Bandit Algorithms for Factorial Experiments

Yutong Yan [1]

Advisors: Audrey Durand [2] Jolle Pineau [1]

[1]School of Computer Science, McGill University, Montreal, QC
[2]Computer Science and Software Engineering Department & Electrical and Computer Engineering Department, Université Laval, Québec City, QC

Table of Contents

List of Algorithms

List of Figures

Abstract

Factorial experiments [Yates, 1978] are experimental designs where one faces a sequence of decisions (factors) between different choices. These designs are commonly used because of their efficiency and the opportunity to detect interactions amongst intervention components. For example, factorial experiments are used for optimizing adaptive health interventions [Collins et al., 2007, Baker et al., 2017]. The goal is typically to identify the sequence of choices which optimizes some objective function. This can formulated as a multi-armed bandit problem [Robbins et al., 1952], where a player episodically select actions (here a sequence) and observes an reward. Given M factors and N choices per factor, this would result in N^M actions in the standard bandit setting. The number of actions therefore scales exponentially with the number of factors, learning algorithms being oblivious to the tree structure underlying the decision sequence. Under the assumption that the end reward is a linear combination of the selected choices rewards, it has been shown previously that the problem can be captured by linear bandits [Abbasi-Yadkori et al., 2011, Agrawal and Goyal, 2013]. In this formulation, an action correspond to a vector of length $M \times N$ encoding the choice selected for each factor. In this work, we aim to get rid of this assumption and tackle the problem using bandit algorithms for tree search [Kocsis and Szepesvári, 2006]. Under the tree search setting, the sequential decision tree is explicitly represented and one tries to optimize the path in the tree. For this task, we consider a popular bandit algorithm for tree search, that is Upper Confidence Bound applied to Trees (UCT) [Kocsis and Szepesvári, 2006]. UCT treats each decision node as a separate standard bandit problem, picking the choice which maximizes an up-

per confidence bound based on Hoeffding bounds, akin to the UCB1 [Auer et al., 2002] algorithm for standard bandits.

Using synthetic experiments, we first show that using tighter concentration bounds proposed in [Abbasi-Yadkori et al., 2011] can significantly improve the performance of UCT for tree search. Using further experiments, we investigate various factorial experimental design configurations: varying number of factors, number of choices per factors, amplitude of noise, and reward function. More specifically, we compare the performance of algorithms under three different formulations of the factorial experiment: 1) standard bandits; 2) linear bandits; and 3) bandits for tree search. We show that simple settings (typically few number of factors and choices per factors) lead to few actions under the standard bandits formulation and therefore can easily and efficiently be tackled using standard bandit algorithms (for example, Thompson Sampling [Thompson, 1933, Chapelle and Li, 2011]). However, as the number of actions increases (due to more factors and/or more choices per factors), we observe that capturing the underlying tree structure is essential for robustness, whether the reward function is linear or not. Finally, we observe that the algorithms working under the bandits for tree search formulation of factorial experimental designs also seem more robust to the noise variance, compared with approaches learning under other formulations.

Abbreviation

BATS Bandit Algorithm for Tree Search

MAB Multi-Armed Bandit

MVB Multivariate Bandit

UCB Upper Confidence Bound

UCT Upper Confidence Bounds applied to Trees

RL Reinforcement Learning

RCT Randomized Controlled Trials

OFU Optimism in the Face of Uncertainty

OFUL Optimism in the Face of Uncertainty Linear Bandit

TS Thompson Sampling

MDP Markov Decision Process

MCTS Monte Carlo Tree Search

LinTS Linear Thompson Sampling

Notation

$\mathbb{1}\{\text{cond}\}$ Indicator function to be 1 if the condition is satisfied, otherwise 0

ϵ Noise generated by the environment

C_p Exploration constant

$\mathcal{D}_k$ Reward distribution of arm k

$R(S)$ Reward function of a super arm

N Total number of times of all arms played

N_k Total number of times of the arm k being played

$\mathcal{N}$ Gaussian distribution

T Time horizon

$\mathcal{K}$ Set of arms

K Total number of arms

k A single arm

k_t An arm selected at time step t

$Y_{k,t}$ Outcome (Reward) sampled from reward distribution $\mathcal{D}_k$ for arm k at time step t

μ_k Expected mean of $\mathcal{D}_k$ for the arm k

$\mu_\star$ Optimal expected outcome (reward) generated from the optimal arm $k_\star$

$\hat{\mu}_k$ Estimated mean of the arm k

$\mathfrak{R}_T$ Cumulative regret till horizon T

S A super arm composed of m individual arms

$\mathcal{S}$ Super arm space of all super arms

m Size of the super arm S, also denoted as $|S|$

x A single arm

$\mathcal{X}$ Set of arms

D_t Features of arms

X_t The chosen arm in linear bandits

Y_t Outcome (Reward) in linear bandits

Chapter 1

Background

1.1 Factorial Experiments

The initiative to study MVB comes from the factorial design problem. A *factorial design* is a test whose structure comprises of at least two factors, each with discrete conceivable values, called choices, and whose exploratory units go up against every single possible combination of these dimensions over every single such factor [Montgomery, 1995]. These designs are commonly used because of their efficiency and the opportunity to detect interactions amongst intervention components. For example, factorial experiments are used for optimizing adaptive health intervention [Collins et al., 2007, Baker et al., 2017]. The goal is typically to identify the sequence of choices which optimizes some objective function.

It is beneficial when researchers need to investigate the effects on a single dependent variable in an environment consisting of two or more independent variables. However, a full factorial experiment requires a large size of samples in real applications. Although fractional factorial designs alias high-order interaction effects of using sample sizes more efficiently, these approaches all seek to select the set of factors to explain given data, rather than maximizing the reward received from the set of factors. Choosing a set of interac-

tions to allow into the bandit is analogous to choosing a particular fractional factorial design in a classical problem.

To illustrate the process of factorial design experiments, we present a simple example: a medical app aims to maximize the participation of patients to keep track of their recovery. Studies have shown that the increasing use of healthcare apps, i.e. health education apps for patients, will improve users' health condition [Loiselle and Ahmed, 2017]. Suppose there are only two *factors* considered, message delivery method and delivery frequency. For the *factor* message delivery method, there exist two *choices*, email and SMS. For the *factor* delivery frequency, there are also two *choices*, for email, every day or every week, and for SMS, every hour or every day. Statisticians usually forms this as two choices, less frequent, or more frequent [1]. Therefore, this example has four *treatment combinations*, called a *2 factors* $\times$ *2 choices per factor* factorial design. A full factorial design considers all possible treatment combinations, which turns out to be unfeasible as the number of factors and the number of levels in each factor increase. In this case, a fractional factorial design may be done, in which some of the treatment combinations may be omitted.

1.2 The Multi-Armed Bandit Problem

Bandit problem is also described as an online decision making problem with discrete time steps. A decision maker sequentially chooses arms from the set of K possible arms $\mathcal{K} = \{1, 2, \ldots, K\}$, and an environment that assigns rewards to the arms and provides the agent with different types of feedback, which we will define later in this chapter.

Algorithm 1 shows the general framework for the MAB game. At each time step $t \in \mathbb{N}$, the environment chooses a vector of K rewards $Y_t \in [0, 1]^K$, whose components are denoted by $Y_{k,t}$ for each arm $k \in \mathcal{K}$. The reward $Y_{k,t}$ of each arm is assumed to be in $[0, 1]$. At the same time, the decision maker chooses the arm k_t which will be played in that time

[1] We will discuss the discrepancy in experiment design between our approach and statistical approaches, and explain the advantages of bandit algorithms for factorial experiments.

step t, based on the feedback from its previous interactions with the environment. The decision maker then observe the reward given by the environment based on the choice of the type of feedback in this game. After obtaining the feedback, the decision maker updates the estimated reward for each arm $k \in \mathcal{K}$. The goal of the decision maker is to maximize the cumulative reward of its choices of selected arms after T time steps, i.e., $\sum_{t=1}^{T} Y_{k_t,t}$. The number of time steps T is called the time horizon.

Algorithm 1: General MAB Framework

Input: Time horizon T, K number of arms with unknown parameters of reward distribution

for *each time step* $t := 1, 2, 3, ..., T$ **do**
    Choose an arm $k_t \in \mathcal{K}$, and sends it to the environment.
    Observe the reward depending on the feedback chosen.
    Update the estimated rewards $\forall k \in \mathcal{K}$.
end

1.2.1 Stochastic Multi-Armed Bandits

In stochastic MAB, the reward of each arm $k \in \mathcal{K}$ at each time step t is drawn from a fixed distribution $\mathcal{D}_k$ on $[0, 1]$, independently from all other rewards of that action and from the rewards of other actions [Robbins et al., 1952]. For every arm $k \in \mathcal{K}$, the distribution $\mathcal{D}_k$, with expected mean we denote by μ_k, is not known to the decision maker. In this work, we will mainly focus on stochastic MAB. For Gaussian Bandits, the reward generated from the environment at t-th time step for arm k is

$$Y_{k,t} \sim \mathcal{N}(\mu_k, \sigma^2), \tag{1.1}$$

where μ_k is the expected mean of the Gaussian distribution $\mathcal{N}_k$ of the reward of the arm k, and σ^2 is the variance of $\mathcal{N}_k$. Both are unknown to the decision maker during the game. It can also be understood as the reward is the sum of the expected mean of the distribution

plus some noise generated by the environment

$$Y_{k,t} = \mu_k + \epsilon_t, \tag{1.2}$$

where ϵ is the noise such that

$$\epsilon_t \sim \mathcal{N}(0, \sigma^2) \tag{1.3}$$

It is clear that the two presentations of the reward $Y_{k,t}$ are equivalent. There are also other variants other than Gaussian bandits, such as Bernoulli bandits. However, we will focus on Gaussian bandits in the following sections.

