

The regret of `LinCORe` is $\tilde{O}(d\sqrt{n \log K})$ (Theorem 1), where d is the number of features and K is the number of arms. This is on the same order as the regret bound of `LinPHE` [Kveton et al., 2020], a state-of-the-art randomized algorithm for linear bandits. In the infinite arm setting, Abeille and Lazaric [2016] proved that the regret of `LinTS` is $\tilde{O}(d^{\frac{3}{2}}\sqrt{n})$, which we also match. Specifically, if the space of arms was discretized on an ε-grid, the number of arms would be $K = \varepsilon^{-d}$ and $\sqrt{\log K} = \sqrt{d \log(1/\varepsilon)}$.

The key idea in our analysis is to inflate α in the reward pool to achieve optimism. In linear bandits, this idea can be traced to Agrawal and Goyal [2012]. Roughly speaking, $\alpha = O(\sqrt{z^{-1} d \log n})$. This setting is too conservative in practice. Therefore, we experiment with less conservative settings and relatively small α in Section 5.

4.2 REWARD POOL

The exploration in `LinCORe` is enabled by the variance of sampled rewards from the reward pool $\mathcal{R}_t$. In this section,

we analyze the variance of sampling i.i.d. rewards from $\mathcal{R}_t$, which lays the foundation for the theoretical analysis of LinCORe. We use $\sigma^2(\mathcal{R}_t)$ to represent the variance of one i.i.d. sampled reward from $\mathcal{R}_t$. Specifically, the rewards in $\mathcal{R}_t$ are simple transformations of all the past $t-1$ observed rewards $(Y_{I_\ell,\ell})_{\ell=1}^{t-1}$ (lines 6-8 in Algorithm 1). $\sigma^2(\mathcal{R}_t)$ is algebraically equivalent to the variance of one sampled reward from $(Y_{I_\ell,\ell})_{\ell=1}^{t-1}$ scaled by α^2,

$$\sigma^2(\mathcal{R}_t) = \frac{1}{|\mathcal{R}_t|}\sum_{y\in\mathcal{R}_t} y^2 = \frac{\alpha^2}{t-1}\sum_{\ell=1}^{t-1}(Y_{I_\ell,\ell} - \mu(\mathcal{R}_t))^2, \tag{4}$$

where $\mu(\mathcal{R}_t)$ is the mean of all past rewards observed by the learning agent, as defined in line 6 of Algorithm 1. Thus a sampled reward from $\mathcal{R}_t$ can provide the variance of $\sigma^2(\mathcal{R}_t)$. We characterize $\sigma^2(\mathcal{R}_t)$ by the following two lemmas, which are proved in Appendix A.4.

Lemma 2. *For any $n \geq 2$ and $z \in (0,1)$, $\sigma^2(\mathcal{R}_t) \geq \frac{\alpha^2 z}{2}\sigma^2$ with probability of at least $1-\frac{1}{n}$, jointly for all rounds $t > \frac{4\log n}{z-1-\log z} + 1$.*

Lemma 2 states that when there are enough rewards in $\mathcal{R}_t$ after the initialization, the variance of sampling a reward from $\mathcal{R}_t$ is $\Omega(\sigma^2)$ with a high probability, which provides the variance needed for exploration. On the other hand, the variance should not be too large, which would hurt the convergence of mean reward estimates. Lemma 3 shows that the rewards in $\mathcal{R}_t$ are bounded with high probability.

Lemma 3. *For any $n \geq 2$ and $t \leq n$, with probability of at least $1-\frac{1}{n}$, the absolute values of the rewards in reward pool $\mathcal{R}_t$ are bounded by $\alpha(4\sqrt{\sigma^2\log(n)}+1)$.*

In particular, in Lemma 2, the lower bound on $\sigma^2(\mathcal{R}_t)$ ensures the overestimate of the mean reward estimate for exploration. Lemma 3 states that the magnitude of rewards in $\mathcal{R}_t$ is bounded with a high probability. Lemmas 2 and 3 provide the justification of using the agent's past observed rewards for effective exploration in LinCORe, and are applied throughout the proof of Theorem 1 in Appendix A.2.

