Multi-Armed Bandits

Keshmiri, Afshein Lau, Justin Lee, Colman Li, Linze

Shi, Lunzhi Siu, Lucas

Supervisor: Dr Ciara Pike-Burke
Imperial College London

June 2022

Contents

Chapter 1

Introduction

1.1 List of Symbols

a	The action, or known as the arm
a_t	The arm chosen at round t
a^*	The optimal arm
K	The total possible arms
T	The rounds that are played
n_j	The number of times arm j has been chosen
P_a	The reward distribution of action a
$\mu(a)$	The mean reward of arm a
$R(T)$	The cumulative regret at round T
$\pi(\theta)$	The prior distribution of θ
$\pi(\theta \mid x)$	The posterior distribution of θ given x

1.2 Motivation

Imagine a game in which one is to play K slot machines or bandits each with a different expected reward, and try to maximise the cumulative reward. We know nothing about the reward distributions of any machines at the start, but we wish to in some way gain just enough information quick enough to play with the optimal strategy and leave with the maximum payoff (often in a limited amount of turns). An application is how Netflix recommends their content to a user according to a user's engagement to the 'personalised' content feed.

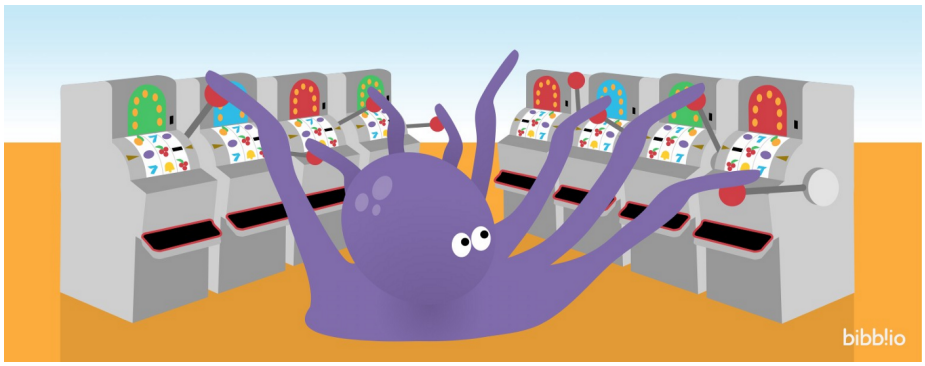


Figure 1.1: K-armed bandits [1]

1.3 Problem Formulation and Regret

The multi-armed bandits problem is a class of sequential resource allocation problems concerned with allocating one or more resources among several alternative (competing) projects [2]. This problem leads to the exploration vs exploitation dilemma: we either keep exploiting what we think gives us a high reward, or sacrifice some resources and explore to acquire new information for potentially better rewards.

In this paper, we will mainly look at stochastic bandits. The stochastic bandit problem has been analysed from two different perspectives: a frequentist approach, where the parameter is a deterministic unknown quantity, and a Bayesian approach, where the parameter is drawn from a prior distribution [3]. We first present methods derived from the frequentist perspective, including the ϵ-greedy algorithm and the upper confidence bound algorithm, followed by the Bayesian approaches (namely Thompson Sampling and randomised probability matching). In the end, we will look to extend this problem further and have a look at a non-stochastic example and also continuum-armed bandits.

Now let us formulate the multi-armed bandits problem formally. A player implements an algorithm that has K possible arms to choose from, and there are in total T rounds/turns to play, for some known K and T. In each round, the algorithm chooses an arm and observes a reward for this arm. The algorithm's goal is to maximise its total reward over the T rounds. We make three essential assumptions [4]:

- The algorithm observes only the reward for the selected action, and nothing else. In particular, it does not observe rewards for other actions that could have been selected.
- The reward for each action is independent and identically distributed (i.i.d). For each action a, there is a reward distribution D_a over the reals.

Every time this action is taken to 'play' this arm, the reward is sampled independently from its respective distribution. The reward distributions are initially unknown to the player or the algorithm.

- Pre-round rewards are bounded; the restriction to the interval $[0, 1]$ is for simplicity.

We are primarily interested in the mean reward vector $\mu \in [0, 1]^K$, where $\mu(a) = E[D_a]$ is the mean reward of arm a. We also analyze the performance of each algorithm, which is quantified by the *regret*. We denote the reward of the best arm as μ^*, so the best-arm benchmark is the expected reward of always playing an optimal arm: $\mu^* \cdot T$. Comparing it to the algorithm's cumulative reward, we define the regret at round T:

$$R(T) = \mu^* \cdot T - \sum_{t=1}^{T} \mu(a_t) \tag{1.1}$$

Since a_t, the arm chosen at round t, has its value realised by sampling from a random distribution, $R(t)$ is also a random variable. Hence we typically talk about the *expected regret* $E[R(T)]$ [4].

Chapter 2

Frequentist Algorithms

2.1 ϵ-Greedy Algorithm

Within our frequentist approach we will assume that all the machines are distributed with fixed unknown means $\mu(i)_{1\leq i\leq K}$. We aim to minimise the expected regret of our sequence of actions.

A naive approach may be to strike a balance between exploitation (here denoting selecting the machine with greatest sample mean) and exploration (denoting selecting a random machine uniformly) at each stage using a Bernoulli r.v, so that we expect to explore for an ϵ-proportion of our actions.

We can use pseudo-code to aid our understanding of algorithms such as this one.

Algorithm 1 ϵ-greedy Algorithm

Require: ϵ, No. of arms: K, Total Turns: T

while $t \in \{1, 2, .., K\}$ **do**
    $a_t = t$
end while
while $t \in \{K+1, K+2, ..., T\}$ **do**
    $X_t \sim Bern(\epsilon)$
    if $X_t = 0$ **then**:
        $a_t \leftarrow argmax(\{\bar{x}_i\}_{1 \leq i \leq K})$
    else
        $a_t \sim Unif(\{1, 2, 3, ...K\})$
    end if
end while

$*$It is necessary to first play all machines once initially so all machines have a sample mean.

A benefit to this algorithm is that we will always be exploring for some proportion of the game, so by the Weak Law of Large Numbers our sample means will converge in probability to the true means, and so our exploitation stage will (theoretically at some point) select the optimal machine with high probability.

However, due to the exploration with probability ϵ, our expected regret is bounded below by $\epsilon \cdot T \cdot \sum_{n=1}^{k} \Delta_j$, so the expected regret will grow (at best) linearly i.e. this algorithm does not make preferences in exploration to machines that less is known about and consequently even when we are confident in our exploitation stage, is equally likely to explore.

We can improve upon this.

2.2 Upper Confidence Bound Algorithm

An Upper Confidence Bound (UCB) algorithm will calculate upper confidence bounds for all machines and select the machine with the highest upper confidence bound.

Unlike the ϵ-greedy algorithm or the Bayesian methods we will see next, UCB is an example of a deterministic algorithm (i.e. given the history of rewards one can say with certainty what the following action will be). UCB is also labelled an 'optimistic' strategy since we choose the machine with the greatest 'potential'.

Algorithm 2 UCB Algorithm

Require: No. of arms: K, Total Turns: T

1: **while** $t \in \{1, 2, ..K\}$ **do**
2: $\quad a_t = t$
3: **end while**
4: **while** $t \in \{K+1, K+2, ..., T\}$ **do**
5: $\quad a_t \leftarrow argmax\left(\bar{x}_j + \sqrt{\frac{\alpha \cdot log(t-1)}{2n_j}}\right)$
6: **end while**

We can show these are Upper Confidence Bounds (line 5) using Hoeffding's inequality. Hoeffding's inequality implies that for independent random variables (X_n) s.t. $0 \leq X_i \leq 1$ almost surely, letting $S_n = \sum_{i=1}^{n} X_i,\ \forall a > 0$:

$$Pr(S_n \leq E[S_n] - a) \leq e^{-2a^2/n}$$

Now let a = $\sqrt{n_j \cdot \frac{\alpha}{2} \cdot log(t-1)}$, we get:

$$Pr\left(\mu(j) \geq \bar{X}_j + \sqrt{\frac{\alpha \cdot log(t-1)}{2 \cdot n_j}}\right) \leq (t-1)^{-\alpha}$$

α is a confidence value - the greater it is the more confident we can be that our true mean lies beneath our UCB. The choice of $\alpha = 4$ gives the so called UCB1 algorithm.

The UCB index consists of a exploitation term (the sample mean) and an exploration term. The $\sqrt{n_j}$ in the denominator of our exploration term causes the algorithm to favour exploring machines that we haven't visited often. Note as time progresses the exploration term becomes less significant as $\lim_{n\to\infty} \frac{log(n)}{n} = 0$.

For UCB1 the regret is bounded above [5] by

$$\sum_{j:\mu(j)<\mu^*} \frac{8log(T)}{\Delta_j} + \left(1 + \frac{\pi^2}{3}\right) \sum_{j=1}^{K} \Delta_j$$

The bound is formed by bounding the expected number of machine trials for sub-optimal machines:

$$E[T_j(t)] \leq \frac{8}{\Delta_j^2} log(t) + 1 + \frac{\pi^2}{3}$$

Note: The only assumption we have made is that the support of the machines is $[0, 1]$ (Hoeffding's inequality only requires a sub-Gaussian reward distribution, which is satisfied by a support of $[0, 1]$, so this would also work for Normally distributed rewards).[6]

We can further improve on the UCB1 algorithm (for Bernoulli machines) using a UCB for the variance as well:

$$V_j = \frac{1}{n_j} \sum_{\tau=1}^{n_j} X_{j,\tau}^2 - \bar{X}_j^2 + \sqrt{\frac{2log(t-1)}{n_j}}$$

So that we are now maximizing:

$$\bar{X}_j + \sqrt{\frac{log(t-1)}{n_j} min(1/4, V_j)}$$

*Note: since rewards are bounded to $[0, 1]$ we have an upper bound on the variance at $1/4$.

Although we cannot prove a regret bound for the strategy it outperforms UCB1 in all of our cases for Bernoulli machines. We call it a fine-tuned UCB or UCB-tuned.

The advantage of the UCB-tuned algorithm over UCB1 is that now the variance is included in the exploration term which allows us to favour terms with greater variance (more so than UCB1), of which more trials are needed for us to be confident in their mean parameter estimation. However this advantage does not necessarily hold to other distributions.