1.3 Types of Information Feedback

In bandits literature, the feedback that the decision maker could receive from the environment can be characterized into three broad categories, full information feedback, partial information (semi-bandit) feedback and bandit (full bandit) feedback.

1.3.1 Full Information Feedback

In *full information feedback*, the environment reveals the rewards of all the arms $k \in \mathcal{K}$. Therefore, the decision maker can fully observe the reward vector chosen by the environment $Y_t \in [0, 1]^K$ [Takimoto and Warmuth, 2003, Kalai and Vempala, 2005, Cesa-Bianchi and Lugosi, 2006].

1.3.2 Partial Information Feedback

In *partial information feedback*, also called *semi-bandit feedback* in combinatorial bandit setting, the environment only reveals the rewards of the selected arm $k_t \in \mathcal{K}$. Hence, the decision maker can only observe the reward of the selected arm $Y_{k_t,t}$ from the arm k_t. In

combinatorial setting, the decision maker observes the outcomes of the selected m arms $\{Y_{k_t,t}\}_{k_t \in S_t}$ from the arms $\{k_t\}_{k_t \in S_t}$, as well as the reward of the super arm S_t, where a super arm S_t is composed of m individual arms [Neu and Bartók, 2013, Sankararaman and Slivkins, 2017]. We will define the reward function, taking a super arm as input in the next section.

1.3.3 Bandit Feedback

Bandit feedback, also called *full bandit feedback*, mostly occurs in combinatorial bandit setting, where the decision maker is allowed to select more than one arm. When a decision maker selects m arms to construct a super arm S_t, the environment only reveals a reward for the super arm S_t to the decision maker. Unlike in other settings, the decision maker is not able to observe the reward for a single arm but a composite of the m arms selected. One special case is that when $m = 1$, bandit feedback is equivalent to partial information feedback. We define the expectation of the reward of the super arm S_t as the outcome from an unknown reward function $R(\cdot)$ at time step t

$$\mathbb{E}[r_t] = R(S_t) \tag{1.4}$$

Notice that we use different notation for the reward of the super arm as r_t to distinguish from the $Y_{k_t,t}$, since $Y_{k_t,t}$ is defined as the reward for a single arm in non-combinatorial bandit setting, but the feedback of individual arms in combinatorial bandit setting with full information feedback or semi-bandit feedback.

The reason why we differentiate the notation in this way is as follows: in combinatorial bandit setting with *semi-bandit feedback*, the decision maker is able to observe the outcomes of any individual arms included in the selected super arm and the reward of the super arm. The reward of the super arm does not affect the decision maker's decision phase while choosing a super arm, rather than in the evaluation metrics when we evaluate the algorithm. On the other hand, the outcomes of any individual arms included in the

selected super arm are updated every time the decision maker observes them, and the decision maker makes decision of next super arm to choose based on that. We will introduce combinatorial bandit later in this chapter. There are not many works that have studied this setting due to its complexity and many existing algorithms designed for semi-bandit feedback might suffer sub-optimality in this setting.

Most works have assumed a linear reward function when solving problems with bandit feedback, however, this assumption might be too strong to fail. We will show the comparison in the experiments section.

1.4 Performance Measures

As noted before, the goal of the decision maker is to maximize its accumulated reward. In this section we describe the way we evaluate the performance of the agent in achieving this goal. Generally, we do not evaluate the performance of the agent by the magnitude of the reward it accumulates as the game goes on, since we are not able to give theoretical guarantees and the magnitude of the reward also affects in the plots for visualization. On the other hand, to maximize the reward is equivalent to minimize the regret, therefore, we could measure the performance by the cumulative regret.

Regret is the difference between the maximum cumulative expected reward the decision maker could obtain and the cumulative expected reward the decision maker actually obtains up to T time steps, which is defined as

$$\mathfrak{R}_T = \sum_{t=1}^{T} \max_{k \in \mathcal{K}} \mu_k - \mu_{k_t} \tag{1.5}$$

Note that the optimal arm $k_\star$ that results in the maximum reward, for any time step t, could vary at different time steps.

In the stochastic setting, since rewards are random, in the long run the best arm be-

comes the arm with largest expected reward

$$k_\star = \operatorname*{argmax}_{k \in \mathcal{K}} \mu_k \tag{1.6}$$

Accordingly, in the stochastic setting, as common in the literature we redefine the regret to be

$$\mathfrak{R}_T = \sum_{t=1}^{T} \mu_{k_\star} - \mu_{k_t} \tag{1.7}$$

1.5 Common Principles

1.5.1 Optimism in Face of Uncertainty

A natural and successful way to design a bandit algorithm is the Optimism in the Face of Uncertainty (OFU) principle. The basic idea is to maintain a confident interval for every arm, or a confidence set of parameters for every arm in multi-dimension case. The confidence interval of every arm is separately determined based on past observations. At each time step, the decision maker picks the arm with the highest value of estimated mean plus the exploration bonus, where the exploration bonus is the confidence interval of the arm. By making decisions optimistically, the decision maker is able to balance off exploitation and exploration. The intuition is that the more an arm is played, the confidence interval of the arm shrinks with more observations, therefore, the decision maker can identify the arm as the sub-optimal arm or not. Exploitation is how the decision maker exploits the observations and estimated means of every arm to make decisions. Exploration is how the decision maker explores by choosing the arms that might potentially be the optimal arm but do not have the highest estimated means at the current time step. The OFU principle elegantly solves the exploration-exploitation dilemma inherent in the problem.

One famous algorithm, representative of OFU, is Upper Confidence Bound (UCB)

Algorithm 2: UCB1 [2] [Auer et al., 2002]

Input: Time horizon T, K number of arms with unknown parameters of reward distribution

for *each time step* $t := 1, 2, 3, ..., T$ **do**
  if $t \leq K$ **then**
    Play an arm k_t from $\mathcal{K}$ with index $t \in \{1, 2, \ldots, K\}$
  else
    Compute $\tilde{\mu}_k \leftarrow \hat{\mu}_k + 2C_p\sqrt{\frac{\ln N}{N_k}}$ for $k \in \mathcal{K}$;
    Choose the arm $k_t = \text{argmax}_{k \in \mathcal{K}} \tilde{\mu}_k$
  end
  The decision maker observes the reward $Y_{k,t}$ of the arm k_t;
  Update the arm k_t with $\hat{\mu}_k \leftarrow (\hat{\mu}_k N_k + Y_{k,t})/(N_k + 1)$ and $N_k \leftarrow N_k + 1$
end

[Auer et al., 2002]. Note that in Algorithm 2, the decision maker chooses an arm

$$k_t = \underset{k \in \mathcal{K}}{\text{argmax}}\, \hat{\mu}_k + 2C_p\sqrt{\frac{\ln N}{N_k}}, \tag{1.8}$$

where C_p is the fixed exploration constant chosen by the algorithm, N is the total number of times of all arms played till this time step, N_k is the total number of times of arm k being played till t-th time step, and $\hat{\mu}_k$ is the estimated mean of arm k

$$\hat{\mu}_k = \frac{\sum_t \mathbb{1}\{k_t = k\} Y_{k,t}}{N_k}, \tag{1.9}$$

where $\mathbb{1}$ is the indicator function to represent whether arm k is selected at time step t or not.

1.5.2 Thompson Sampling

Thompson Sampling (TS), also known as posterior sampling and probability matching, was first introduced for allocating experimental effort in two-armed bandit problems arising in clinical trials in [Thompson, 1933]. The basic idea is as follows: assuming a

[2] Partial information feedback is given by the environment. We will assume partial information feedback in the following when the decision maker is allowed to choose one arm at any time step.

simple prior distribution on the expected means of the reward distributions of every arm. At every time step, the decision maker chooses an arm based on its posterior probability of being the best arm. The intuition is that the more an arm being played, the less uncertainty the play holds for that arm, therefore, if the arm appears to be the sub-optimal arm, it would not be played later, vice versa. Algorithm 3 has shown TS for Gaussian bandits. The difference between UCB and TS is the belief of an arm when the decision maker makes the decision of choosing the arm.