5 EXPERIMENTS

We evaluate our proposed algorithm empirically in both multi-armed and linear bandits. In all experiments, we use CORe to denote the proposed algorithm in the multi-armed bandit setting, and LinCORe in the linear case. We compare it with several state-of-the-art baselines and show how it adapts to different problems without parameter tuning. In Section 5.1, we evaluate CORe in multi-armed bandits. We experiment with LinCORe in linear bandits in Section 5.2 and investigate the robustness of its parameters in Section 5.3. Finally, we generalize CORe to a learning to rank problem to evaluate its performance in real-world problems.

5.1 MULTI-ARMED BANDIT

We evaluate CORe in three classes of multi-armed bandit problems. The first class is Bernoulli bandits where $P_i = \mathrm{Ber}(\mu_i)$. The second class is beta bandits where $P_i = \mathrm{Beta}(v\mu_i, v(1-\mu_i))$ with $v = 4$. The third class is Gaussian bandits where $P_i = \mathcal{N}(\mu_i, \sigma^2)$ with $\sigma = 0.5$. Each bandit problem has $K = 10$ arms and the mean rewards are chosen uniformly at random from $[0.25, 0.75]$. The horizon of each experiment is $n = 10,000$ rounds. We experiment with 100 randomly chosen problems in each class and report the average regret.

We compare CORe to six baselines: UCB1 [Auer et al., 2002], UCB-V [Audibert et al., 2009], TS [Agrawal and Goyal, 2013], PHE [Kveton et al., 2019a], NP-TS [Riou and Honda, 2020] and SSMC [Chan, 2019]. UCB-V can estimate the variance of the reward distribution based on the observed rewards, which automatically adapts to the variance. NP-TS and SSMC are two non-parametric solutions proposed in the multi-armed bandit setting. In particular, NP-TS is a non-parametric randomized algorithm. At each step, it computes an average of the observed rewards with random weights. SSMC is a non-parametric arm allocation procedure inspired by sub-sampling approaches [Baransi et al., 2014a]. For TS, we use Bernoulli TS (Ber-TS) with a $\mathrm{Beta}(1,1)$ prior for Bernoulli and beta bandits. We use Gaussian TS (Gauss-TS) with a $\mathcal{N}(0.5, \sigma^2)$ prior for Gaussian bandits [Agrawal and Goyal, 2013], where the parameter σ is set to match the variance of the actual reward distribution. PHE belongs to the same class of bandit algorithms as CORe that randomize the reward history for exploration. We do not include Giro [Kveton et al., 2019b] as PHE explores similarly but in a more efficient way. We add Bernoulli pseudo rewards in PHE (Ber-PHE) in Bernoulli and beta bandits and set the parameter a to values that achieve the best performance as reported in [Kveton et al., 2019a]. For Gaussian bandit, we add Gaussian pseudo rewards (Gauss-PHE) as suggested in the paper. To tune baselines, we do grid search for their hyperparameters in a sufficiently wide range, so that the minimum regret can be attained in it. For example, to tune Gauss-PHE, we search for the standard deviation of the Gaussian pseudo rewards in $[0.1, 2]$ and tune parameter a in the range of $[0.1, 2]$, both with step size of 0.1. Since the baselines and experimental settings are relatively standard, we do not describe all of them in detail. To set α and z in CORe, we initially run CORe in Bernoulli bandits with the settings described above. Then we choose the parameters $\alpha = z = 0.6$ that minimize the regret over 100 runs. We fix these parameters for all other experiments without any further tuning.

Our results are reported in Figure 1. We show the cumulative regret as a function of the number of rounds. CORe achieves strong empirical performance that is comparable to or better than all the baselines. In particular, CORe outperforms UCB1 and UCB-V in all three classes of bandit problems. Al-

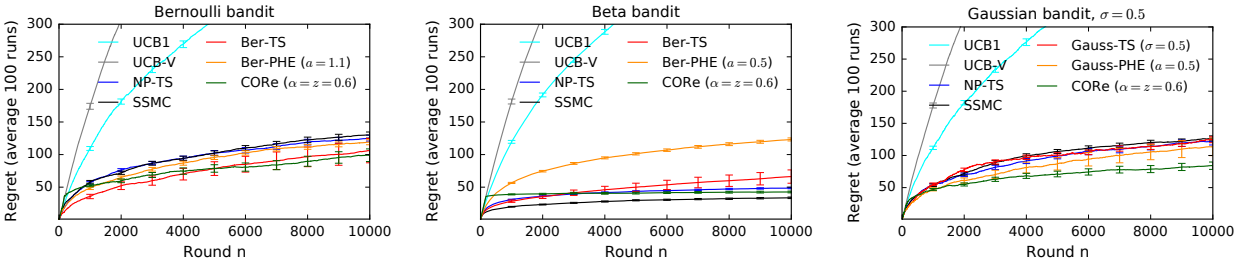


Figure 1: Comparison between CORe and the baselines in Bernoulli, beta and Gaussian multi-armed bandits. x-axis is the number of rounds. y-axis is the cumulative regret (lower the better). All results are averaged over 100 randomly chosen problems and error bars represent the standard deviation over the runs.