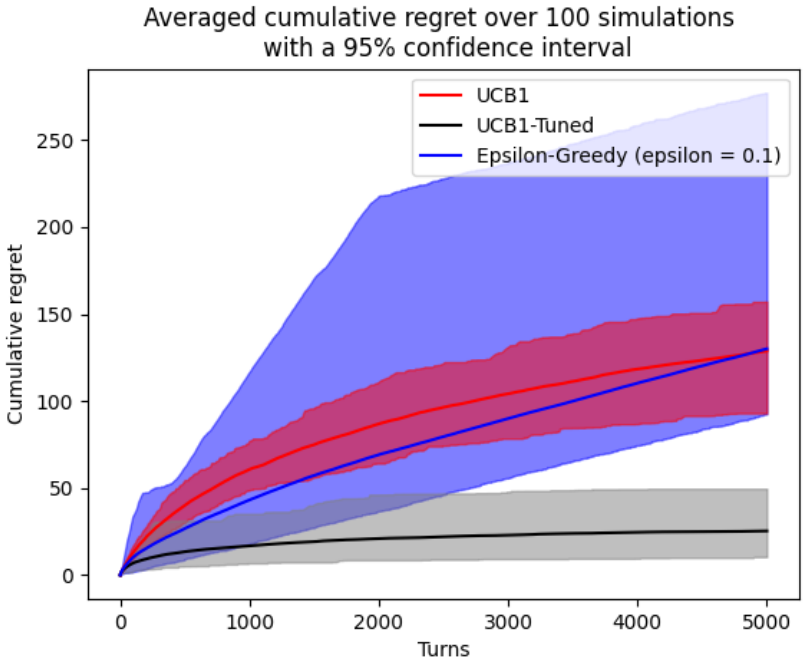


Figure 2.1: Comparison with Bernoulli machines with means (0.33, 0.55, 0.6)

Chapter 3

Bayesian Algorithms

3.1 Bayesian Inference

Assume we have a random variable X whose distribution lies in a statistical model $\{P_\theta \,|\, \theta \in \Theta\}$. In our example, X refers to the reward while θ refers to the mean reward from a certain machine.

As opposed to the frequentist approach, where θ is regarded as fixed (though not necessarily known), the Bayesian approach treats θ as a random variable. The prior distribution $\pi(\theta)$ is the distribution of θ before we observe any data. There are many ways to pick a prior - it can come from our personal experiences, beliefs and knowledge, or from the opinion of some expert in the field. On the other hand, if we do not know much about the situation, we may assume a "uniform prior" which assigns equal weights to possible values of θ. [7]

The idea of the Bayesian approach is to update our model for θ after observing data to obtain the posterior $\pi(\theta|x)$. This is done using Bayes' theorem:

$$\pi(\theta|x) = \frac{f(x|\theta)\pi(\theta)}{f_X(x)} \propto f(x|\theta)\pi(\theta) \qquad (*)$$

where $f_X(x) = \int_\Theta f(x|\theta')\pi(\theta')d\theta'$ is the marginal likelihood of X.

Let $Y_1, ..., Y_n$ be the rewards arriving sequentially obtained from playing a certain machine repeatedly. Suppose after time n, we have obtained the posterior $\pi_n(\theta) \coloneqq \pi(\theta|\boldsymbol{y}_n)$, where $\boldsymbol{y}_n = (y_1, ..., y_n)$ is the vector of observed rewards (up to this current time step). If we observe an additional reward y_{n+1} from the same machine, we can update our distribution by using (*) with the original prior and $\boldsymbol{y}_{n+1}$ to obtain

$$\pi_{n+1}(\theta) \propto f(\boldsymbol{y}_{n+1}|\theta)\pi(\theta)$$

Alternatively, we can adopt a sequential approach [8] by using $\pi(\theta|\boldsymbol{y}_n)$ as the

prior and updating it with the new observation y_{n+1} to obtain

$$\tilde{\pi}_{n+1}(\theta) \propto f(y_{n+1}|\theta, \boldsymbol{y}_n)\pi_n(\theta)$$

As shown below, these two methods for updating are equivalent:

$$\begin{aligned}\tilde{\pi}_{n+1}(\theta) &\propto f(y_{n+1}|\theta, \boldsymbol{y}_n)\pi_n(\theta) \\ &\propto f(y_{n+1}|\theta, \boldsymbol{y}_n)f(\boldsymbol{y}_n|\theta)\pi(\theta) \\ &= f(y_{n+1}, \boldsymbol{y}_n|\theta)\pi(\theta) \\ &= f(\boldsymbol{y}_{n+1}|\theta)\pi(\theta)\end{aligned}$$

It is convenient to pick a conjugate prior, which is defined to be a prior whose posterior distribution is in the same family of distributions as the prior distribution. This simplifies calculations as it gives a closed form for the posterior, which allows the marginal likelihood $f_X(x)$ to be computed in a much easier way.

The following result will be useful for the Bayesian algorithms introduced later:

Proposition 3.1.1 (Bernoulli distribution with beta prior)**.** Suppose $X \sim Bern(\theta)$ and a $Beta(\alpha, \beta)$ prior is assumed for θ, where α, β are known. Then given x, a realisation of X, the posterior distribution $\theta|x \sim Beta(\alpha+x, \beta+1-x)$.

Proof.

$$\begin{aligned}\pi(\theta|x) &\propto f(x|\theta)\pi(\theta) \\ &= \theta^x(1-\theta)^{1-x}\theta^{\alpha-1}(1-\theta)^{\beta-1} \\ &= \theta^{\alpha+x-1}(1-\theta)^{\beta+1-x-1}\end{aligned}$$

□

It follows that the beta distribution is a conjugate prior for the Bernoulli distribution. We will be conducting analysis of Bayesian methods on Bernoulli machines in this paper, assuming the results from this proposition.

3.2 Greedy Bayesian Algorithms

One may wish to start to explore implementing the Bayesian approach to multi-armed bandits using a simple and intuitive method. The greedy Bayesian method borrows its idea from non-Bayesian methods discussed in the previous section, and it can be seen as a naïve precursor to methods such as Thompson Sampling.

The algorithm typically involves the following steps: (1) Estimate a model from existing data/history of data. (2) Make the 'optimal' decision in the current time step based on the model. One may also wish to introduce an element of 'randomness' where each time step has a certain probability that instead of exploiting the 'best' option, we randomly explore another machine in an attempt

to gather more data. This is similar to the ϵ-greedy approach in a non-Bayesian setting.

Defining an optimal choice is sometimes subject to a player's manipulation. An example would be to use UCB: in the non-Bayesian setting, there is no prior distribution assumption on the reward so we must rely on Hoeffding's Inequality as a tool for estimation. In a Bayesian setting, we work off of a prior distribution upfront, giving us more to work with to achieve an upper bound estimation. As an example, if at the current time step we estimate the mean reward μ of a machine is distributed normally as follows, we can then find a (roughly) 95% upper confidence bound $\hat{U}$ to be the mean plus twice the standard deviation of the assumed distribution.

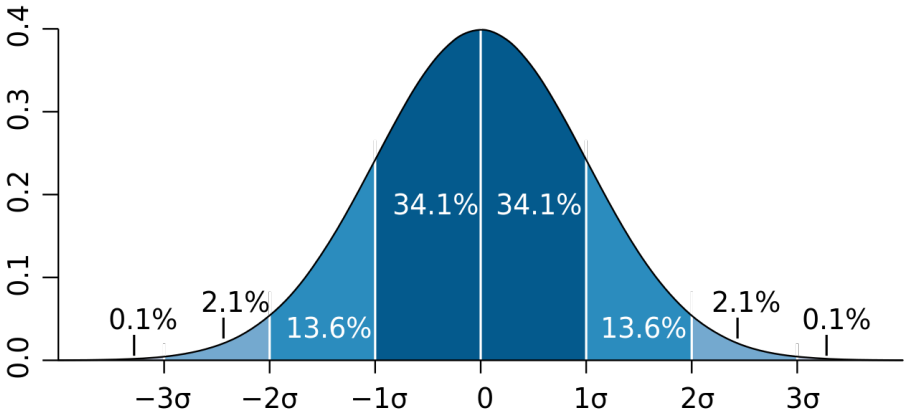


Figure 3.1: Density of a standard normal deviation

ϵ-Greedy Bayesian on Bernoulli Bandits with UCB

We implement this method on Bernoulli bandits assuming a prior distribution (e.g. $Beta(1,1)$) on the parameters of the reward distribution for each arm, where each arm i gives a payoff with distribution $Bern(\theta_i)$. The algorithm is outlined below:

Algorithm 3 Greedy Bayesian for Bernoulli Bandits

Require: Number of arms K, horizon T, prior parameters (α_i, β_i) of each θ_i

1: **for** $t = 1, 2, ..., T$ **do**
2: **for** $a = 1, 2, ..., K$ **do**
3: Compute $\hat{U}_a$ for θ_a from $\hat{\theta_a} \sim Beta(\alpha_a, \beta_a)$ ▷ Compute UCB
4: **end for**
5: **if** (with $1 - \epsilon$ probability) exploit **then**
6: $x_t \leftarrow argmax_a \hat{U}_a$
7: Choose arm x_t and observe the reward r_t
8: **else if** (with ϵ probability) explore **then**
9: Choose random arm x_t excl. $argmax_a \hat{U}_a$ and observe the reward r_t
10: **end if**
11: $(\alpha_{x_t}, \beta_{x_t}) \leftarrow (\alpha_{x_t} + r_t, \beta_{x_t} + 1 - r_t)$ ▷ Update posterior
12: **end for**

Note: the posterior $(\alpha_{x_t}, \beta_{x_t})$ is used as a prior for the subsequent Bayesian update. In line 6 of the algorithm, we are picking the optimal arm using UCB on Beta distributions. For a given arm at a given time step, we have a prior distribution for the parameter, and the n%-UCB gives us the upper bound for which the estimated parameter has probability n% taking a value lower than it. We can obtain this computationally using Beta inverse (CDF).

3.2.1 Application

We now apply the Greedy Algorithm to the scenario where there are 3 Bernoulli machines A, B, C with mean payoff $0.33, 0.55$ and 0.6 respectively and an initial prior uniform prior distribution $Beta(1, 1)$, which is equivalent to $Unif(0, 1)$. The explore probability is set to 0.02 and the upper confidence percentile is set at 95%.

The following graph visualises the progression of our parameter estimation as we go through more and more turns in a single simulation. Roughly speaking, the more concentrated the distribution is, the more knowledge we have on the actual mean of the machine. Since machine A has a low sample mean, we play it less often and hence have less information on its true mean, leading to a distribution with large spread, which becomes more apparent as we progress through the turns. We observe that the parameter estimation curves become more and more concentrated around the true mean, and after 1000 turns, the distribution is almost concentrated for machine C. The algorithm does seem to have an inability to make decisive choices early on in an simulation, which can lead to instability issues that we will discuss next.

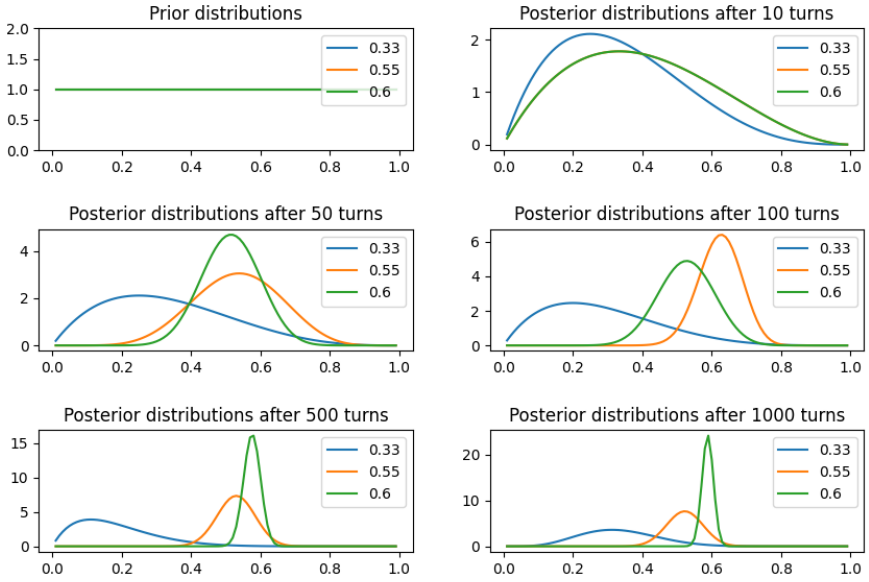


Figure 3.2: pdf of posterior distributions

3.2.2 Pitfalls

Assuming we would like to waste no resources to explore randomly, hence always playing the optimal arm each turn. Then, once the algorithm is set on exploiting a particular machine, it fails to account for the uncertainty in the expected reward of other machines, since there is no active exploration. As a result, one would observe the algorithm to settle on an 'incorrect' and sub-optimal machine to exploit indefinitely. The following plots illustrate a simulation where this phenomenon has occurred (one would expect to see a linear regret curve).