Algorithm 3: TS for Gaussian Bandits [Durand and Gagné, 2017]

Input : Time horizon T, K number of arms with unknown parameters of reward distribution

Initialization: Initialize all arms $k \in \mathcal{K}$ with $N_k = 0$, $\hat{\mu}_k = 0$

for *each time step* $t := 1, 2, 3, ..., T$ **do**
  if $t \leq K$ **then**
    Play an arm k with index $t \in \{1, 2, \ldots, K\}$.
  else
    Sample all arms $\tilde{\mu}_k \sim \mathcal{N}(\hat{\mu}_k, \frac{1}{N_k+1})$ for $k \in \mathcal{K}$
    Choose the arm $k_t = \text{argmax}_{k \in \mathcal{K}} \tilde{\mu}_k$
  end
  The decision maker observes the reward $Y_{k,t}$ of the arm k_t
  Update the arm k_t with $\hat{\mu}_k \leftarrow (\hat{\mu}_k N_k + Y_{k,t})/(N_k + 1)$ and $N_k \leftarrow N_k + 1$
end

1.6 Linear Bandits

In the linear bandit problems, likewise in context-free MAB, a decision maker chooses an arm at t-th time step, and receives a stochastic reward. However, the expected value of this stochastic reward is an unknown linear function of the arm choice. Same as in all bandit problems, the decision maker seeks to maximize the cumulative reward over T time steps. The stochastic context-free MAB can be seen as a special case of the linear bandit problem: the set of available arms at t-th time step is the standard basis e_i for the Euclidean space $\mathbb{R}^d$, i.e. the vector e_i is the vector with all zeros except for a one in the i-th coordinate. In this scenario, all arms are independent of each other, and the reward

depends only on a single parameter as in the context-free case [Auer et al., 2002, Dani et al., 2008, Rusmevichientong and Tsitsiklis, 2010]. Likewise in context-free MAB, there are also algorithms that utilize OFU and TS strategies for linear bandits [Abbasi-Yadkori et al., 2011, Agrawal and Goyal, 2013, Abeille et al., 2017].

Algorithm 4: General Linear Bandit Framework

Input: Time horizon T, a set of arms $x \in \mathcal{X}$

for *each time step* $t := 1, 2, 3, ..., T$ **do**
- Observe the arm space $\mathcal{X}_t$.
- Choose the rewards $\forall x \in \mathcal{X}_t$.
- Choose an arm X_t, and sends it to the environment.
- Observe the reward Y_t.
- Update the estimated parameter $\tilde{\theta}_t$.

end

1.6.1 Linear Bandits with OFU

Algorithm 4 shows the general framework of linear bandits. Note that in linear bandit case, the notation for arms are different than in the context-free case. At t-th time step, a decision maker first observes the arm space $\mathcal{X}_t \subseteq \mathbb{R}^d$ as well as every arm $x \in \mathcal{X}_t$. Though discrepancies exist in different works, the arm space could stay unchanged or change at any t-th time step. If the arm space stays unchanged, $\mathcal{X}_t = \mathcal{X}_{t'}\ \forall t, t' = 1, 2, 3, ..., T$. Same as in the context-free MAB, the environment then chooses the rewards for every arm $x \in \mathcal{X}_t$ by

$$Y_{x,t} = \langle x, \theta_\star \rangle + \epsilon_t, \tag{1.10}$$

where $\theta_\star \in \mathbb{R}^d$ is an unknown parameter to the decision maker, and ϵ_t is a noise parameter, defined in previous sections. The decision maker then chooses an arm X_t, and receives a

reward Y_t, for simplicity, defined as

$$Y_t = \langle X_t, \theta_\star \rangle + \epsilon_t, \tag{1.11}$$

where ϵ_t satisfies the tail constraints, with $\mathbb{E}[\epsilon_t | X_{1:t}, \epsilon_{1:t-1}] = 0$. Note that the capitalized notation means they represent numerical values, after the decision maker makes the decision at every time step. The strategy to maximize the expected cumulative rewards over T time steps given by $\sum_{t=1}^{T} \langle X_t, \theta_\star \rangle$ is to an estimate $\tilde{\theta}_t$ of the unknown parameter with observations from the environment. To use the existing information, the decision maker can first construct a least-squares estimate of $\theta_\star$ given as follows

$$\hat{\theta}_t = (\boldsymbol{X}_{1:t}^T \boldsymbol{X}_{1:t} + \lambda I)^{-1} \boldsymbol{X}_{1:t}^T \boldsymbol{Y}_{1:t}, \tag{1.12}$$

where $\boldsymbol{X}_{1:t}$ is the matrix whose rows are $X_1^T, X_2^T, X_3^T, ..., X_t^T$, $\boldsymbol{Y}_{1:t} = (Y_1, ..., Y_t)^T$, and λ is the regularization parameter. As we mentioned before, there are also algorithms developed using OFU and TS for linear bandits. Optimism in the Face of Uncertainty Linear Bandit (OFUL) is the *state-of-the-art* algorithm with theoretical guarantees for linear bandits [Abbasi-Yadkori et al., 2011]. Let $V = \lambda I_d$, define

$$\overline{V_t} = V + \sum_{s=1}^{t} X_s X_s^T \tag{1.13}$$

$$||x||_{\overline{V}_t} = \sqrt{x^T \, \overline{V_t} \, x} \tag{1.14}$$

Assume $||\theta_\star||_2 \leq S$. Then, for any $\delta > 0$, with probability $1 - \delta$,

$$\beta_t = R\sqrt{2\log(\frac{\det(\overline{V_t})^{1/2} \det(\lambda I)^{-1/2}}{\delta})} + \lambda^{1/2} S, \tag{1.15}$$

where R is the standard deviation for the R-sub-Gaussian noise ϵ_t. The arm then selection according to the estimated $\tilde{\theta}_t$ follows

$$X_t = \underset{x \in \mathcal{X}_t}{\operatorname{argmax}} \langle x, \hat{\theta}_t \rangle + \beta_t \cdot ||x||_{\bar{V}_t}, \tag{1.16}$$

which is equivalent to

$$X_t = \underset{x \in \mathcal{X}_t}{\operatorname{argmax}} \langle x, \tilde{\theta}_t \rangle \tag{1.17}$$

1.6.2 Linear Bandits with Thompson Sampling

Linear Thompson Sampling (LinTS), however, employs another strategy [Agrawal and Goyal, 2013, Abeille et al., 2017]. As in context-free case, we will illustrate the inference process for Linear Gaussian Bandits. The estimate of unknown parameter $\tilde{\theta}_t$ is selected through TS, in linear bandits, Multivariate Normal distribution ($\mathcal{MVN}$) is used instead of Normal distribution

$$\tilde{\theta}_t \sim \mathcal{MVN}(\hat{\theta}, v^2 \overline{V_t}^{-1}), \tag{1.18}$$

where $\hat{\theta}$ follows as Equation 1.12, $\overline{V_t}$ follows as Equation 1.13, and v is a scalar parameter of the covariance matrix. Although in order to obtain theoretical guarantees, v is suggested to be $R\sqrt{9d \ln \frac{T}{\delta}}$ with confidence $1 - \delta$, or $R\sqrt{9d \ln \frac{t}{\delta}}$ if T is not known [Agrawal and Goyal, 2013]. Note that R is defined same as Equation 1.15. However, this suggestion of v for bounding the regret is indeed loose compared to OFUL, and implementations of real applications often choose $v = 1$ to outperform OFUL, though without theoretical guarantees. The arm selection then follows Equation 1.17.

In the stochastic setting, since rewards are stochastic with noise, the best arm at every

t-th time step can vary, therefore, the best arm at t-th time step is denoted as

$$x_t^\star = \underset{x \in \mathcal{X}_t}{\operatorname{argmax}} \langle x, \theta_\star \rangle \tag{1.19}$$

Accordingly, in the stochastic setting, as common in the literature we redefine the regret to be

$$\mathfrak{R}_T = \sum_{t=1}^{T} \langle x_t^\star, \theta_\star \rangle - \langle X_t, \theta_\star \rangle = \sum_{t=1}^{T} \langle x_t^\star - X_t, \theta_\star \rangle \tag{1.20}$$

1.7 Combinatorial Bandits

In combinatorial MAB problems, the decision maker can choose m arms, where $m \geq 1$, and receive feedback according to different types of feedback chosen[3], as well as the reward of the super arm composed of the m individual arms. The reward function is unknown to the decision maker. At each time step $t \in \mathbb{N}$, the decision maker chooses the super arm S_t, composed of m individual arms, denoted as $\{k_t\}_{k_t \in S_t}$, which will be played in the time step t, based on the feedback from its previous interactions with the environment. The decision maker then observes the outcomes of depending on the feedback type chosen, as well as the reward of the super arm $R(S_t)$ through the reward function $R(\cdot)$. After obtaining the information, the decision maker updates the estimated outcome for each arm $k \in \mathcal{K}$. The goal of the decision maker is to maximize the cumulative reward of its choices of selected super arms after T time steps, i.e., $\sum_{t=1}^{T} R(S_t)$ [Neu and Bartók, 2013, Sankararaman and Slivkins, 2017, Combes et al., 2015, Cesa-Bianchi and Lugosi, 2012]. Note that if $m = 1$, MAB is a special case of combinatorial MAB.