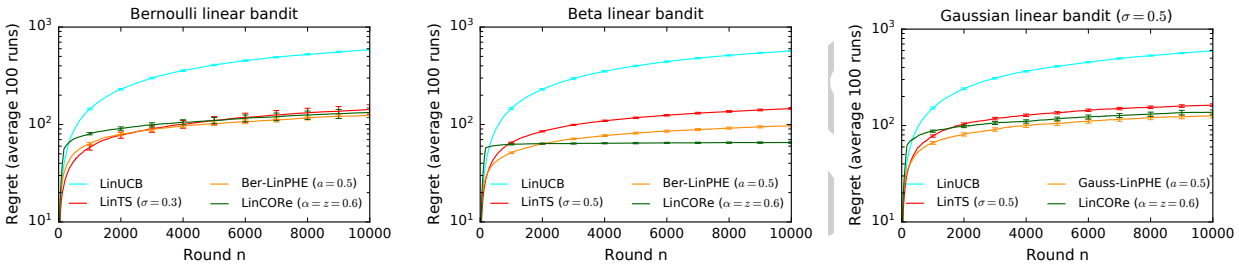


Figure 2: Comparison between LinCORe and the baselines in linear bandits. The cumulative regret is displayed on log scale to include all baselines.

though UCB-V estimates the variance of observed rewards to explore, it is too conservative and performs poorly in practice. TS and PHE can have similar performance as CORe, but the variance of the posterior (parameter σ) in TS and the perturbation scale in PHE (parameter a) are tuned based on the knowledge of the specific bandit problems, which is usually not accessible in real-world scenarios. In contrast, CORe consistently performs well in different problems without tuning the parameters. This is a significant advantage in real-world applications when the reward distribution is unknown. NP-TS and SSMC also achieve strong performance in multi-armed bandits, but they do not generalize to structured problems as CORe does.

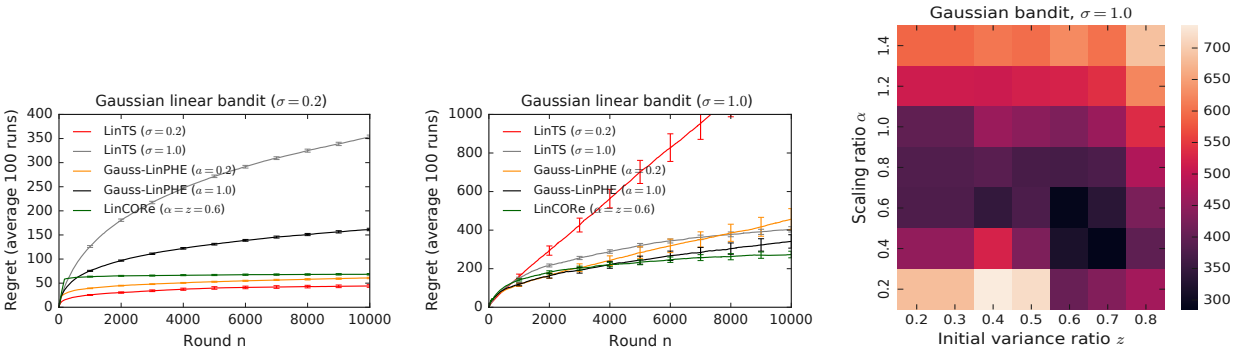
5.2 LINEAR BANDIT

We evaluate LinCORe in several linear bandit problems. We set the number of arms to $K = 50$ and the dimension of the feature vectors to $d = 10$. We follow the generation of feature vectors and the parameter vector θ_* in [Kveton et al., 2020] (see their Section 5.1). Following the experiments in Section 5.1, we consider Bernoulli, beta, and Gaussian reward distributions by setting the mean reward of each arm to $x_i^\top \theta_* \in [0, 1]$. The horizon of each experiment is $n = 10,000$ rounds and we report the average results over 100 randomly chosen problems.