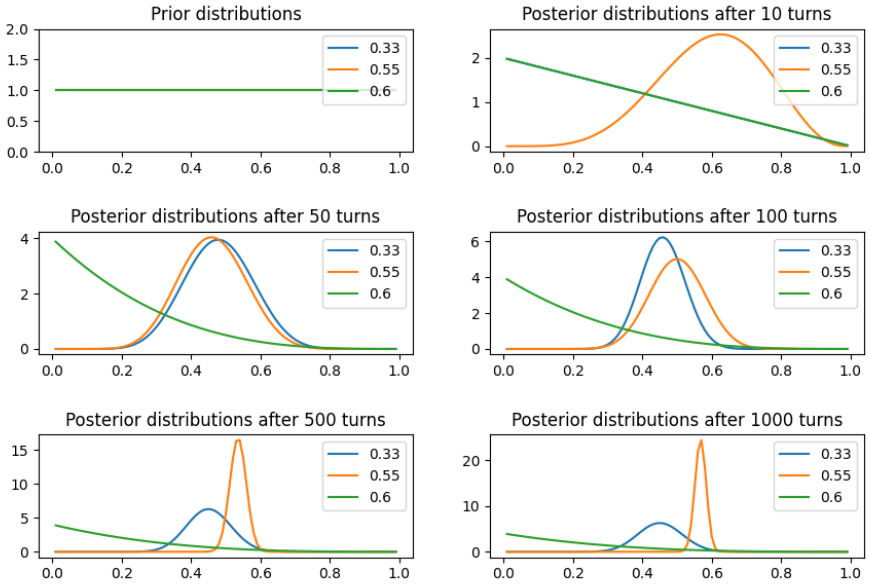


Figure 3.3: pdf of posterior distributions

In our ϵ-Greedy Bayesian algorithm, we encourage active exploration via *dithering* - an approach of randomly perturbing actions [9] that can be manipulated by the user. A downside to this is that it wastes resources by allocating a proportion of trials to randomly explore without considering how unlikely those actions are going to be optimal. We can visualise this in the following plot that shows us a history of actions taken to play the arms in 1000 turns ($\epsilon = 0.02$). Machine A has nearly no chance of being optimal hence does not deserve to be 'explored' further after a certain point (doing so only results in a guaranteed -proportion of 'wasted' resources), while the uncertainty between machines B and C deserves to be explored.

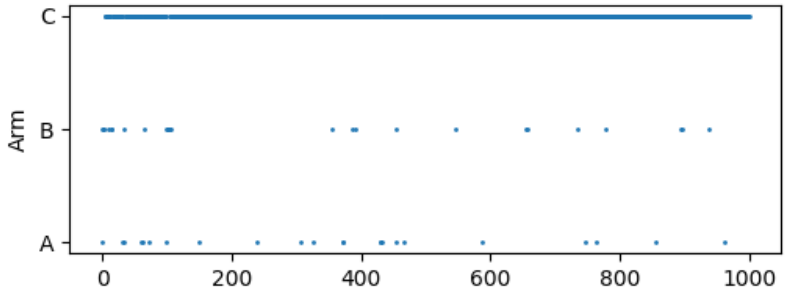


Figure 3.4: History of ϵ-greedy actions

Thompson sampling is a great alternative that employs a Bayesian approach which explores more intelligently and efficiently.

3.3 Thompson Sampling

The idea of Thompson sampling is to assume a prior distribution (e.g. $Beta(1,1)$) on the parameters on the reward distribution for each arm. At each time step and for each arm, we have a posterior distribution of the parameter. We then pick an arm randomly with reference to the probability of the arm being the best arm.

More precisely, for the Bernoulli bandit problem, where each arm i gives a payoff with distribution $Bern(\theta_i)$, the parameters in question would be θ_i. To pick an arm, we sample from the posterior distributions of θ_i and pick the arm with the greatest realised value. [10]. The algorithm [9] is given below:

Algorithm 4 Thompson Sampling for Bernoulli Bandits

Require: Number of arms K, horizon T, prior parameters (α_i, β_i) of each θ_i

for $t = 1, 2, ..., T$ **do**
  for $a = 1, 2, ..., K$ **do**
    Sample $\hat{\theta_a} \sim Beta(\alpha_a, \beta_a)$ ▷ Sample from posterior
  end for
  $x_t \leftarrow argmax_a \hat{\theta_a}$
  Choose arm x_t and observe the reward r_t
  $(\alpha_{x_t}, \beta_{x_t}) \leftarrow (\alpha_{x_t} + r_t, \beta_{x_t} + 1 - r_t)$ ▷ Update the posterior
end for

3.3.1 Application

We now apply Thompson Sampling to the scenario where there are 3 Bernoulli machines A, B, C with mean payoff $0.33, 0.55$ and 0.6 respectively and an initial prior uniform prior distribution $Beta(1,1)$, which is equivalent to $Unif(0,1)$.

We would like to explore how the posterior distribution changes as the game progresses with the plots given in the next page. The curves start off with large horizontal spread; as time goes on, the curves become more and more concentrated around a vertical line near the true mean reward just like in Greedy Bayesian.

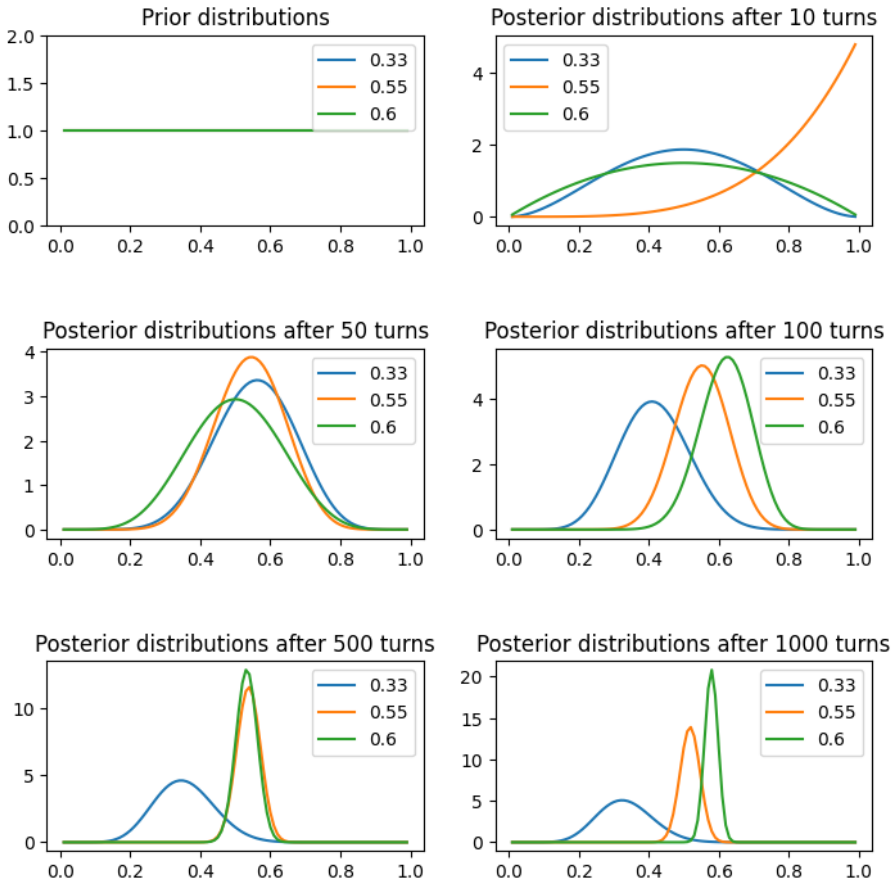


Figure 3.5: pdf of posterior distributions

Next, we would like to investigate the cumulative regret of the game. We provide the plots of cumulative regret against turn counts of 6 games, each representing the scenario stated above but with different time horizons. As turn count increases, regret grows more slowly - this is because we get more information on which machine is the best and pick incorrect machines less often. In fact, one can find more precise bounds on the expected regret as proven by Shipra Agrawal and Navin Goyal [10].

Proposition 3.3.1. In an K-armed bandit problem where θ_i is the mean of arm i (WLOG the first arm is the unique optimal arm) and $\Delta_i = \theta_1 - \theta_i$, the

expected regret at time T of the Thompson Sampling algorithm satisfies

$$E(\mathcal{R}(T)) \leq O\left(\left(\sum_{a=2}^{K} \frac{1}{\Delta_a^2}\right)^2 lnT\right)$$

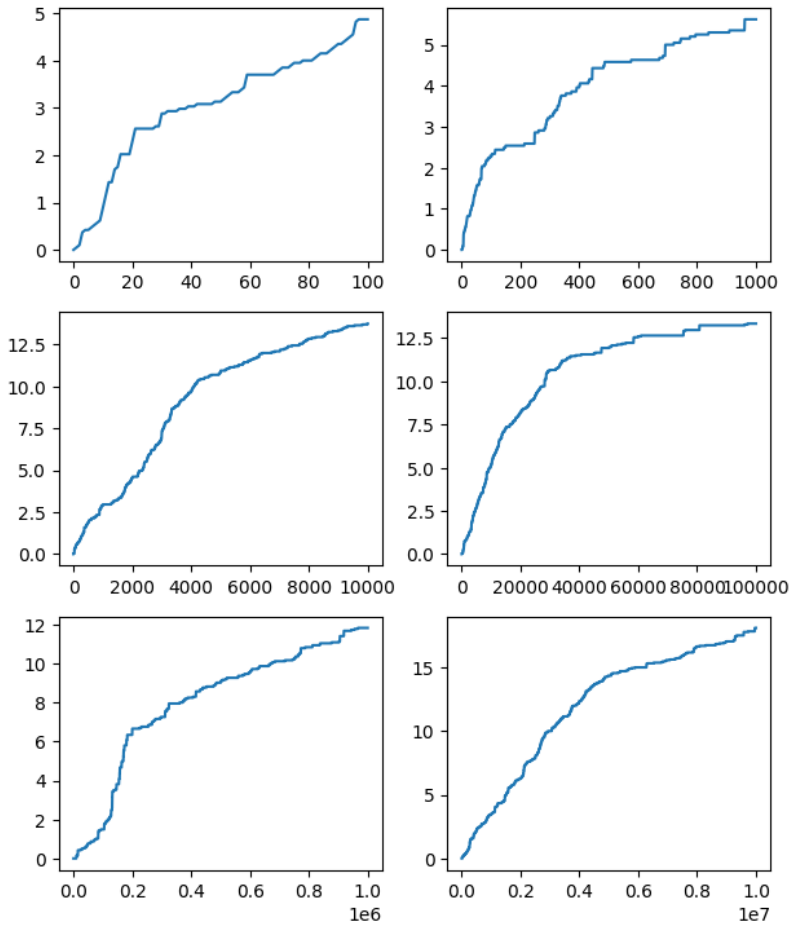


Figure 3.6: Plots of cumulative regret against turn count of 6 games

Randomisation plays a fundamental role in Thompson Sampling, since the choice made on each step is affected by the random sample drawn from the posterior distribution of each arm. It is natural to ask whether it is possible to adopt the same strategy in some deterministic way in the sense of UCB algorithms.