At every t-th time step, a decision maker selects a feature vector of the super arm S_t, M_t from a finite set $\mathcal{M} \subset \{0, 1\}^K$, where K is the size of the arm set, equivalent to $|\mathcal{K}|$. An arm $k \in \mathcal{K}$ chosen in the super arm S_t makes the component $M_i = 1$, where i is the

[3] As we introduced in previous sections, there are full information feedback, semi-bandit feedback, and bandit feedback.

Algorithm 5: General Combinatorial MAB Framework

Input: Time horizon T, K number of arms with unknown parameters of reward distribution

for *each time step* $t := 1, 2, 3, ..., T$ **do**
  Choose a super arm S_t, and sends it to the environment.
  Observe the reward as well as feedback of individual arms depending on the feedback chosen.
  Update the estimated expectation vector $\hat{\boldsymbol{\mu}}$.
end

index of the arm in the arm set. Let $\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3, ..., \mu_K)$ be a vector of expectations of all arms $k \in \mathcal{K}$. The expectation vector $\boldsymbol{\mu}$ is also unknown to the decision maker. We assume all super arms $S \in \mathcal{S}$ consist of the same number m of individual arms so that $||M||_1 = m \ \ \forall M \in \mathcal{M}$. The reward function need not be linear in this case. One example of linear reward function taking input as a super arm S from the super arm set $\mathcal{S}$, as well as constructed feature vector $\boldsymbol{M}$, could be

$$\mathbb{E}[R(S)] = \sum_{i=1}^{K} \boldsymbol{M_i \mu}_i \tag{1.21}$$

One example of non-linear reward function, known as *max* function, is as follows

$$\mathbb{E}[R(S)] = \max_{i \in [K]} \boldsymbol{M_i \mu}_i \tag{1.22}$$

Algorithm 5 shows the general combinatorial MAB framework. One thing to note is that in full information bandit setting, the outcomes of all individual arms would be shown to the decision maker, in the stochastic case, with noise. In the semi-bandit setting, the outcomes of arms with expectation μ_i, in the stochastic case, with noise, only would be shown to the decision maker when $M_i = 1$. In the bandit setting, there would be none of the outcomes of individual arms shown to the decision maker. It is worth to note that there are numerous works focusing on semi-bandit setting, mainly to aim to obtain accurate estimated expectation vector $\hat{\boldsymbol{\mu}}$ in order to maximize the reward of the selected super arm, based on the outcomes observed from individual arms in the selected super

arm at every t-th time step, instead of the reward received by selecting the super arm.

In the stochastic setting, since rewards are stochastic with noise, in the long run the best arm becomes the arm with largest expected reward

$$S_\star = \underset{S \in \mathcal{S}}{\operatorname{argmax}}\, R(S) \tag{1.23}$$

Accordingly, in the stochastic setting, as common in the literature we redefine the regret to be

$$\mathfrak{R}_T = \sum_{t=1}^{T} R(S_\star) - R(S_t) \tag{1.24}$$

Chapter 2

Multivariate Bandits

2.1 Upper Confidence Bounds applied to Trees

UCB1 has been one of the most popular methods in bandit family for decades. Upper Confidence Bounds applied to Trees (UCT) algorithm adapts UCB1 in tree structure as treating the choice of child node as a multi-armed bandit problem, the value of a child node is the expected reward approximated by the Monte Carlo simulations, and hence these rewards correspond to random variables with unknown distributions [Kocsis and Szepesvári, 2006, Kocsis et al., 2006, Kaufmann and Koolen, 2017]. Therefore, it is a promising candidate to address the exploration-exploitation dilemma in Monte Carlo Tree Search (MCTS): every time a node (action) is to be selected within the existing tree, the choice can be modelled as a MAB problem.

To illustrate a UCT algorithm, consider a tree search optimization problem on a uniform tree of depth D where each node has K children[1]. A reward distribution $\mathcal{D}_l$ is assigned to each leaf node l[2]. The goal is to find the path, sequence of nodes starting from the node to be selected at initial state s_0 to a leaf node at terminal state s_T, with highest

[1]Generally, different nodes are allowed to have different numbers of children nodes in a search tree. It is only for illustration here.

[2]In this case, there are K^D such leafs.

Algorithm 6: Bandit Algorithms for Tree Search [Coquelin and Munos, 2007]

Input: Time horizon T, K^D number of leaves with unknown parameters of reward distribution

for *each time step* $t := 1, 2, 3, ..., T$ **do**
  Set v_d to v_0.
  Start running trajectory.
  for *depth* $d := 1, 2, ..., D$ **do**
    Compute the B-value for every children node $v' \in \mathcal{C}(v_d)$.
    Select the node v' with highest B-value.
    Set selected children node v' to v_d.
  end
  Reach the leaf $l_t \leftarrow v_D$.
  End trajectory.
  Receive reward $r_t \sim \mathcal{D}_{l_t}$.
  Update the estimates of the nodes visited in the trajectory.
end

mean value μ_l, where the mean value of every leaf node is defined as

$$\mu_l = \mathbb{E}[\mathcal{D}_l], \tag{2.1}$$

and the optimal can result in the highest mean value across all leaf nodes, with the so-called optimal leaf node,

$$\mu_{l_\star} = \mathbb{E}[\mathcal{D}_{l_\star}] \tag{2.2}$$

Define the value of any node v, excluding the leaf nodes, as

$$\mu_v = \max_{v' \in \mathcal{L}(v)} \mu_{v'}, \tag{2.3}$$

where $\mathcal{L}(v)$ denotes the set of leaves that belong to the branch originating from node v. At every $t+1$-th time step, the decision maker selects a leaf l_t from the tree and receives a reward $r_t \sim \mathcal{D}_{l_t}$[3], which enables backpropagation of the mean value μ_v and the visit count

[3]It can also be written as $r_t = \mu_{l_t} + \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ if $\mathcal{D}_{l_t}$ is a Gaussian distribution with mean μ_{l_t} and variance σ^2 as $\mathcal{N}(\mu_{l_t}, \sigma^2)$.

N_v for the node v for every node v such that $l_t \in \mathcal{L}(v)$ [4], where $\mathcal{L}(v)$ denotes all nodes that can be reached by node v through a trajectory starting from the node v. Therefore, the cumulative regret over T time steps is defined as

$$\mathfrak{R}_T = \sum_{t=1}^{T} \mu_{l_\star} - \mu_{l_t} \tag{2.4}$$

The way the leaf is selected is by following a path starting from the root node at initial state and such that from each node v along the path, the next selected node v' is the one with highest B-value among children nodes, where the B-value is not fixed, nor guaranteed to be the best across all domains. The first work proposed UCT has suggested B-value, using UCB1, to be defined as [Kocsis and Szepesvári, 2006]

$$B_{v'} = \hat{\mu}_{v'} + 2C\sqrt{\frac{\ln N_v}{N_{v'}}}, \tag{2.5}$$

where $C = 1/\sqrt{2}$ was shown to satisfy the Hoeffeding inequality with rewards in the range $[0, 1]$ [Kocsis et al., 2006], $N_{v'}$ is the number of paths that went through node v' till time step t, defined as[5]

$$N_{v'} = \sum_{t} \mathbb{1}\{l_t \in \mathcal{L}(v')\} \tag{2.6}$$

$\hat{\mu}_{v'}$ is the empirical average of rewards obtained from leaves originating from node v', i.e.,

$$\hat{\mu}_{v'} = \frac{\sum_t \mathbb{1}\{l_t \in \mathcal{L}(v')\} r_t}{N_{v'}} \tag{2.7}$$

[4] Details are illustrated in the Algorithm 6

[5] N_v can also be defined in a similar way.

Other choices of B-values could also be the UCB(δ) proposed in [Abbasi-Yadkori et al., 2011]. Assume the noise ϵ_t is conditionally 1-sub-Gaussian, with probability $1-\delta$,

$$B_{v'} = \hat{\mu}_{v'} + \sqrt{\frac{(1+\frac{1}{N_{v'}})\ln(K\sqrt{N_{v'}+1})/\delta}{2N_{v'}}}, \tag{2.8}$$

where K is the number of children nodes of the parent node v.

As shown in Algorithm 6, a trajectory is a sequence of nodes from the root to a leaf, where at each node v, the next node is chosen as the (or one) child having the highest B value among its children $\mathcal{C}(v)$. A reward is received at the leaf. After a trajectory is run, the decision maker then updates the B-value and number of times being visited for the nodes in the trajectory.

The intuition for the UCT algorithm is that at the state of a given node v, there are K possible arms corresponding to the children nodes, and the use of a UCB-type of bandit algorithm should balance exploitation-exploration trade-offs and select the best arm given noisy rewards samples.

2.2 Multivariate Optimization

Multivariate optimization means optimization of a scalar function of a several variables [Boyd and Vandenberghe, 2004]:

$$y = P(x) \tag{2.9}$$

and has the general form:

$$\min_{x} P(x) \tag{2.10}$$

where $P(x)$ is a nonlinear scalar-valued function of the vector variable $x = (x_1, x_2, x_3, ..., x_n)$.