We compare LinCORe with LinUCB [Abbasi-Yadkori et al., 2011], LinTS [Agrawal and Goyal, 2012], and LinPHE [Kveton et al., 2020]. There are no linear variants of UCB-V, NP-TS or SSMC. For LinPHE, we add Bernoulli pseudo rewards in Bernoulli and beta bandit (Ber-LinPHE), and add Gaussian rewards in Gaussian bandit (Gauss-LinPHE). The parameters for LinTS and LinPHE are tuned in the range of $[0.1, 2]$, while we still use the same parameters $\alpha = z = 0.6$ for LinCORe as in Section 5.1. The results are shown in Figure 2. In all three classes of linear bandit problems, LinCORe can achieve the best performance without tuning the parameters. Note that unlike LinUCB and LinTS, whose upper confidence sets and posterior need to be designed differently for multi-armed bandits and linear bandits, LinCORe is simply applying the same randomization strategy to different bandit settings. Although LinPHE is also a direct generalization of the multi-armed bandit setting, its perturbation from pseudo rewards depends on the knowledge of the arms' reward distribution.

5.3 ADAPTATION TO PROBLEM HARDNESS

We further investigate how LinCORe automatically adjusts its exploration in problems with different levels of hardness. Besides, we also show that LinCORe works properly with a wide range of parameters. Specifically, we consider linear Gaussian bandits with different levels of variance. We set the standard deviation of the reward distributions to $\sigma = 0.2$ as an easy problem, and set $\sigma = 1$ as a hard



(a) Easy problem (b) Hard problem (c) Tuning parameters of LinCORe

Figure 3: In the first two figures, we compare LinCORe, LinPHE, and LinTS in easy and hard problems. We fix the parameters of LinCORe and tune the parameters of LinPHE and LinTS to perform well in either the easy or the hard case. In the third figure, we tune the parameters of LinCORe and show its cumulative regret in 10,000 rounds. All results are averaged over 100 runs.

problem. We compare LinCORe to LinTS and LinPHE who achieve similar performance in Section 5.2. For LinTS and LinPHE, we use two sets of parameters for each of them, with each set specially tuned for either the easy or the hard problem. In particular, for LinTS we set σ to 0.2 for the easy problem and 1.0 for the hard problem that performs well in two problems correspondingly. In LinPHE, we tune the parameter a and set it to 0.2 and 1.0 for the easy and the hard problem, respectively. We still use the same fixed parameters as in Sections 5.1 and 5.2 in LinCORe for both problems. As shown in Figures 3a and 3b, LinCORe is able to perform well in both easy and hard problems without tuning the parameters. For LinTS and LinPHE, they can achieve equally good performance as LinCORe when the parameters are specially set for the problems. However, the parameters tuned for the easy problem under-explore in the hard problem and have almost linear regret. Similarly, the parameters tuned for the hard problem explore too much in the easy problem, and converge slowly.

We further tune the parameters α and z of LinCORe in the hard problem in Figure 3c to see how it performs under different combinations of parameters. The results show that LinCORe works well under a wide range of parameters and thus is easy to configure. For example, the area of $\alpha \in [0.4, 0.8]$ and $z \in [0.5, 0.7]$ provides similarly competitive performance. When α and z are too small, such as $\alpha = z = 0.2$, LinCORe mainly exploits and explores too little to find the optimal arm. On the other hand, when α and z are too large, such as $\alpha = 1.4$ and $z = 0.8$, it over-explores and suffers from high regret. Moreover, it is worth noting that when setting z to a large value, we have a large number of random pulls for initialization in order to have a high variance in the reward pool, which also leads to high regret in the early stage.

5.4 ONLINE LEARNING TO RANK

We finally evaluate CORe in a real-world problem, online learning to rank [Liu, 2009, Radlinski et al., 2008]. Online learning to rank is a sequential decision-making problem where the learning agent repeatedly recommends a list of items. In round t, the learning agent recommends a ranked list of K items out of all $L \geq K$ items. The user clicks on the recommended items. The clicks are treated as bandit feedback. The performance of the agent is measured by the expected cumulative regret, which is the expected loss in clicks relatively to the optimal ranking. The goal of this experiments is to showcase CORe in a challenging bandit problem, where the action space is large (any ranked list of items) and feedback is complex (clicks on lower ranked items are less likely than on the higher ones).