In fact, Thompson Sampling is a stationary randomised strategy. It is stationary in the sense that the choice made only depends on the posterior distribution and is otherwise independent of the time elapsed and randomised as in the sense explained above. [9] The question we wish to ask is whether a stationary deterministic strategy can perform similarly well. The greedy Bayesian algorithm is an example of this.

The answer is no. The exploration factor in Thompson Sampling comes from its randomness, and there are examples where stationary deterministic strategies will give an expected cumulative regret which grows linearly with time, such as the greedy method discussed in the previous section.

Example 3.3.1. [11] Consider the scenario where there are only two machines, A and B. A is known to give rewards with a $Bern(0.5)$ distribution, while B is known to give rewards with a $Bern(\theta)$ distribution and the distribution of θ is known to be $P(\theta = 0.25) = P(\theta = 0.75) = 0.5$. Suppose we have played machine B once and received a reward $r = 0$. By noting that

$$P(\theta = 0.25|r = 0) = \frac{P(r = 0|\theta = 0.25)P(\theta = 0.25)}{P(r = 0)} = 0.75$$

, the expected reward of machine B under the posterior distribution will be

$$\begin{aligned} & E(\theta|r = 0) \\ =& 0.25 \times 0.75 + 0.75 \times 0.25 \\ =& 0.375 \\ <& 0.5 \\ =& \text{expected reward of machine A} \end{aligned}$$

In other words, if we play machine B once and observe $r = 0$, which happens with positive probability, we will play machine A in the next turn. However, since we already know the reward distribution of machine A, we gain no new information from playing machine A and will keep on playing machine A in all future turns. With positive probability, machine B is the optimal machine with distribution $Bern(0.75)$. In this case, each turn beyond this point increases our regret by 0.25, so cumulative regret grows linearly in time. [9]

If we wish to use a deterministic strategy which performs well, we need the strategy to be non-stationary, where the arm selection criterion depends on the current time.

3.4 Randomised Probability Matching

3.4.1 Optimal Arm Probability

Randomised probability matching (RPM) is another Bayesian approach in which the conditional probability of an arm a being the optimal arm a^*_{t+1} at turn $t+1$

given observed rewards is computed for each arm. Formally, this probability is

$$p_{a,t+1} := P(a = a^*_{t+1} \mid \boldsymbol{y}_t) \tag{3.1}$$

This probability is the crux of RPM algorithm because the next arm to be played is decided purely on it. Specifically, each arm a has probability $p_{a,t+1}$ of being played at turn $t+1$. It remains to compute $p_{a,t+1}$.

Proposition 3.4.1.

$$p_{a,t+1} = \int_0^1 I(\theta_a)\pi(\theta_a \mid \boldsymbol{y}_t)\, d\theta_a \text{ , where}$$
$$I(\theta_a) = \begin{cases} 1, & \text{if } \mu(\theta_a) = \max\{\mu(\theta_1), \mu(\theta_2), ..., \mu(\theta_K)\} \\ 0, & \text{otherwise} \end{cases}$$

Proof.

$$\begin{aligned} p_{a,t+1} &= P(a = a^*_{t+1} \mid \boldsymbol{y}_t) \\ &= \int_0^1 P(a = a^*_{t+1} \mid \theta_a, \boldsymbol{y}_t)\pi(\theta_a \mid \boldsymbol{y}_t)\, d\theta_a \\ &= \int_0^1 P\Big(\mu(\theta_a) = \max\{\mu(\theta_1), \mu(\theta_2), ..., \mu(\theta_K)\} \mid \boldsymbol{y}_t\Big)\pi(\theta_a \mid \boldsymbol{y}_t)\, d\theta_a \,, \end{aligned}$$

where $\pi(\theta_a \mid \boldsymbol{y}_t)$ is the pdf of the posterior distribution of θ_a given $\boldsymbol{y}_t$.

Notice the event $\big\{\mu(\theta_a) = \max\{\mu(\theta_1), \mu(\theta_2), ..., \mu(\theta_K)\}\big\}$ for $\boldsymbol{y}_t$ given is deterministic, so the probability of it given $\boldsymbol{y}_t$ is

$$I(\theta_a) = \begin{cases} 1, & \text{if } \mu(\theta_a) = \max\{\mu(\theta_1), \mu(\theta_2), ..., \mu(\theta_K)\} \\ 0, & \text{otherwise} \end{cases}$$

Hence,

$$p_{a,t+1} = \int_0^1 I(\theta_a)\pi(\theta_a \mid \boldsymbol{y}_t)\, d\theta_a \,.$$

□

3.4.2 Monte Carlo Integration

To compute the integral we employ the Monte Carlo sampling method. Intuitively, since we are integrating the function $I(\cdot)$ over the pdf of θ_a given $\boldsymbol{y}_t$, we could draw samples from the posterior distribution of θ_a and apply $I(\cdot)$ to them for an approximation [12]. Mathematically,

$$p_{a,t+1} \approx \hat{p}^{(M)}_{a,t+1} := \frac{1}{M}\sum_{m=1}^{M} I(\theta_a^{(m)}) \,, \tag{3.2}$$

where $\theta_a^{(m)}$ for $m = 1, 2, ..., M$ are drawn from the posterior distribution of θ_a.

Algorithm 5 Randomised Probability Matching Algorithm

Require: Number of arms K, number of Monte Carlo samples M, and horizon T

Require: Prior parameters $(\alpha_{a,1}, \beta_{a,1})$ of each θ_a

for $t = 1, 2, ..., T$ **do**
  for $a = 1, 2, ..., K$ **do**
    Draw M parameter samples ▷ Monte Carlo sampling

$$\theta_a^{(m)} \sim Beta(\alpha_{a,t}, \beta_{a,t})$$

  end for
  for $a = 1, 2, ..., K$ **do**
    $\hat{p}_{a,t+1}^{(M)} = \frac{1}{M} \sum_{m=1}^{M} I(\theta_a^{(m)})$ ▷ Compute optimal arm probability
  end for
  Choose an arm with probability $\hat{p}_{a,t+1}^{(M)}$ for each arm a
  Play chosen arm k
  Observe reward y_t
  Update posterior parameters

$$(\alpha_{k,t+1}, \beta_{k,t+1}) = \begin{cases} (\alpha_{k,t} + 1, \beta_{k,t}), & \text{if } y_t = 1 \\ (\alpha_{k,t}, \beta_{k,t} + 1), & \text{if } y_t = 0 \end{cases}$$

$$(\alpha_{a,t+1}, \beta_{a,t+1}) = (\alpha_{a,t}, \beta_{a,t}) \text{ for } a \neq k$$

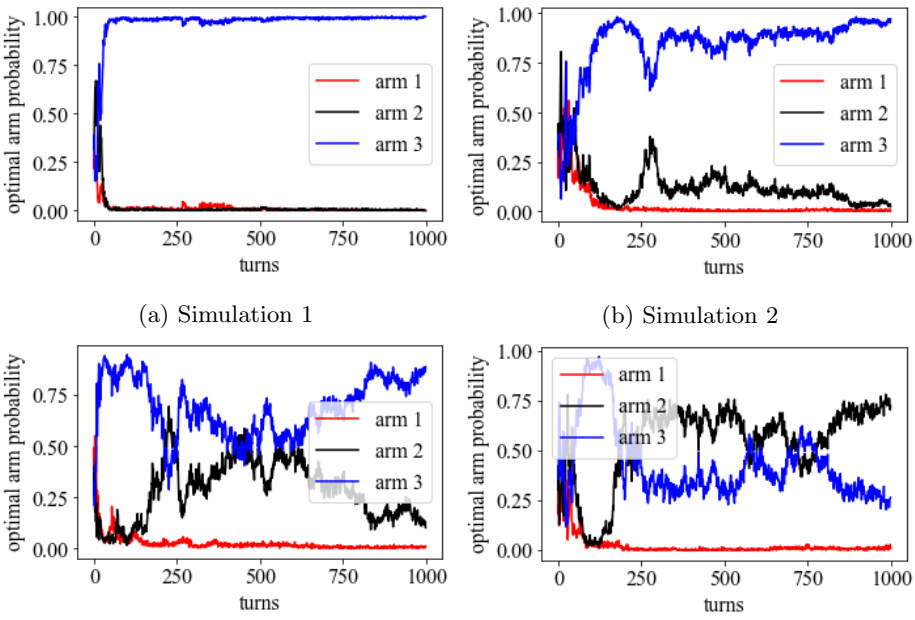
end for

3.4.3 Application

Similar to that in Thompson Sampling, we apply RPM (setting $M = 500$) to the scenario with 3 Bernoulli machines, namely arms 1, 2, and 3, with mean payoff 0.33, 0.55, 0.6 respectively. The prior for the parameters is assumed to follow a $Beta(1, 1)$, or equivalently, $Unif(0, 1)$.

Figure 3.7 plots the optimal arm probability against turns in 4 simulations. Immediately, we see that arm A, which has the lowest mean payoff (0.33), is quickly discarded as an optimal arm in all simulations. However, RPM struggled to decide on whether arm B or C is the optimal arm and understandably so, as their mean payoffs are close to each other.

In figure 3.7a, arm A is quickly selected as the most optimal arm after only around 50 turns, and its optimal arm probability remains high throughout the rest of the turns. That is not always the case. From figure 3.7b we see that around turn 200, the optimal arm probability of arm A peaked but failed to remain high. Here, we experienced the exploration-heavy nature of RPM. Indeed, we can see a similar occurrence in figure 3.7c and 3.7d, with the latter being



(a) Simulation 1

(b) Simulation 2

(c) Simulation 3

(d) Simulation 4

Figure 3.7: Optimal arm probability against turns (arm 1 refers to arm A, 2 refers to arm B, 3 refers to arm C

the most serious offender, not selecting the correct optimal arm even after 1000 turns.

This exploration-heavy nature of RPM is discussed more thoroughly by Urteaga and Wiggins [13]. They devised a double sampling approach which balances between the exploration-heavy RPM approach and the exploitation-heavy greedy approach (choosing the most optimal arm *all* the time). In our example, we see that arm C was selected as the clear, most optimal arm within the first 200 turns in simulation 4, but RPM favoured exploration, and was indecisive about which arm, arm B or arm C, is the most optimal arm even after many turns.

Chapter 4

Comparing Different Strategies

To compare the performance of different strategies, we analyse their realised regret. Recall from the introduction that

$$R(T) = \mu^* \cdot T - \sum_{t=1}^{T} \mu(a_t) \tag{4.1}$$

Table 4.1 shows the set-ups used. Note that the arms in all set-ups follow a Bernoulli distribution. For each set-up, 100 simulations are run using each strategy. The cumulative regret against horizon with various degrees of confidence intervals are plotted in the figures below.