2.3 Motivates to Study Multivariate Bandits

Factorial design (Section 1.1) has caught our attention since there are many necessities in real world problems as existing algorithms do not provide a satisfactory solution to those problems. Previous works have shown contextual MAB has outperformed traditional Randomized Controlled Trials (RCT) and non-contextual MAB when the target population is divided into subgroups [Shaikh, 2019]. In addition, although massive works have done in full information feedback and semi-bandit (partial information) feedback, there exists few works about bandit feedback in bandits literature (Section 1.3). It is true that in full information feedback and semi-bandit feedback, theoretical guarantees for existing algorithms are easier to achieve, and in bandit feedback, most algorithms proposed in other two types of feedback are not able to achieve logarithmic performance. Bandit algorithms have been popular for online learning for decades. We will show in the following sections that to solve MVB problems, we can use online methods such as bandit algorithms constructed in specific structure.

2.4 Multivariate Bandit Problems

As shown in Algorithm 7, given a vector variable x, at every t-th time step, the decision maker selects the arm $k_{x_i} \in \mathcal{K}_{x_i}$ for every variable $x_i \in x$, where $\mathcal{K}_{x_i}$ is the set of choices the decision maker is allowed to choose from for variable x_i. The decision maker then stores the choices in the decision set D_t, and sends D_t to the environment. The decision maker is assumed to be hidden from the reward function $R(\cdot)$, henceforth, the decision maker has to infer the estimates from the reward r_t received. One thing to note that here we assume the numerical values of all choices for every $x_i \in x$ are discrete. If the values are continuous, we can split the continuous values in certain intervals, and treat every interval as a discrete numerical value. Like we stated in Section 1.7, the reward function $R(\cdot)$ does not have to be linear, nor the decision maker tries to reconstruct the reward function, but the decision maker rather seeks to maximize the reward, or equivalently,

Algorithm 7: General Multivariate Bandit Framework

Input: Time horizon T, vector variable x

for *each time step* $t := 1, 2, 3, ..., T$ **do**
  $D_t \leftarrow \emptyset$
  for *every* $x_i \in x$ **do**
    Make choice for x_i by choosing k_{x_i} from the set $\mathcal{K}_{x_i}$.
    Store the choice in the set D_t by $D_t \cup \{k_{x_i}\}$.
  end
  Send D_t to the environment.
  Receive the reward r_t.
  Update the estimates.
end

minimize the regret, with the collected observations from previous time steps by selecting the optimal decision set $D_\star$. The cumulative regret obtained till time step t is defined as

$$\mathfrak{R}_T = \sum_{t=1}^{T} R(D_\star) - R(D_t) \tag{2.11}$$

Note that to solve the selection problem for decision set D, we need an efficient algorithms, i.e. Linear TS and UCT. Algorithm 7 only showed a new framework of bandit, MVB, where a substantial number of real applications can fit in the framework naturally. For example, the factorial experiments we shortly introduced in Section 1.1 can be solved within the General MVB Framework with an efficient algorithm.

2.5 Connection to Combinatorial MAB

What distinguishes MVB from combinatorial MAB is that combinatorial MAB optimize subset selection while multivariate bandits optimize by considering variable selection [Hill et al., 2017], i.e., combinatorial MAB considers the effect (beneficial or not) to include this individual arm in the super arm versus exclude among all available arms, however, MVB gauges that which arm of a specific variable can lead to a greater reward. Nevertheless, the difference behind the intuition of the two frameworks does not affect

the fact that we could still use combinatorial MAB algorithms to solve MVB problems, though inefficient.

Combinatorial MAB with bandit feedback can be treated as a linear bandit problem. Like in Section 1.7, we can treat every possible super arms[6], along with their feature vector, as an arm in linear bandit problems (Section 1.6). Therefore, we could solve the problem by constructing an estimated underlying parameter. There are two issues here: one is that we have to consider all possible choices, which results in computationally expensive problems in runtime, especially in real applications. Although we could apply kernel methods to reduce the dimension of the feature vector [Valko et al., 2013, Jun et al., 2017,Filippi et al., 2010], due to the fact that the reward function is hidden and the reward function might not be linear, the number of choices needed to be considered would not significantly decrease as the decision maker has to extensively sweep over the arm space to avoid missing the optimal arm.

2.6 MVB Problems in Tree Structure

As shown in Algorithm 7, we can employ the tree search setting so that the sequential decision tree is explicitly represented and the goal would be to optimize the path in the tree. Figure 2.1 has shown an example of how MVB can be formulated in tree structure, corresponded with Algorithm 7. The decision process is analogous to factorial experimental design, and can be in an online setting, if online algorithms are used, i.e. bandit algorithms. In this example, there are three *factors*, each with two *choices*, also called *3 factors* $\times$ *2 choices per factor* factorial design. Note that some questions may be raised due to the assumption on the tree structure. In practice, sometimes the orders of factors is predefined by the problem, i.e. graphical design for mobile applications and adaptive health

[6]In MVB problems, every super arm mentioned here denotes which arm selected of every variable (factor), as a large binary vector, signaling whether selected or not. The super arm is equivalent to the decision set in MVB problems.

intervention optimization. In this case, we would not be worried that the assumption is incorrect, or deterministic.

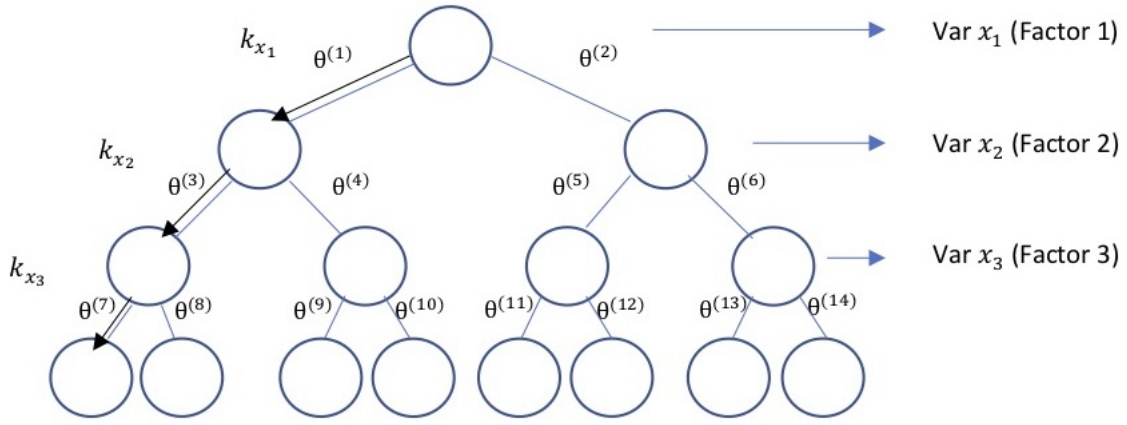


Figure 2.1: MVB in Tree Structure

2.7 Linear v.s. Non-Linear Reward Function

In Section 2.5, we shortly introduce the connection between combinatorial MAB and MVB, and how we can use linear bandit algorithms to solve combinatorial MAB with bandit feedback, which is equivalent to solving MVB problems. As shown in Figure 2.1, assume a search tree with underlying parameters $\boldsymbol{\theta}_\star = (\theta^{(1)}, \theta^{(2)}, ..., \theta^{(14)})$, which would affect the reward by making different decisions at every state, i.e, at state s_1, selecting between the two choices result in different influence in the reward since only $\theta^{(1)}$ or $\theta^{(2)}$ would affect the reward received, but not both. In combinatorial bandit settings, to solve the problem, an estimated parameter $\tilde{\boldsymbol{\theta}}$ is constructed based on collected information. At every time step, a super arm S is selected from the super arm space, associated with its feature vector M. In this example, a feature vector M for super arm S, composed of left decisions at every state in the search tree, could be $[1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$. Therefore,

the cumulative expected regret is defined as

$$\mathfrak{R}_T = \sum_{t=1}^{T} \mathbb{E}[R(S_\star)] - \mathbb{E}[R(S_t)], \tag{2.12}$$

where $S_\star$ is the optimal super arm that could achieve zero regret, and S_t is the super arm selected at time step t.