We experiment with the *Yandex* dataset and follow the experimental setup as in [Zoghi et al., 2017, Lattimore et al., 2018]. In each query, the user is shown 10 documents and the search engine records clicks of the user. We use the 60 most frequent queries from the dataset and learn their cascade models (CM) with PyClick [Chuklin et al., 2015]. The goal of the learning agent is to rerank $L = 10$ most attractive items to maximize the expected number of clicks at the first $K = 5$ positions. The application of bandit algorithms is similar to the multi-armed bandit setting, despite that the agent ranks the items based on their mean reward estimates rather than selecting a single item. The corresponding cascade model learned under each query is used to generate clicks. We experiment with a horizon of $n = 50,000$ rounds and the regret is averaged over 10 runs.

We compare CORe to CascadeKL-UCB [Kveton et al., 2015], which is specifically designed for online learning to rank in the cascade model. We also evaluate Ber-TS and Ber-PHE in this problem. They are applied in the same way as

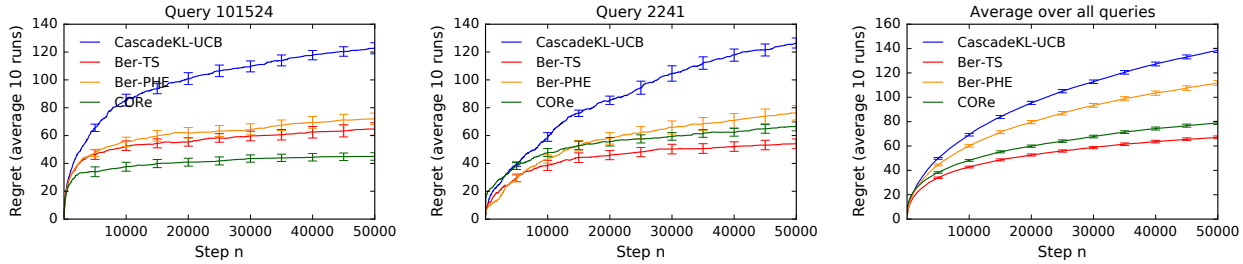


Figure 4: The cumulative regret of different algorithms in the learning to rank problem. The results are averaged over 10 runs per query. We sample two queries to demonstrate the performance in the first two figures, and display the results averaged over all queries in the third figure.

CascadeKL-UCB, with the UCB of each item replaced by its TS or PHE mean reward estimate. TopRank [Lattimore et al., 2018] is another algorithm for online learning to rank based on topological sort. It tends to perform worse than CascadeKL-UCB in the cascade model [Lattimore et al., 2018] and thus we do not include it. We still use the default parameters for CORe ($\alpha = z = 0.6$) and set $a = 0.5$ in Ber-PHE. The results are presented in Figure 4, where we show the results for two specific queries in the first two figures and the average performance over all queries in the third figure. Under the default parameters, CORe already achieves competitive performance that consistently outperforms Ber-PHE and CascadeKL-UCB across queries, and is comparable to Ber-TS. We also observed further improvement of CORe if tuned, to $\alpha = z = 0.4$, which achieves almost the same performance as Ber-TS when averaging over all queries. The promising results from this experiment demonstrate the wide applicability and robustness of CORe in real-world structured problems, and its ability in solving a new problem without prior knowledge.

6 RELATED WORK

The key to statistically-efficient exploration in stochastic bandits is to perturb the mean reward estimates of arms sufficiently. Algorithms based on upper confidence bounds (UCBs) [Auer et al., 2002, Abbasi-Yadkori et al., 2011] perturb the mean reward estimates by adding high-probability confidence intervals to them. The confidence intervals are constructed by theory. Although theoretically optimal, they are often conservative in practice, because they are designed for hardest problem instances. UCB-V [Audibert et al., 2009] is a variant of UCB1 that adapts confidence intervals using an empirical estimate of the variance from observed rewards. This algorithm also tends to be conservative in practice, as we show in Section 5.1.

Posterior sampling [Thompson, 1933, Agrawal and Goyal, 2013] perturbs mean reward estimates by sampling from posterior distributions. To be statistically efficient, proper variance needs to be specified in the posterior updates, which is often unknown in real-world problems. As we show in Section 5.3, when the variance of the posterior in Gauss-LinTS is misspecified, the algorithm suffers from high regret, due to either under- or over-exploration. CORe is closely related to posterior sampling in Gaussian bandits (Section 3.1). However, instead of relying on knowing the variance of reward distributions, it utilizes the randomness in the agent's observed rewards, to have its data-dependent exploration that adapts to problem hardness.