Set-up	Number of arms	Mean payout of each arm
1	3	0.01, 0.01, 0.02
2	3	0.1, 0.1, 0.9
3	2	0.5, 0.52
4	10	0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65

Table 4.1: Set-ups used

In the following section, we walk through each strategy and comment on some of their interesting behaviours in different scenarios. The uniform prior is adopted for Bayesian strategies. 100 Monte Carlo samples are used for RPM; Greedy Bayesian explores with probability 0.02 and uses the 95th percentile for the upper confidence bound; UCB-tuned refers to the improved version of UCB1 as presented in the final part of the corresponding section.

UCB performs poorly in set-up 1 when the mean payout of each arm is very low (close to zero), as seen from figure 4.2. This is because the exploitation term is small and negligible compared to the exploration term. However, this is not to be confused with when the mean payout of each arm is *close* to each other (but not low). In fact UCB outperforms other strategies noticeably in this case, as seen from figure 4.9. It is also worth noting that in set-ups with less arms, UCB has a narrow confidence interval compared to others due to it being the only deterministic algorithm (hence less randomness). One final thing to note is that while most of the regret curves of the algorithms presented here are mainly smooth, there is one particular jump which can be observed for set-up 2 at around the 1000th turn. This sort of plateauing then having a jump is not uncommon in UCB algorithms and is because of the deterministic nature of the algorithm. The plateauing represents picking the correct arm repeatedly. At some point, it will be beneficial for the algorithms to switch to exploring some other arm, which corresponds to the jump.

Greedy Bayesian displays quite a peculiar behaviour. In most cases, its performance is comparable to and at times even better than those of other algorithms. However, on set-up 2, it seems like regret scales linearly with time. Because of the large difference between the true means (0.9 and 0.1), it should be easy to find the optimal arm. Hence, the overall regret should be small for efficient algorithms. However, with fixed probability (0.02 in this case), the algorithm explores one of the arm which it thinks is non-optimal, which dominates the overall regret. In other cases, it is less easy to find the optimal arm, so there is more regret from other sources, and this linear factor is not too significant.

The cumulative regret of Thompson sampling and RPM are close to each other in all set-ups. Although they are never clear "winners" in any of the set-ups, their behaviours are very consistent across the wide range of scenarios presented here. It is advisable to use Thompson sampling and RPM when there is no prior knowledge about the mean payouts, e.g. if the mean payout of each arm is very low (set-up 1), very close to each other (set-up 4), or if there is a clear most optimal arm (set-up 2).

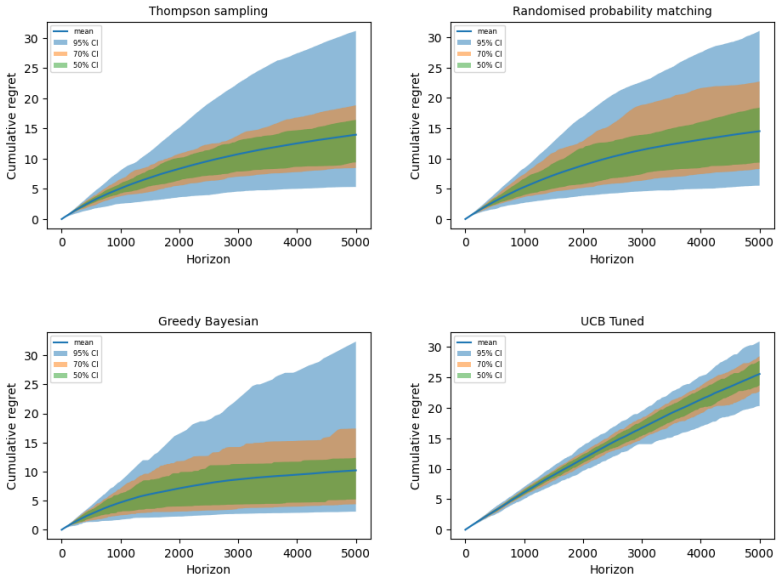


Figure 4.1: Set-up 1: Cumulative regret against horizon using different strategies; separate plots

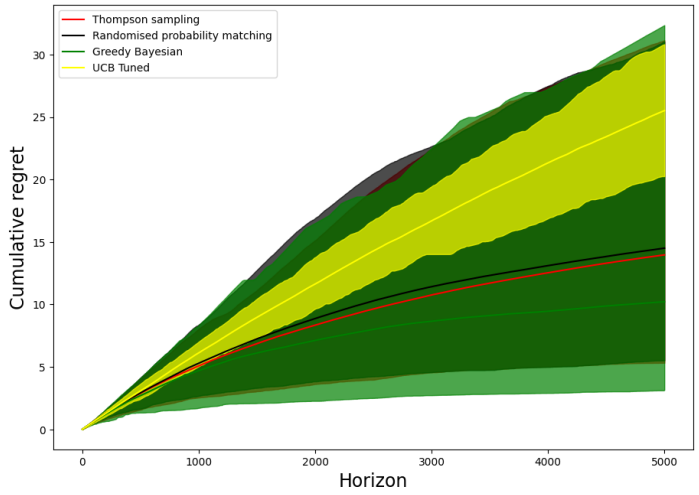


Figure 4.2: Set-up 1: Averaged cumulative regret against horizon using different strategies with 95% confidence interval; single plot

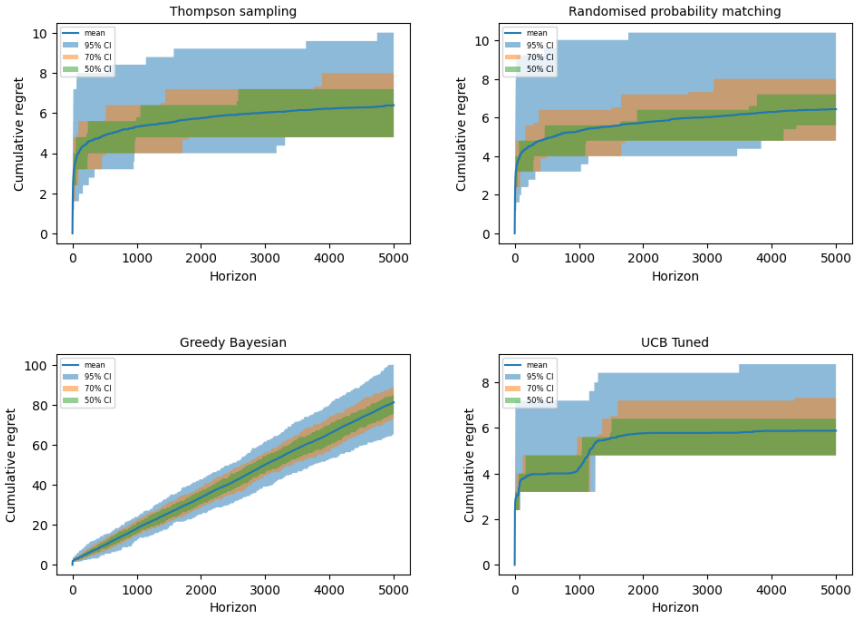


Figure 4.3: Set-up 2: Cumulative regret against horizon using different strategies; separate plots

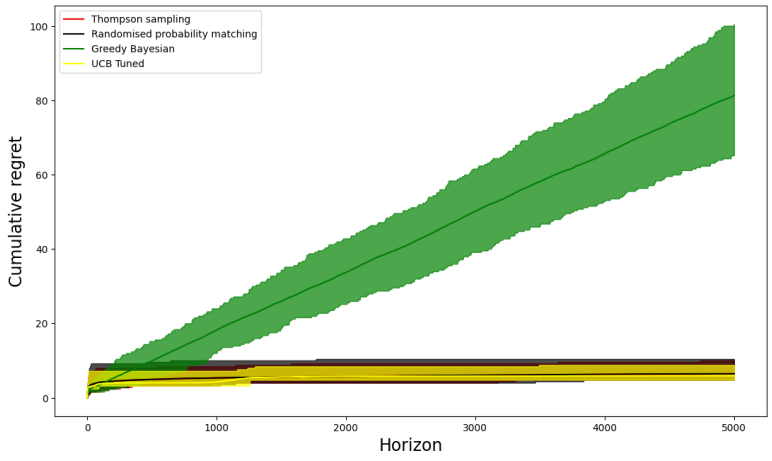


Figure 4.4: Set-up 2: Averaged cumulative regret against horizon using different strategies with 95% confidence interval; single plot

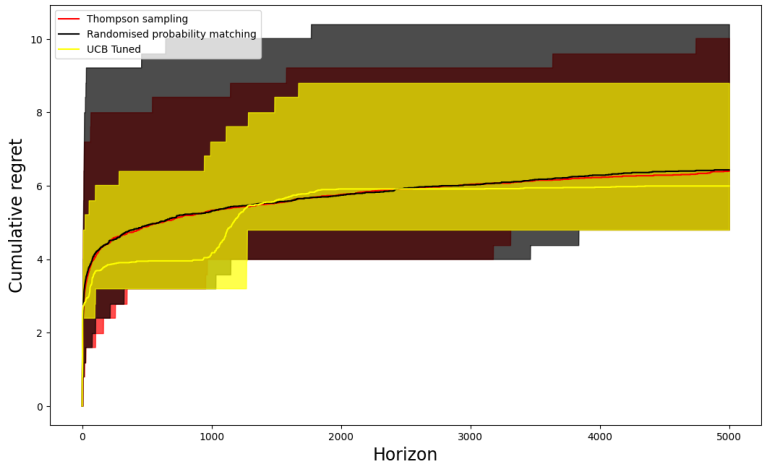


Figure 4.5: Set-up 2: Averaged cumulative regret against horizon using different strategies with 95% confidence interval; without greedy Bayesian; single plot

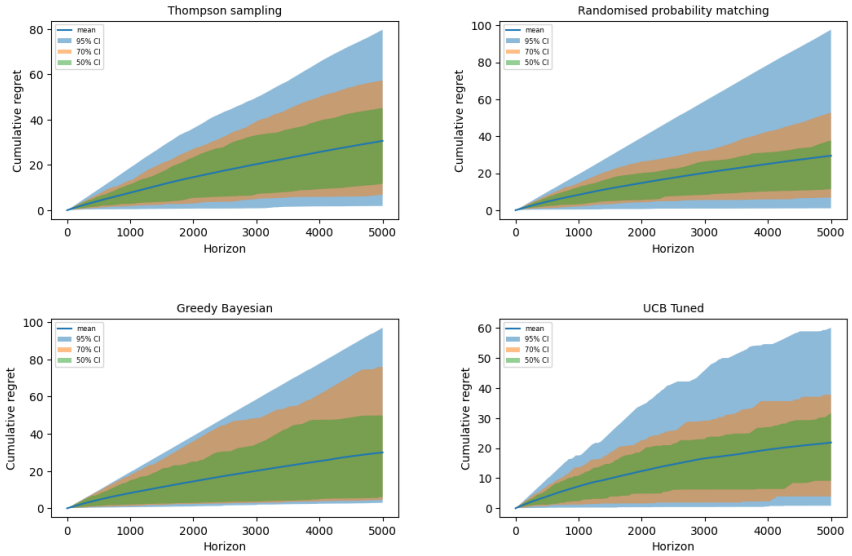


Figure 4.6: Set-up 3: Cumulative regret against horizon using different strategies; separate plots