If the reward function is defined same as Equation 1.21, a linear reward function with all weights equal to one, the reward is a linear combination of the feature vector and the underlying parameter $\theta_\star$:

$$\mathbb{E}[R(S)] = \langle M, \theta_\star \rangle \tag{2.13}$$

If the reward function is non-linear, more common than linear reward function in practice, same as Equation 1.22, the max reward function is defined as

$$\mathbb{E}[R(S)] = \max_{i \in \{1,2,\ldots,|\theta|\}} M_i \cdot \theta_\star^{(i)} \tag{2.14}$$

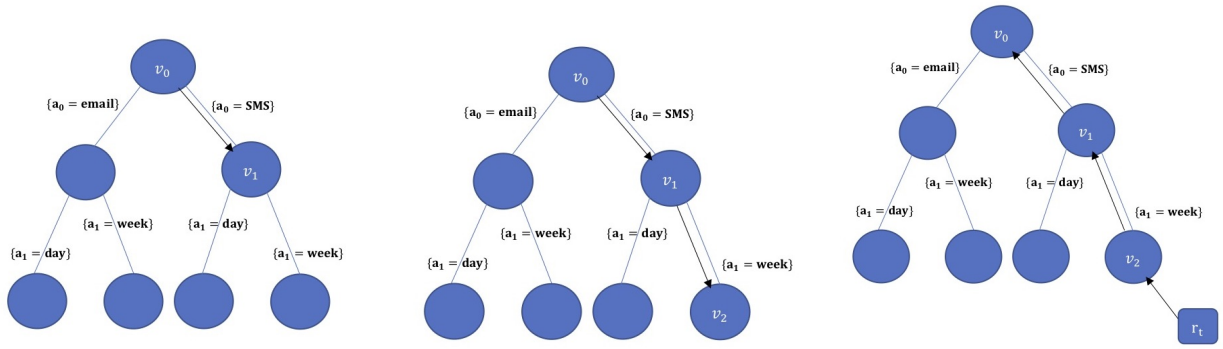
Furthermore, the cumulative regret is defined as

$$\mathfrak{R}_T = \sum_{t=1}^{T} R(S_\star) - R(S_t) \tag{2.15}$$

2.8 Tree Search Bandit Algorithm for MVB

Bandit Algorithm for Tree Search (BATS), as a successful algorithm in board games, seems to be a good choice than combinatorial MAB with bandit feedback in MVB problems. The decision made for every specific variable can be seen as a visited node in the trajectory at every time step. Compared to combinatorial MAB, where each time the decision maker has to consider a significant number of super arms, using BATS, the decision maker should not consider a super arm, composed of decisions needed to be made for

several variables, but rather one variable at a time. With BATS, the computation can be significantly reduced. Since the information is backed up to every visited node in the trajectory when reaching the leaf, the information update is more efficient than in combinatorial MAB, where intuitively every available choice for every decision needs to be updated, especially, in linear bandits, it is considerably expensive to update the mean value matrix and the covariance matrix, compared to updating scalar values in BATS.



(a) SMS over Email **(b)** Week over Day **(c)** Update nodes visited

Figure 2.2: One episode of tree search

As shown in Figure 2.2, it is the simplest example of tree search for factorial experiments. Instead of obtaining the estimated rewards of every arm in Linear TS, using Algorithm 6, tree search presents to be a much more efficient method.

Chapter 3

Experiments

3.1 Formations

3.1.1 Standard Bandits

Standard bandits formation follows MAB problem. Details of MAB can be referred to Section 1.2. Treating every possible treatment combination as an arm, when the number of factors or choices per factor increases, the number of arms increases exponentially. Given M factors and N choices per factor, there exists N^M arms in the standard bandit setting.

3.1.2 Linear Bandits

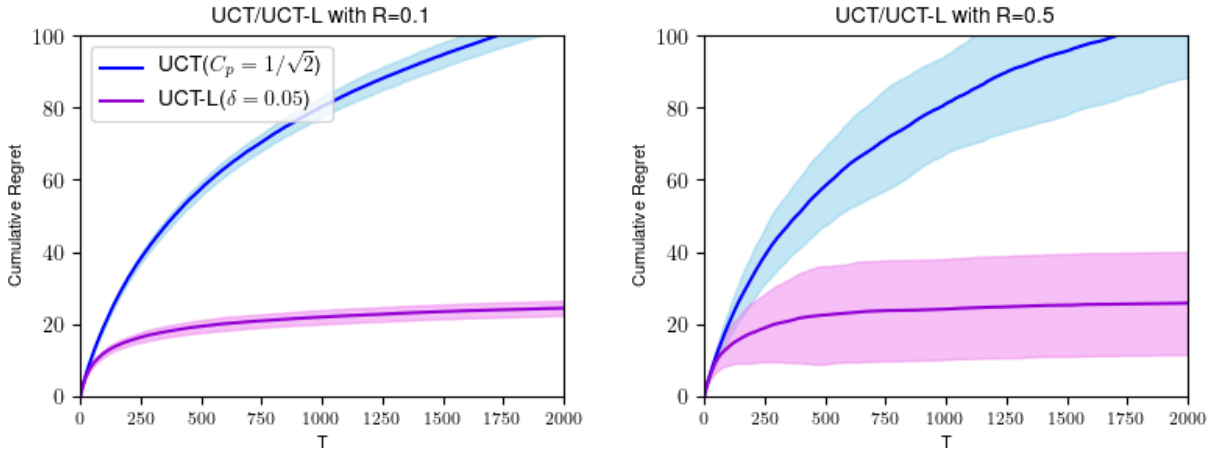
Linear bandits formation is a special case of MAB problem, and it makes an assumption on the linear reward function. This assumption is brittle, since the reward function is unknown and making the assumption is dangerous. Details of linear bandits can be referred to Section 1.6. In linear bandits setting, we assume there is an underlying parameter $\theta_\star$, whose vector dimension is the number of factors multiplied by the number of choices per factor. For example, in Figure 2.2, the dimension of the estimated parameter θ is six. The feature vectors of arms are binary vector, indicating whether or not the choice is made. Even the assumption of linearity is allowed to be made, computation-efficiency

is also a problem in linear bandits setting. Given M factors and N choices per factor, there exists N^M arms with binary feature vectors, of length $M \times N$ encoding the choice selected for each factor in linear bandits setting.

3.1.3 Tree Search Bandits

In tree search setting, we are able to get rid of the linearity assumption. In this setting, the sequential decision tree is explicitly represented and we aim to optimize the path in the tree. Details of BATS can be referred to Section 2.1. We use a popular algorithm, UCT. UCT treats each decision node as a separate standard bandit problem, picking the choice according to the UCB. In tree setting, computation-efficiency can be easily achieved.

3.2 UCT with Tighter Concentration Bounds

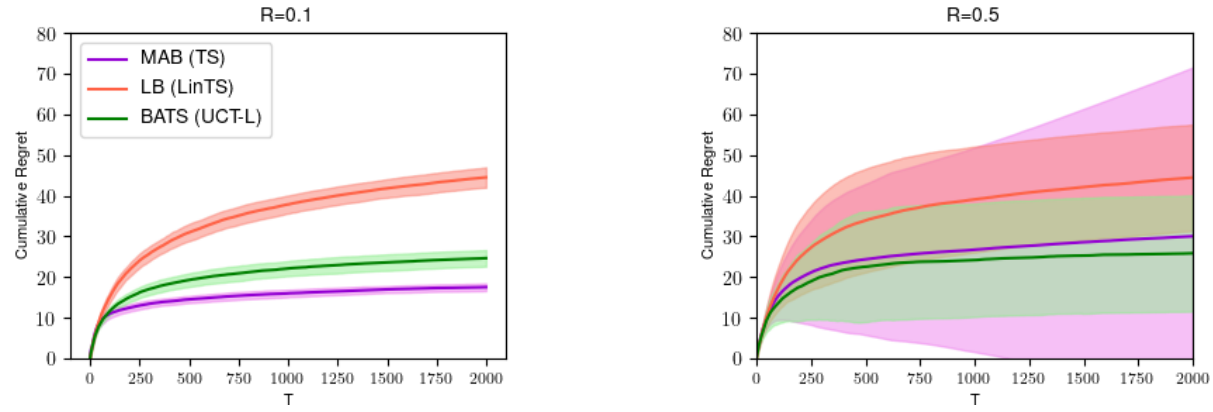


(a) 3 factors $\times$ 2 choices per factor

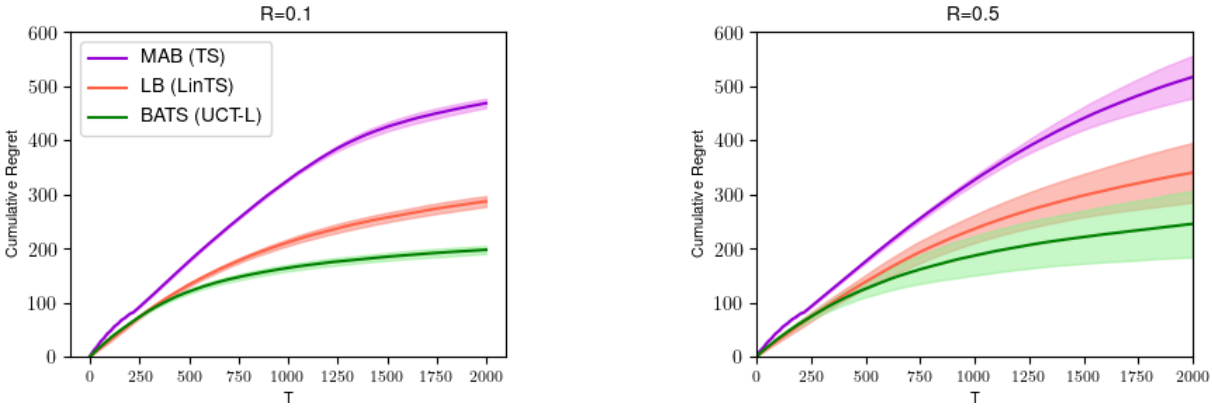
(b) 3 factors $\times$ 2 choices per factor

Figure 3.1: UCT v.s. UCT-Laplace

In this section, we will show that using the tighter concentration bounds proposed in [Abbasi-Yadkori et al., 2011] can significantly improve the performance of UCT for tree search proposed in [Kocsis and Szepesvári, 2006, Kocsis et al., 2006].