Randomized exploration algorithms, such as Giro [Kveton et al., 2019b] and PHE [Kveton et al., 2019a], add pseudo-rewards to the reward history and use the perturbed mean reward estimates for arm selection. The added pseudo-rewards add sufficient variance for exploration and lead to provably sublinear regret in multi-armed bandits. However, similarly to UCB designs and posterior sampling, the right amount of perturbation is needed to explore at a near-optimal rate. In contrast, instead of adding external noise from pseudo rewards, CORe samples from the agent's past observed rewards to induce exploration. This provides sufficient variance when all reward distributions have comparable variance and we analyze LinCORe theoretically in the setting of identical Gaussian noise.

The idea of efficient exploration with no prior knowledge on the arms' distribution has emerged in recent years. Non-parametric solutions have been proposed in the multi-armed bandit setting. The most representative works are non-parametric Thompson sampling (NP-TS) [Riou and Honda, 2020] and subsample-mean comparison (SSMC) [Chan, 2019]. Specifically, NP-TS proposes a generalization of the Bernoulli Thompson sampling to multinomial distributions, and a non-parametric adaption of this algorithm. SSMC is inspired from the sub-sampling approaches [Baransi et al., 2014b] and is asymptotically optimal for exponential-family distributions. We compare them with CORe in the multi-armed bandit setting in Section 5.1. However, it is unclear how to generalize NP-TS and SSMC to linear bandits.

7 CONCLUSIONS

We propose a new online algorithm, capitalizing on rewards (CORe), that explores by utilizing the randomness of the agent's past observed rewards. In particular, CORe samples rewards from a well designed reward pool from the agent's past observations to perturb the reward histories. The variance introduced by sampled rewards automatically adapts to the noise of the reward distributions. Thus CORe can impose proper exploration in different problems without parameter tuning. We prove a $\tilde{O}(d\sqrt{n \log K})$ gap-free bound on the n-round regret of CORe in a stochastic linear bandit. Our comprehensive empirical evaluation shows that CORe achieves impressive performance in various problems.

CORe is general enough to be applied to different structured problems, such as generalized linear bandits [Filippi et al., 2010] or neural bandits [Zhou et al., 2020]. The randomization strategy remains the same for different problems. We analyze the regret of CORe in a linear Gaussian bandit. Our analysis is under the assumption that the reward distributions of all arms have the same variance. An interesting future direction is a more general analysis of CORe.

Finally, we also believe that CORe can be further extended by other randomization designs, with the essential idea of capitalizing on the randomness in the agent's observed rewards and being fully data-dependent. For example, we can dynamically exchange rewards among arms with certain probability and keep the exchanged rewards in the arm's history along the n-round game. This can greatly improve the efficiency of sampling i.i.d. rewards from the reward pool in every single round. We have observed promising empirical performance of such algorithms and leave their more detailed study for future work.

REFERENCES

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2312–2320. Curran Associates, Inc., 2011.

Marc Abeille and Alessandro Lazaric. Linear thompson sampling revisited. *Electronic Journal of Statistics*, 11, 11 2016. doi: 10.1214/17-EJS1341SI.

Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs, 2012.

Shipra Agrawal and Navin Goyal. Further optimal regret bounds for thompson sampling. In Carlos M. Carvalho and Pradeep Ravikumar, editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 99–107, Scottsdale, Arizona, USA, 29 Apr–01 May 2013. PMLR.

Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. *Theor. Comput. Sci.*, 410 (19):1876–1902, April 2009. ISSN 0304-3975.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2–3):235–256, May 2002. ISSN 0885-6125.

Akram Baransi, Odalric-Ambrym Maillard, and Shie Mannor. Sub-sampling for multi-armed bandits. In Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 115–131, Berlin, Heidelberg, 2014a. Springer Berlin Heidelberg. ISBN 978-3-662-44848-9.

Akram Baransi, Odalric-Ambrym Maillard, and Shie Mannor. Sub-sampling for multi-armed bandits. In Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 115–131, Berlin, Heidelberg, 2014b. Springer Berlin Heidelberg.

Hock Peng Chan. The multi-armed bandit problem: An efficient non-parametric solution, 2019.

Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2249–2257. Curran Associates, Inc., 2011.