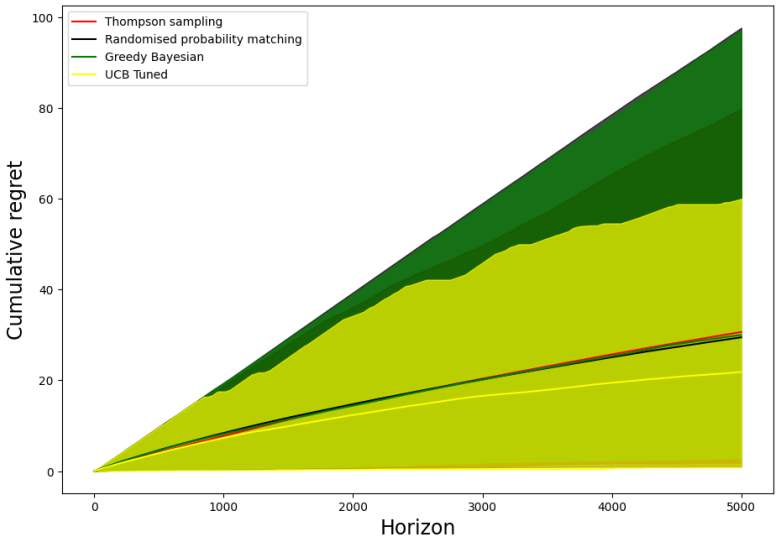


Figure 4.7: Set-up 3: Averaged cumulative regret against horizon using different strategies with 95% confidence interval; single plot

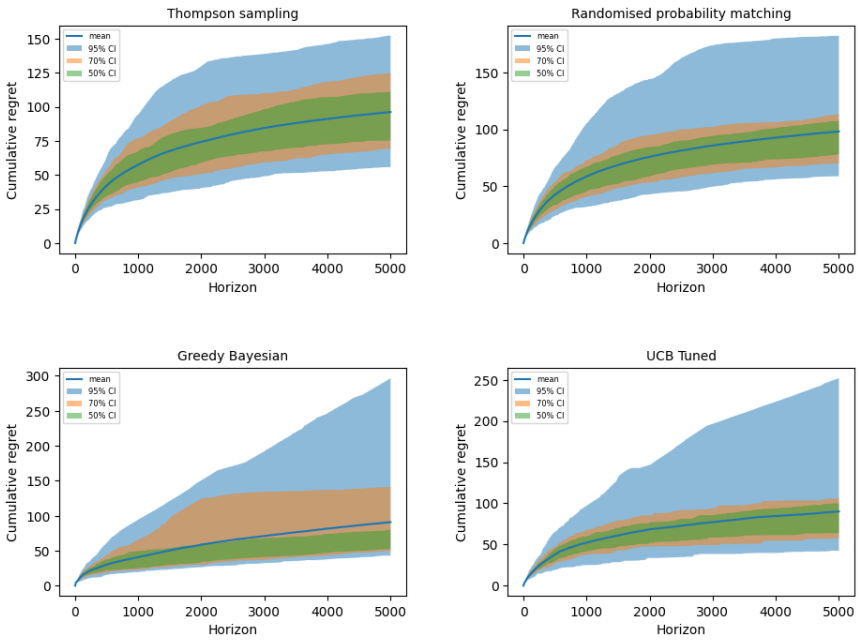


Figure 4.8: Set-up 4: Cumulative regret against horizon using different strategies; separate plots

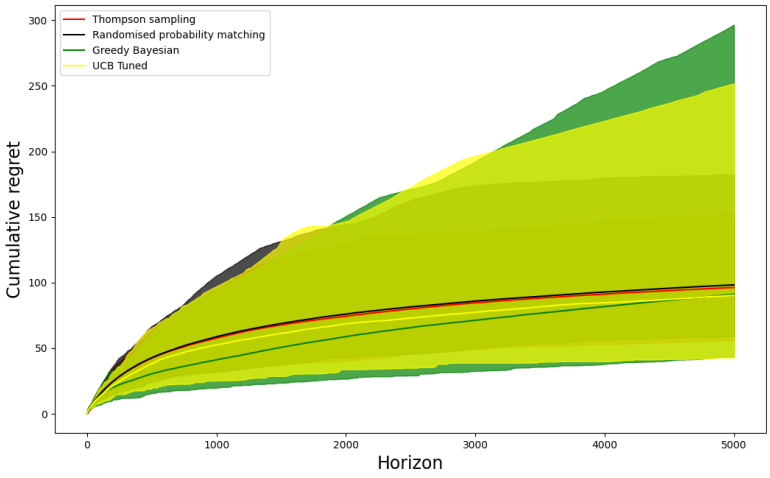


Figure 4.9: Set-up 4: Averaged cumulative regret against horizon using different strategies with 95% confidence interval; single plot

Chapter 5

Non-Stochastic Gambling

5.1 Exp3 Algorithm

Instead of playing against machines whose distributions remain fixed, we now consider playing against an adversary; a player who at each round of the game decides the rewards of each of the K machines, and then we in turn choose a machine, and receive the reward associated with it. In some cases the adversary may be oblivious in which case they will not adjust their output based off our previous actions, but in other scenarios they may do. When playing against a non-oblivious adversary, any deterministic player will do very poorly as the 'good' rewards can be distributed to the arms that are not being played, meaning that algorithms similar to 'playing a machine the next turn if you receive a high reward, otherwise playing the next machine' are useless in these games.

For the previous algorithms we have assumed that the machines follow a stochastic process; the reward distribution for each machine is independent of the time at which the machine is played. However, when we take out this assumption the previous algorithms become redundant, and they can no longer optimally provide a low regret. It is necessary for us to consider the situations without this assumption as in general it is difficult or impossible to correctly determine the true assumptions about the problem.

An example of where we cannot make the stochastic assumption is the problem of choosing a route for transmitting data packets between points in a network. In this case there are K possible routes, and the cost of transmission is reported back to the sender; we would like to minimise the total cost of transmitting a large data set. Unlike the gambler in a casino example, for this example we cannot use stationary distributions to model the costs of each route, as it is likely that the cost is changing with each transmission.

Here we provide a different algorithm, Exp3 [14], to provide a lesser regret

in these scenarios. Exp3 is based off of Hedge, an algorithm designed as a solution to the full information version of the game, where after each round the player receives the full spectrum of rewards from the machine, unlike in our games where we only know the reward from the machine we select.

Algorithm 6 Exp3

Require: Number of arms K, horizon T, prior parameters $(\eta) > 0, \gamma \in (0, 1]$
Set $G_i(0) = 0$ for $i = 1, ..., K$
for $t = 1, 2, ..., T$ **do**
  for $i = 1, 2, ..., K$ **do**
    $$p_i(t) = \frac{exp(\eta G_i(t-1))}{\sum_{j=1}^{K} exp(\eta G_j(t-1))}$$
    $$\hat{p}_j(t) = (1-\gamma)p_j(t) + \frac{\gamma}{K}$$
  end for
  Select action i_t, from the $\hat{p}_i(t)$ distribution
  Receive reward $x_{i_t}(t)$
  Set $G_{i_t}(t) = \frac{x_{i_t}(t)}{\hat{p}_{i_t(t)}}$, and $G_i(t) = G_i(t-1)$ otherwise
end for

5.2 Application

In our simulations we are assuming that the reward from each bandit is bounded, on the interval $[0, 1]$. In this scenario we then can form an upper bound for the expected regret of Exp3; that being:

$$R_{Exp3} \leq 2\sqrt{e-1}\sqrt{gKlnK}$$

where

$$g > E(G_{max}), \eta = \frac{\gamma}{K}, \gamma = min\{1, \sqrt{\frac{KlnK}{(e-1)g}}\}$$

Here $E(G_{max})$ is the maximum value of the expected total reward of playing any machine repeatedly, and hence we know $E(G_{max}) \leq T$.

To demonstrate the algorithm, we are running simulations of 20 Bernoulli machines, each with a random probability of success drawn from the $Unif(0, 1)$ distribution for 2000 simulations. Our graphs show 5000 turns because this better indicates the long term behaviour of Exp3.

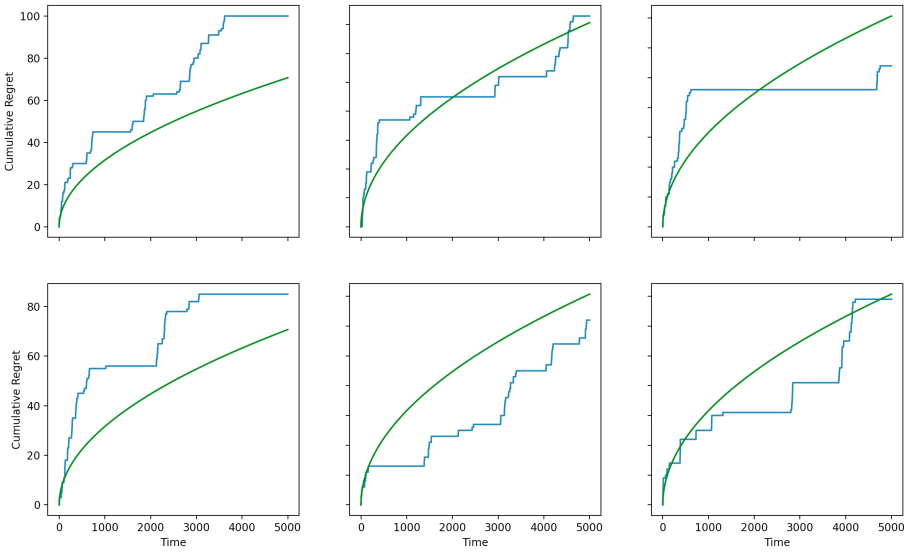


Figure 5.1: Exp3 simulations (Blue)

From the simulations, we can see that the regret approximately follows the square root curve and it is useful to note that this result is much less than the expected upper bound of this algorithm. After having run 1000 simulations, we obtain a mean regret of 50.47, with a 95 percent confidence interval of $(49.25, 51.69)$. While we expect the regret to follow the square root curve, possible reasons that the mean regret is higher than expected is because the number of turns is too low for the number of machines we are playing, meaning there is not enough exploitation of a potentially very good machine.