(a) 3 factors $\times$ 2 choices per factor



(b) 3 factors $\times$ 6 choices per factor

Figure 3.2: Varying Number of Choices per Factor

We tested the algorithms in less noisy and noisier environments, with standard deviation of noise R equal to 0.1 and 0.5.

As Figure 3.1 shows, UCT and UCT-Laplace are robust to variance noise. UCT-Laplace, based on tighter concentration bounds, outperform UCT with faster convergence.

3.3 Varying Number of Choices per Factor

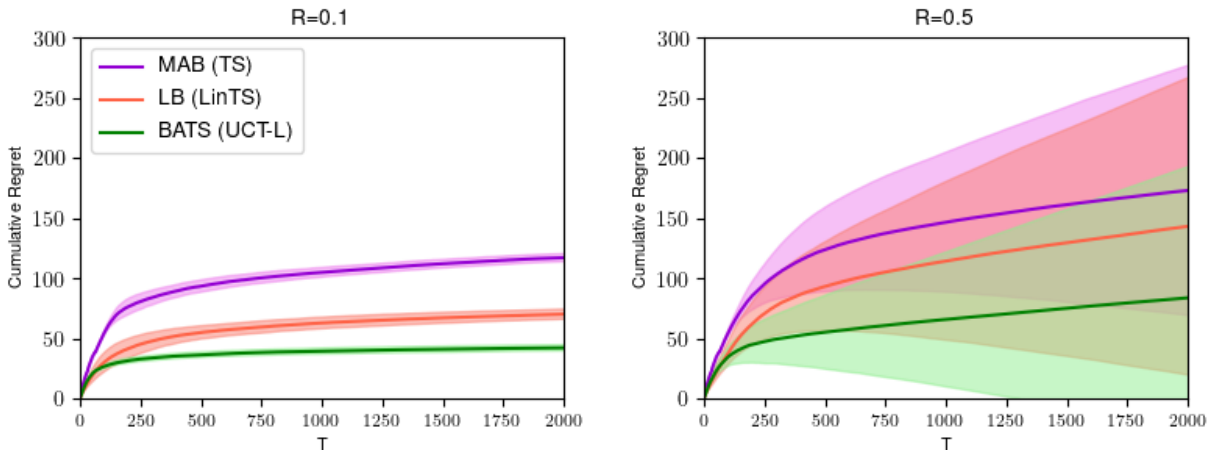
In this section, we will show that different numbers of choices per factor influence the performance of algorithms under standard bandit setting, linear bandit setting, and tree search bandit setting. We will compare the the best-performing algorithm in each setting.

In standard bandit setting, we select TS for MAB. In linear bandit setting, we select LinTS. In tree search bandit setting, we select UCT-Laplace.

As Figure 3.2 shows, when the number of arms is small, in Figure 3.2a, TS for MAB is still able to perform well. Although the noise increases, it seems less robust. However, when the number of arms increases, in Figure 3.2b, LinTS shows to be better than TS for MAB since it employs the mutual information, on the other hand, algorithms under standard bandit setting do not.

In both cases, algorithms under tree bandit setting can capture the underlying tree structure and shows its robustness and satisfactory performance.

3.4 Varying Number of Factors

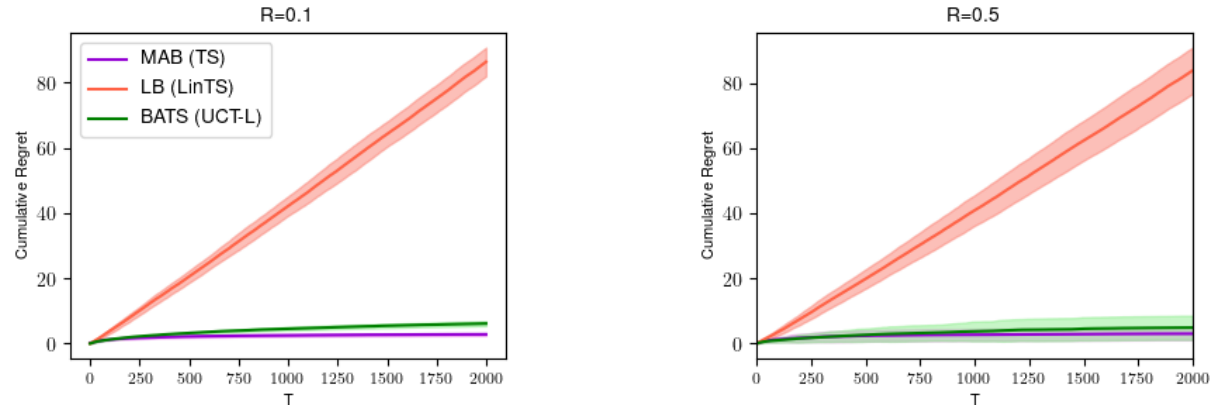


(a) 6 factors $\times$ 2 choices per factor

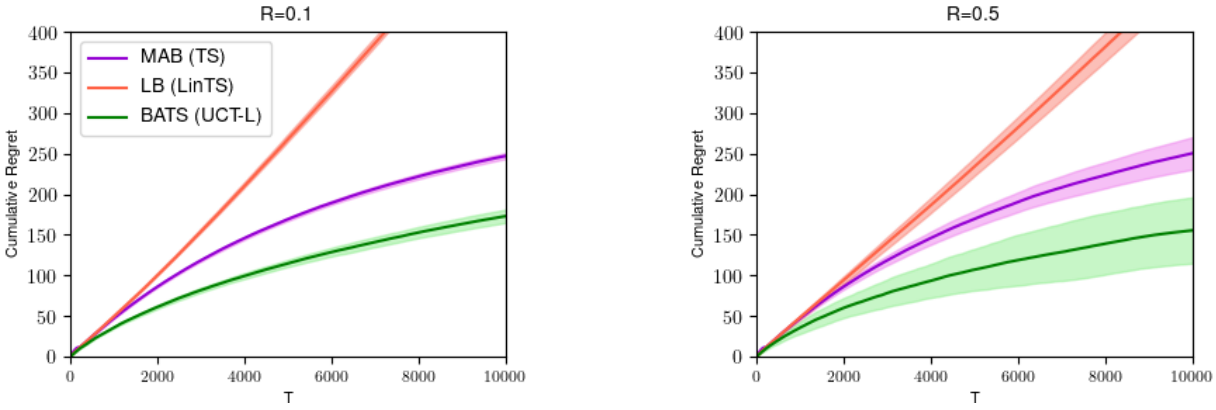
(b) 6 factors $\times$ 2 choices per factor

Figure 3.3: Varying Number of Factors

As shown in Figure 3.3, as the number of factors increase, TS under standard bandit setting does not perform well any more. UCT-Laplace is still robust to variance noise and outperform the other two algorithms. At this point, we can conclude that tree search bandit setting can capture the sequential decision-making tree structure and with little assumption, the formation is better than standard bandit setting and linear bandit setting.



(a) 3 factors $\times$ 2 choices per factor



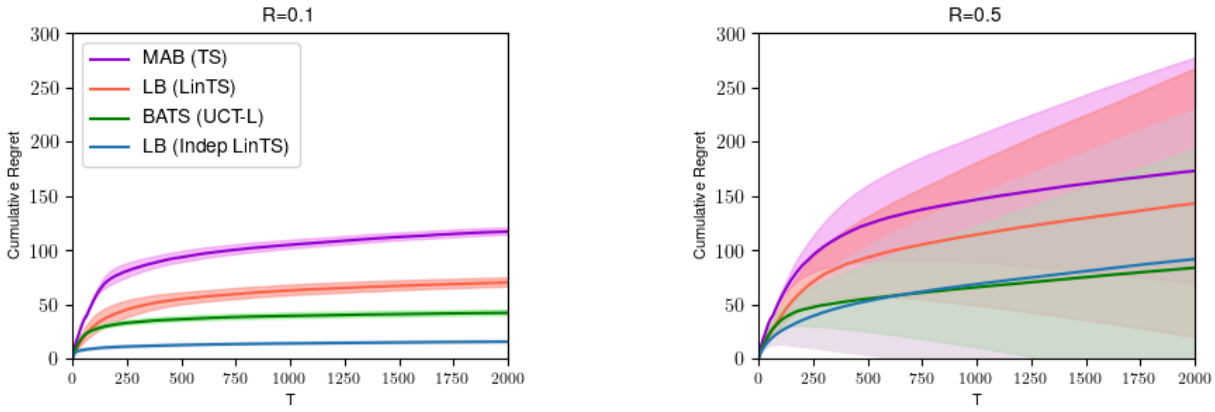
(b) 3 factors $\times$ 6 choices per factor