A. Chuklin, I. Markov, and M. Rijke. Click models for web search. In *Click Models for Web Search*, 2015.

Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Struct. Algorithms*, 22(1):60–65, January 2003. ISSN 1042-9832.

D. Eckles and M. Kaptein. Thompson sampling with the online bootstrap. *ArXiv*, abs/1410.4009, 2014.

Sarah Filippi, Olivier Cappe, Aurélien Garivier, and Csaba Szepesvári. Parametric bandits: The generalized linear case. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 586–594. Curran Associates, Inc., 2010.

Aditya Gopalan, Shie Mannor, and Yishay Mansour. Thompson sampling for complex bandit problems, 2013.

Baekjin Kim and Ambuj Tewari. On the optimality of perturbations in stochastic and adversarial multi-armed bandit problems. In *Advances in Neural Information Processing Systems 32*, pages 2695–2704. 2019.

Branislav Kveton, Csaba Szepesvari, Zheng Wen, and Azin Ashkan. Cascading bandits: Learning to rank in the cascade model, 2015.

Branislav Kveton, Csaba Szepesvari, Mohammad Ghavamzadeh, and Craig Boutilier. Perturbed-history exploration in stochastic multi-armed bandits, 2019a.

Branislav Kveton, Csaba Szepesvari, Sharan Vaswani, Zheng Wen, Tor Lattimore, and Mohammad Ghavamzadeh. Garbage in, reward out: Bootstrapping exploration in multi-armed bandits. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3601–3610. PMLR, 09–15 Jun 2019b.

Branislav Kveton, Csaba Szepesvári, Mohammad Ghavamzadeh, and Craig Boutilier. Perturbed-history exploration in stochastic linear bandits. In Ryan P. Adams and Vibhav Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 530–540. PMLR, 22–25 Jul 2020.

T.L Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.*, 6(1):4–22, March 1985. ISSN 0196-8858.

T. Lattimore and C. Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020. ISBN 9781108486828.

Tor Lattimore, Branislav Kveton, Shuai Li, and Csaba Szepesvári. Toprank: A practical algorithm for online stochastic ranking. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 3949–3958, Red Hook, NY, USA, 2018. Curran Associates Inc.

Lihong Li, Yu Lu, and Dengyong Zhou. Provably optimal algorithms for generalized linear contextual bandits. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2071–2080. JMLR.org, 2017.

Tie-Yan Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, March 2009. ISSN 1554-0669.

Xiuyuan Lu and Benjamin Van Roy. Ensemble sampling. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Peter McCullagh. Generalized linear models. *European Journal of Operational Research*, 16(3):285–292, 1984.

Ian Osband and Benjamin Van Roy. Bootstrapped thompson sampling and deep exploration, 2015.

Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 784–791, New York, NY, USA, 2008. Association for Computing Machinery.

Charles Riou and Junya Honda. Bandit algorithms based on thompson sampling for bounded reward distributions. In Aryeh Kontorovich and Gergely Neu, editors, *Proceedings of the 31st International Conference on Algorithmic Learning Theory*, Proceedings of Machine Learning Research, San Diego, California, USA, 2020. PMLR.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. In *International Conference on Learning Representations*, 2018.

Paat Rusmevichientong and John N. Tsitsiklis. Linearly parameterized bandits, 2008.

Liang Tang, Yexi Jiang, Lei Li, Chunqiu Zeng, and Tao Li. Personalized recommendation via parameter-free contextual bandits. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 323–332, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336215.

William R Thompson. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 25(3-4):285–294, 12 1933.

Sharan Vaswani, Branislav Kveton, Zheng Wen, Anup Rao, Mark Schmidt, and Yasin Abbasi-Yadkori. New insights into bootstrapping for bandits, 2018.

Sharan Vaswani, Abbas Mehrabian, Audrey Durand, and Branislav Kveton. Old dog learns new tricks: Randomized ucb for bandit problems. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1988–1998. PMLR, 26–28 Aug 2020.

Lijun Zhang, Tianbao Yang, Rong Jin, Yichi Xiao, and Zhi-Hua Zhou. Online stochastic linear optimization under one-bit feedback. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 392–401. JMLR.org, 2016.

Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with ucb-based exploration, 2020.

Masrour Zoghi, Tomas Tunys, Mohammad Ghavamzadeh, Branislav Kveton, Csaba Szepesvari, and Zheng Wen. Online learning to rank in stochastic click models. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 4199–4208. JMLR.org, 2017.