To see the benefits of Exp3 it is useful to consider the following scenarios. In each scenario, there are 10 biased coins in front of you, 9 of which have low probability of landing on heads, and the last coin which has a high probability of landing on heads. You then earn a £1 reward each time you land a coin on heads, and you have 2000 turns. In scenario 1, this is the entire game; however in scenario 2, unbeknownst to you, at the 250th flip an adversary changes the order of the coins so the most favourable coin is no longer in last position. We demonstrate simulations of each scenario now:

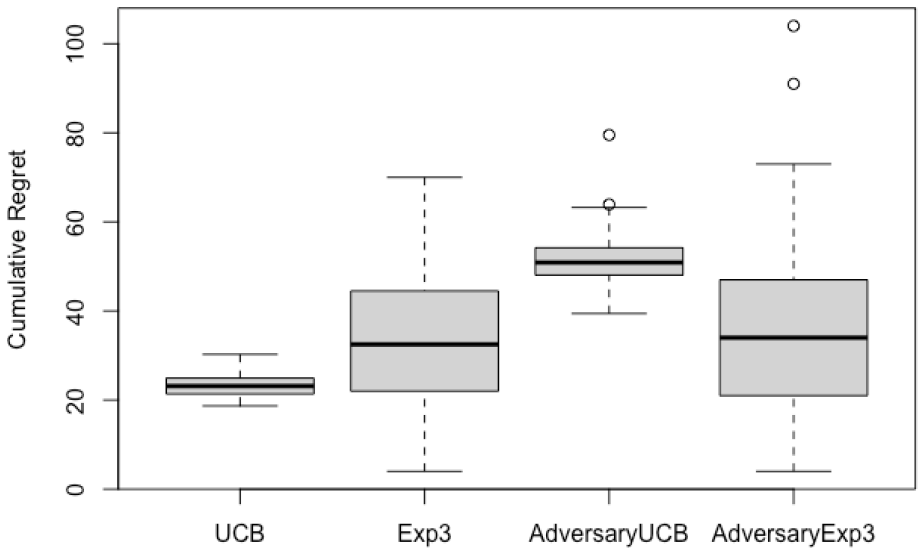


Figure 5.2: Scenario 1 and 2 Results

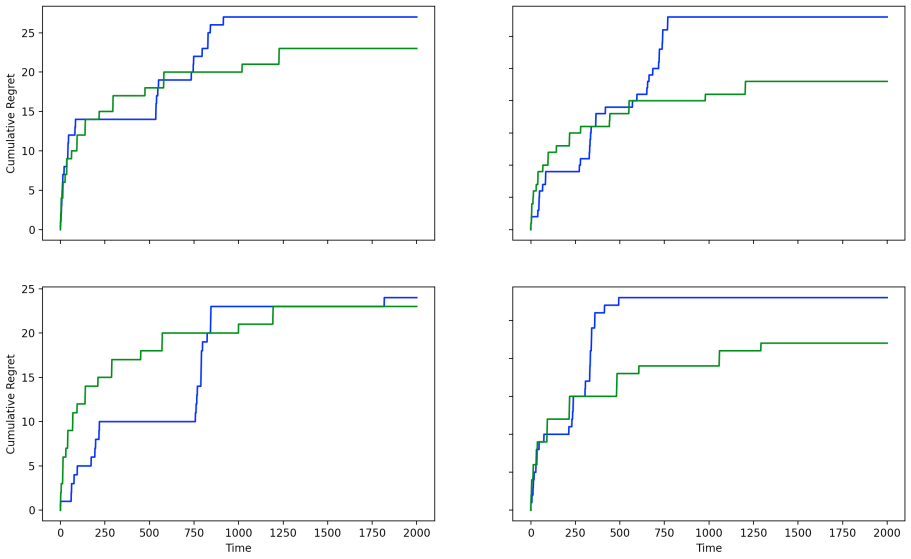


Figure 5.3: Scenario 1 simulations, Exp3 results blue, UCB results green

In scenario 1 we find that the mean regret of Exp3 is greater than that of the UCB algorithm, as we would expect. We obtain a 0.975 confidence interval of $(30.23, 37.31)$ for the Exp3 algorithm, and an interval of $(22.59, 23.67)$ for the UCB algorithm. While Exp3 itself is not massively under-performing in

these simulations, we would naturally assume that the mean of UCB would be less than Exp3 due to UCB benefitting from assumptions that are true in this scenario. By using the Bonferri correction, we can obtain a 0.95 confidence interval that the vector $(E(R_{Exp3}), E(R_{UCB}))$ lies within the region ((30.23, 37.31), (22.59, 23.67)), and hence as there is no overlap or intervals, We can say at a level of 0.95 that $E(R_{Exp3}) > E(R_{UCB})$.

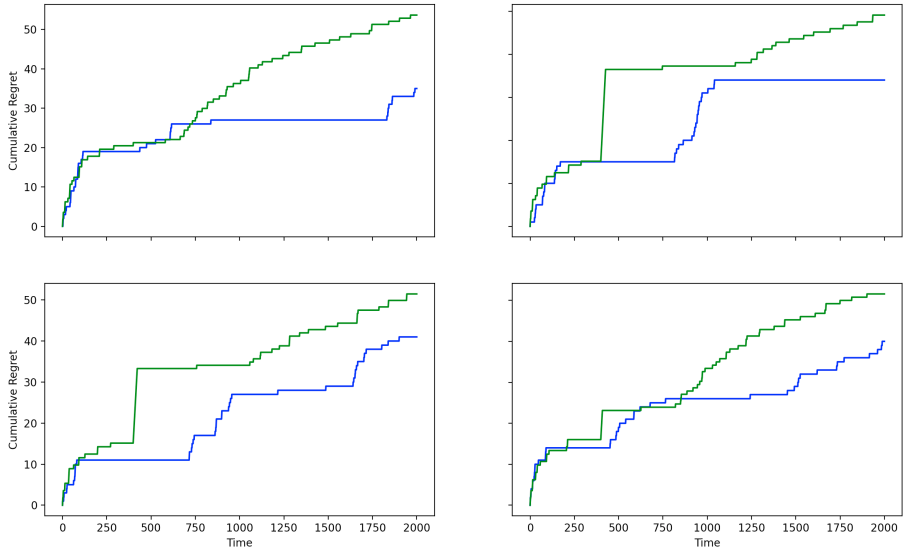


Figure 5.4: Scenario 2 simulations, Exp3 results blue, UCB results green

We compare this result to that of scenario 2. Here Exp3 quickly reacts to the changed version of the game, while on the other hand UCB does not and eventually after enough turns of continuing to play the last coin, the regret builds up massively. Having run simulations of this game, the mean regret of Exp3 is 35.65, while the mean regret of UCB is 51.46. Similarly, we can construct confidence intervals for both the UCB and the Exp3 algorithms, and so we obtain a 0.975 confidence interval that the regret of Exp3 lies in the interval $(31.42, 39.88)$, and a 0.975 confidence interval that the regret of UCB lies in the interval $(50.22, 52.69)$. As there is no overlap between these regions, we can use the bonferri correction to obtain a 0.95 confidence level that the expected regret of running Exp3 in this game is less than the expected regret of running UCB; showing that in some situations it is beneficial to use Exp3 over algorithms designed for stochastic environments.

We should note the expected regret of running UCB has more than doubled in scenario 2 compared to scenario 1, while Exp3 has increased by only a factor of 1.05. Despite the naturally higher spread of results when using Exp3, for games against an adversary as opposed to a machine, it seems necessary to run Exp3

instead of UCB, and other algorithms designed for stochastic environments.

Chapter 6

Continuum-armed Bandits

6.1 Lipschitz Bandits and Zooming Algorithm

In this section we delve into some extensions of the problem we have been dealing with so far. In particular, we focus on the similarity between arms by introducing *Lipschitz Bandits*, a prominent version that studies large, structured action spaces such as the $[0, 1]$ interval. For example, actions can correspond to products with feature vectors, the similarity between two products can be quantified by the distance between two vectors using p-norm, and similar actions are guaranteed to have similar rewards[15].

Definition. Assume we have a action set X, with an unknown mean reward function $u : X \to [0, 1]$ satisfying:

$$|\mu(x) - \mu(y)| \leq \delta(x, y), \forall x, y \in X \text{ and } \delta(x, y) > 0 \tag{6.1}$$

Now, define $D(x, y)$ to be $\inf \sum_t \delta(x_t, x_{t+1})$ over all finite paths $(x = x_0, x_1, ..., x_k = y)$ in X, which is a metric. Therefore, we can rewrite the Lipschitz inequality formula as:

$$|\mu(x) - \mu(y)| \leq D(x, y), \forall x, y \in X \tag{6.2}$$

μ is a Lipschitz function of Lipschitz constant 1 on the metric space (X, D).

From now on, we denote (X, D, μ) as an instance of the Lipschitz bandit problem. [16]

Since for simplicity we set the reward to be either 1 or 0, without loss of generality the diameter of (X, D) is at most 1.

Definition. For any fixed subset $S \subset X$ which can be either discrete or uniform, it is called a discretisation of X.

In the Lipschitz bandits problem, it is quite common to use the discretisation set S as an approximation for X, and it is natural to consider the error of such approximation.

Definition. Denote the optimal reward in S by $\mu^*(S) = \sup_{x \in S} \mu(x)$. The discretisation error is:

$$DE(S) = \mu^*(X) - \mu^*(S) \tag{6.3}$$

By definition, observe that the discretisation error of S is at most the minimal distance between S and the best arm x^* [4]:

$$DE(S) \leq D(S, x^*) := \min_{x \in S} D(x, x^*) \tag{6.4}$$

Definition. Thinking of arms in S as 'probes' that the algorithm places in the metric space. An algorithm could approximately learn the mean rewards overtime, and adjust the placement of the probes accordingly, making sure that one has more probes in more 'promising' regions of the metric space. This approach is called adaptive discretization [4].

One implementation of this approach we introduce in this paper is **zooming algorithm**. The algorithm maintains a set $S \subset X$ of 'active arms'. In each round, some arms may be 'activated' according to the *activation rule*, and one active arm is selected according to the *selection rule*. Once an arm is activated, it can not be 'deactivated' [4].

Confidence ball.: Fix round t and an arm x that is active at this round. T is the time horizon. Let $n_t(x)$ be the number of rounds arm x has been played before round t. Let $\mu_t(x)$ be the average reward in these rounds, and $\mu_t(x) = 0$ if $n_t(x) = 0$.

Definition. Recall the time horizon is T. Then confidence radius of arm x at time t is defined as:

$$r_t(x) = \sqrt{\frac{2logT}{1 + n_t(x)}} \tag{6.5}$$

The meaning of the confidence radius is that with high probability, it bounds from above the deviation of $\mu_t(x)$ from its expectation $\mu(x)$ for all times t and arms x:

$$\text{w.h.p } |\mu_t(x) - \mu(x)| \leq r_t(x)$$

Definition. Confidence ball of arm x at time t: $B(x, r_t(x)) = \{y \in X : D(x, y) \leq r_t(x)\}$.

We consider the samples from arm x available at time t allow us to estimate $\mu(x)$ only up to $\pm r_t(x)$. The available samples from x do not provide enough confidence to distinguish x from any other arm in the the confidence ball [16].

Activation rule. Say that an arm is *covered* at time t if it is contained in the confidence ball of some active arm. Suppose arm y is not active in round t but it is covered in the confidence ball of x, such that $D(x, y) \ll r_t(x)$. The algorithm does not have enough samples of x to distinguish x and y. Thus, there is no need to activate y yet. Instead, we choose x still. However, we do need to maintain the following invariant:

In each round, all arms are covered by confidence balls of the active arms.

The confidence radius gets smaller as the turns go on, and some arm y may become uncovered. Now it is the time to activate any such uncovered arm. Note that the confidence radius of the newly activated arm is initially greater than 1, so all arms are trivially covered [16].

With this activation rule, the zooming algorithm has the 'self-adjusting property'. The algorithm 'zooms in' on a given region R of the metric space if and only if the arms in R are played often if and only if the arms in R have high mean rewards [4].