Figure 3.4: Max reward function

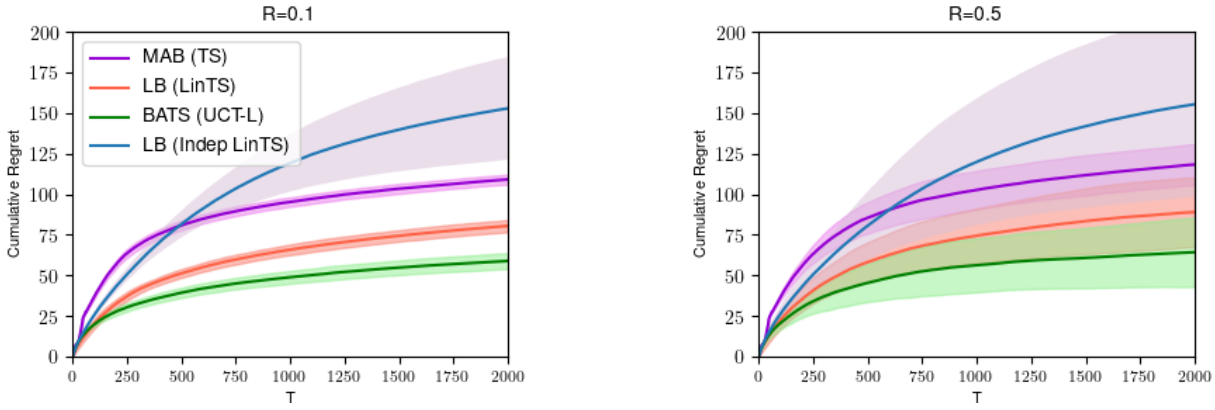
3.5 Non-Linear Reward Function

In this setion, we will test performance under non-linear reward function, such as max reward function, to show how LinTS breaks in a general setting.

In Figure 3.4, we compare the less and more number of arms under a non-linear reward function. As Figure 3.4a shows, similarly as in Figure 3.2, when the number of arms is small, TS under standard bandit setting seems sufficient for the problem. LinTS, with no doubt, cannot capture the non-linearity of the rewards received at all. As the number of arms increases, in Figure 3.4b, UCT-Laplace under tree bandit setting seems to be robust and perform well, compared to LinTS and TS.



(a) 6 factors $\times$ 2 choices per factor (Independence)



(b) 6 factors $\times$ 2 choices per factor (Dependence)

Figure 3.5: Independence v.s. Dependence

Using the experiments, we show that in non-linear reward setting, algorithms under linear bandit formulation appear to show the sub-optimal performance. If the number of arms is large, algorithms under tree search bandit setting seems to be a good choice due to the capability to capture the tree structure without a strong assumption.

3.6 Independence of Factors

In all experiments all this section, we assume there is a dependence among factors. In other words, the choices of the preceding factors affects the choices of the following factors. The choices might be different following different choices made previously.

If we assume there is independence among factors, it would be beneficial to linear algorithms due to the easiness of computation. Instead of a geometric sum of the number of choices per factor as the feature vector dimension, the feature vector dimension would be the product of number of factors and number of choices per factor[1].

In this section, we will show two experiments under one the assumption is held and under other not.

As Figure 3.5 shows, in Figure 3.5a, when the independence assumption is not broken in the environment, in other words, the decision for preceding factors would not affect the choices of the following factors, Independent LinTS under linear bandit formation seems to be comparable with the algorithm under tree search bandit formulation. We can conclude that in the independent setting, it is worth to use Independent LinTS.

However, in Figure 3.5b, when the assumption of independence of factors is broken, the performance of Independent LinTS is not satisfactory. It is even not comparable with the worst-performing algorithm, TS under standard bandit setting. What is more, we observe that algorithm under tree search bandit is consistently robust across all different settings.

[1]We assume the number of choices per factor is same for all factors here.

Bibliography

[Abbasi-Yadkori et al., 2011] Abbasi-Yadkori, Y., Pál, D., and Szepesvári, C. (2011). Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 2312–2320.

[Abeille et al., 2017] Abeille, M., Lazaric, A., et al. (2017). Linear thompson sampling revisited. *Electronic Journal of Statistics*, 11(2):5165–5197.

[Agrawal and Goyal, 2013] Agrawal, S. and Goyal, N. (2013). Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pages 127–135.

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

[Baker et al., 2017] Baker, T. B., Smith, S. S., Bolt, D. M., Loh, W.-Y., Mermelstein, R., Fiore, M. C., Piper, M. E., and Collins, L. M. (2017). Implementing clinical research using factorial designs: a primer. *Behavior therapy*, 48(4):567–580.

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[Cesa-Bianchi and Lugosi, 2006] Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge university press.

[Cesa-Bianchi and Lugosi, 2012] Cesa-Bianchi, N. and Lugosi, G. (2012). Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422.

[Chapelle and Li, 2011] Chapelle, O. and Li, L. (2011). An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*.

[Collins et al., 2007] Collins, L. M., Murphy, S. A., and Strecher, V. (2007). The multiphase optimization strategy (most) and the sequential multiple assignment randomized trial (smart): new methods for more potent ehealth interventions. *American journal of preventive medicine*, 32(5):S112–S118.

[Combes et al., 2015] Combes, R., Shahi, M. S. T. M., Proutiere, A., et al. (2015). Combinatorial bandits revisited. In *Advances in Neural Information Processing Systems*, pages 2116–2124.

[Coquelin and Munos, 2007] Coquelin, P.-A. and Munos, R. (2007). Bandit algorithms for tree search. *arXiv preprint cs/0703062*.

[Dani et al., 2008] Dani, V., Hayes, T. P., and Kakade, S. M. (2008). Stochastic linear optimization under bandit feedback.

[Durand and Gagné, 2017] Durand, A. and Gagné, C. (2017). Estimating quality in multi-objective bandits optimization. *arXiv preprint arXiv:1701.01095*.

[Filippi et al., 2010] Filippi, S., Cappe, O., Garivier, A., and Szepesvári, C. (2010). Parametric bandits: The generalized linear case. In *Advances in Neural Information Processing Systems*, pages 586–594.

[Hill et al., 2017] Hill, D. N., Nassif, H., Liu, Y., Iyer, A., and Vishwanathan, S. (2017). An efficient bandit algorithm for realtime multivariate optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1813–1821. ACM.

[Jun et al., 2017] Jun, K.-S., Bhargava, A., Nowak, R., and Willett, R. (2017). Scalable generalized linear bandits: Online computation and hashing. In *Advances in Neural Information Processing Systems*, pages 99–109.

[Kalai and Vempala, 2005] Kalai, A. and Vempala, S. (2005). Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307.

[Kaufmann and Koolen, 2017] Kaufmann, E. and Koolen, W. M. (2017). Monte-carlo tree search by best arm identification. In *Advances in Neural Information Processing Systems*, pages 4897–4906.

[Kocsis and Szepesvári, 2006] Kocsis, L. and Szepesvári, C. (2006). Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer.

[Kocsis et al., 2006] Kocsis, L., Szepesvári, C., and Willemson, J. (2006). Improved monte-carlo search. *Univ. Tartu, Estonia, Tech. Rep*, 1.

[Loiselle and Ahmed, 2017] Loiselle, C. G. and Ahmed, S. (2017). Is connected health contributing to a healthier population? *Journal of medical Internet research*, 19(11).

[Montgomery, 1995] Montgomery, D. C. (1995). Design of experiments. *New York*, pages 225–364.

[Neu and Bartók, 2013] Neu, G. and Bartók, G. (2013). An efficient algorithm for learning with semi-bandit feedback. In *International Conference on Algorithmic Learning Theory*, pages 234–248. Springer.

[Robbins et al., 1952] Robbins, H. et al. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535.

[Rusmevichientong and Tsitsiklis, 2010] Rusmevichientong, P. and Tsitsiklis, J. N. (2010). Linearly parameterized bandits. *Mathematics of Operations Research*, 35(2):395–411.

[Sankararaman and Slivkins, 2017] Sankararaman, K. A. and Slivkins, A. (2017). Combinatorial semi-bandits with knapsacks. *arXiv preprint arXiv:1705.08110*.

[Shaikh, 2019] Shaikh, Modiri, W. R. (2019). Balancing student success and inferring personalized effects in dynamic experiments. In *The 12th International Conference on Educational Data Mining*, pages 1–9. EDM.

[Takimoto and Warmuth, 2003] Takimoto, E. and Warmuth, M. K. (2003). Path kernels and multiplicative updates. *Journal of Machine Learning Research*, 4(Oct):773–818.

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

[Valko et al., 2013] Valko, M., Korda, N., Munos, R., Flaounas, I., and Cristianini, N. (2013). Finite-time analysis of kernelised contextual bandits. *arXiv preprint arXiv:1309.6869*.

[Yates, 1978] Yates, F. (1978). *The design and analysis of factorial experiments*. Imperial Bureau of Soil Science Harpenden.