Selection rule. Play an active arm with the largest index $I_t(x)$ which is defined as follow:

$$I_t(x) = \mu_t(x) + 2r_t(x) \tag{6.6}$$

The intuition behind the index is that if $I_t(x)$ is large, then either $\mu_t(x)$ is large, implying x is likely to be a good arm, or $r_t(x)$ is large, implying arm x has not been played very often, and should be explored more [4].

Algorithm 7 Zooming Algorithm[4]

Initialize:
    set of active arms $S \leftarrow \emptyset$
for t = 1, 2, ... **do**
    // activation rule
    if there is any arm y not covered by the confidence balls of active arms **then**
        pick any such arm y and 'activate' it: $S \leftarrow S \cap \{B(y, r_t(y))\}$
    end if
    // selection rule
    play an active arm x with largest index $I_t(x)$
end for

6.2 Analysis and Regret

Definition. For each arm x, the clean event is $\mathcal{E}_x = \{|\mu_t(x) - \mu(x)| \leq r_t(x)\}$ for all rounds $t \in [T+1]$ where $[T] := \{1, 2, ..., T\}$[4].

If arm x has not been played yet, define $\mu_t(x) = 0$ for convenience. We are primarily interested in the event $\mathcal{E} = \cap_{x \in X} \mathcal{E}_x$.

Claim 1. Assume that realized rewards take values on a finite set. Then $Pr[\mathcal{E}] \geq 1 - \frac{1}{T^2}$[4].

Proof. Fix an instance of Lipschitz bandits. Let X_0 be the set of all arms that can possibly be activated by the algorithm on this problem instance. Under the

assumption of finite realised rewards, we know the algorithm is deterministic, and the time horizon is fixed, then X_0 is finite.

Let N be the total number of arms activated by the algorithm. Define arms $y_j \in X_0$, $j \in [T]$ as follows:

$$y_j = \begin{cases} \text{j-th arm activated} & \text{if j} \leq N \\ \text{y}_N & \text{otherwise} \end{cases}$$

Note that $\{y_1, ..., y_T\}$ is precisely the set of arms activated in a given execution of the algorithm. Since the clean event holds trivially for all arms that are not activated, the clean event can be rewritten as $\mathcal{E} = \cap_{j=1}^{T} \mathcal{E}_{y_j}$. Now we only need to prove the clean event $\mathcal{E}_{y_j}$ happens w.h.p for each $j \in [T]$.

Fix an arm $x \in X_0$ and fix $j \in [T]$. And the events $\{y_j = x\}$ and $\mathcal{E}_x$ are independent. Therefore, if $Pr[j_i = x] > 0$, then:

$$Pr[\mathcal{E}_{y_j} | y_j = x] = Pr[\mathcal{E}_x | y_j = x] = Pr[\mathcal{E}_x] \geq 1 - \frac{1}{T^4}$$

$$Pr[\mathcal{E}_{y_j}] = \sum_{x \in X_0} Pr[y_j = x] \cdot Pr[\mathcal{E}_{y_j} | y_j = x] \geq 1 - \frac{1}{T^4}$$

Applying the Union bound over all $j \in [T]$, we get:

$$Pr[\mathcal{E}_{y_j}, j \in [T]] \geq 1 - \frac{1}{T^3}$$

□

Define the reward gap between arm x and the best arm as $\Delta(x) = \mu^* - \mu(x)$ and let $n(x) = n_{T+1}(x)$ be the total number of samples from arm x.

Lemma 6.2.0.1. $\Delta(x) \leq 3r_t(x)$ for each arm x and each round t.

Proof. By the covering invariant, the optimal arm x^* is covered by the confidence ball of some arm y i.e. $x^* \in B(y, r_t(y))$. It follows that:

$$I_t(x) \geq I_t(y) = \mu_t(y) + r_t(y) + r_t(y) \geq \mu(x^*) = \mu^*$$

by Lipschitz condition. On the other hand:

$$I_t(x) = \mu_t(x) + 2r_t(x) \leq \mu(x) + 3r_t(x)$$

Putting these two equations together: $\Delta(x) := \mu^* - \mu(x) \leq 3r_t(x)$ [4] □

Corollary 6.2.0.1. For any two active arms x, y, $D(x, y) > \frac{1}{3} \min(\Delta(x), \Delta(y))$.

Corollary 6.2.0.2. For each arm x, $n(x) \leq \frac{O(logT)}{\Delta^2(x)}$.

Now we will formulate the regret of the zooming algorithm.

Consider the set of arms whose gap Δ is between r and $2r$, for $r > 0$: $X_r = \{x \in X : r < \Delta(x) < 2r\}$. Fix $i \in N$ and let $Y_i = X_r$ where $r = 2^{-i}$.

By corollary 6.2.0.1, we know that if we cover Y_i with subsets of diameter $\frac{r}{3}$, then arms x and y cannot lie in the same subset since $D(x, y) > \frac{r}{3}$ [4].

Define $N_{\frac{r}{3}}(Y_i)$ as the number of sets of diameter less than $\frac{r}{3}$ sufficient to cover such $Y_i \implies |Y_i| \leq N_{\frac{r}{3}}(Y_i)$.

Definition. Zooming dimension d: The smallest non-negative number such that $N_{\frac{r}{3}}(Y_i) \leq c \cdot r^{-d}$, $\forall r > 0$, for some constant positive multiplier c.

Following corollary 6.2.0.2, we have

$$R_i(T) = \sum_{x \in Y_i} \Delta(x) \cdot n_t(x) \leq \frac{O(logT)}{\Delta(x)} \cdot N_{\frac{r}{3}}(Y_i) \leq \frac{O(logT)}{r} \cdot N_{\frac{r}{3}}(Y_i)$$

Pick $\delta > 0$, and consider arms with $\Delta(\cdot) \leq \delta$ separately from those with $\Delta(\cdot) \geq \delta$, therefore: [4]

$$\begin{aligned} R(T) &\leq \delta \cdot T + \sum_{i: r = 2^{-i} > \delta} R_i(T) \\ &\leq \delta \cdot T + \sum_{i: r = 2^{-i} > \delta} \frac{\Theta(logT)}{r} \cdot N_{\frac{r}{3}}(Y_i) \\ &\leq \delta \cdot T + O(c \cdot logT) \cdot (\frac{1}{\delta})^{d+1} \end{aligned} \tag{6.7}$$

Choose $\delta = (\frac{logT}{T})^{\frac{1}{d+2}} \implies R(T) \leq O(T^{\frac{d+1}{d+2}} \cdot (clogT)^{\frac{1}{d+2}})$, and we are ready to reach our final conclusion:

Theorem 6.2.1. *Consider Lipschitz bandits with time horizon T. Assume that realised rewards take values on a finite set. For any given problem instance and any $c > 0$, the zooming algorithm attains regret*

$$E(R(T)) \leq O(T^{\frac{d+1}{d+2}} \cdot (clogT)^{\frac{1}{d+2}}) \tag{6.8}$$

where d is the zooming dimension with multiplier c [4].

Here is a simulation of zooming algorithm generated by referencing [17].

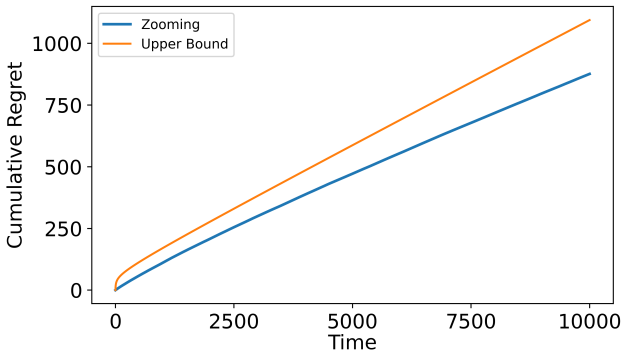


Figure 6.1: Simulation of zooming algorithm and upper bound

We have chosen $c = 1$ and zooming dimension d to be 0.01. Time horizon T is set to be $10,000$ and we perform 500 trials. Let $\delta = (\frac{logT}{T})^{\frac{1}{d+2}} \approx 0.03$, and we can see our regret is indeed well bounded by our theoretical upper bound.

Bibliography

[1] Zhang J. Reinforcement Learning — Multi-Arm Bandit Implementation. Towards Data Science Inc.; 2019. https://towardsdatascience.com/reinforcement-learning-multi-arm-bandit-implementation-5399ef67b24b.

[2] Mahajan A, Teneketzis D. Multi-armed bandit problems. Foundations and Applications of Sensor Management. 2008:121–151.

[3] Kaufmann E, Cappe O, Garivier A. On bayesian upper confidence bounds for bandit problems. PMLR; 2012. Available from: http://proceedings.mlr.press/v22/kaufmann12.html.

[4] Slivkins A. Introduction to multi-armed bandits; 2022. Available from: https://arxiv.org/abs/1904.07272.

[5] Auer P, Cesa-Bianchi N, Fisher P. Finite Time Analysis of the Multi-Armed Bandit Problem. Kluwer Academic Publishers; 2002. https://doi.org/10.1023/A:1013689704352.

[6] ;.

[7] Ray K. Statistical theory; 2021. [Lecture], Imperial College London.

[8] Lauritzen S. Sequential Bayesian Updating; 2009. [Lecture], University of Oxford. Available from: https://www.stats.ox.ac.uk/~steffen/teaching/bs2HT9/kalman.pdf.

[9] Russo D, Roy BV, Kazerouni A, Osband I, Wen Z. A Tutorial on Thompson Sampling. Foundations and Trends in Machine Learning. 2020. Available from: https://arxiv.org/abs/1707.02038.

[10] Agrawal S, Goyal N. Analysis of Thompson Sampling for the Multi-armed Bandit Problem. JMLR. 2012;23. Available from: http://proceedings.mlr.press/v23/agrawal12/agrawal12.pdf.

[11] Russo D, Roy BV. Learning to Optimize via Information-Directed Sampling. Operations Research. 2018. Available from: https://arxiv.org/abs/1403.5556.

[12] Weinzierl S. Introduction to Monte Carlo methods. arXiv; 2000. Available from: https://arxiv.org/abs/hep-ph/0006269.

[13] Urteaga I, Wiggins C. Bayesian bandits: balancing the exploration-exploitation tradeoff via double sampling. 2017 09.

[14] Auer P, Cesa-Bianchi N, Freund Y, E Schapire R. Gambling in a rigged casino: The adversarial multi-armed bandit problem; 1998. Available from: https://www.researchgate.net/publication/2265004_Gambling_in_a_rigged_casino_The_adversarial_multi-armed_bandit_problem.

[15] Podimata C, Slivkins A. Adaptive discretization for adversarial Lipschitz Bandits; 2021. Available from: https://arxiv.org/abs/2006.12367.

[16] Kleinberg R, Slivkins A, Upfal E. Bandits and experts in metric spaces; 2019. Available from: https://arxiv.org/abs/1312.1277.

[17] Lu S. Lipschitiz-bandit-experiment. GitHub; 2019. https://github.com/runninglsy/Lipschitz-bandits-experiment.

Chapter 7

Appendix

7.1 GitHub Repository

To access the Python package we have created to run simulations and conduct analysis, please see the following GitHub repository:

`https://github.com/n88k/multi-armed-bandit`