DECISION MAKING USING THOMPSON SAMPLING

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2014

By
Joseph Mellor
School of Computer Science

Contents

Word Count: 42687

List of Tables

List of Algorithms

List of Figures

Glossary

$H(j_1, \ldots, j_T)$ Adversarial hardness measure.

$A(\nu(\mathcal{K}))$ Aggression measure of a best arm algorithm with rank distribution $\nu(\mathcal{K})$..

Anytime SR An algorithm similar to Successive Rejects but which does not require the time horizon to be known..

a_t The arm chosen at time t.

$N_{k,T}$ The number of times arm k is pulled up to and including time t.

π_k The distribution from which rewards are drawn for arm k.

Δ_k The gap, $\mu_{\max} - \mu_k$, between the best arm and arm k.

μ_k The mean reward of arm k.

ρ_k The parameters specifying distribution π_k.

$\sigma(k)$ The index of the arm with rank k.

$\mathcal{K}$ The set of arms..

ξ Discount rate.

$\kappa(P(\mu_k))$ The Gittins index of arm k.

Global-CTS Global Changepoint Thompson Sampling..

Global-CTS2 A variant of Global Changepoint Thompson Sampling that uses a heuristic method of setting prior distributions to try and track changes in only the best arm..

$\mathcal{H}_1$ Hardness measure 1 for a best arm identification problem.

$\mathcal{H}_2$ Hardness measure 2 for a best arm identification problem.

h hypothesis.

MBoP Maximum Boundary of Pairs..

P_e Probability of error..

o observation.

OSTS Ordered-Statistic Thompson Sampling..

OSTS-Gap A variant of OSTS where the rank distribution is formed using beliefs of the gaps between arms..

OSTS-Poisson A variant of OSTS where the rank distribution is a Poisson distribution..

OSTS-Zipf A variant of OSTS where the rank distribution is based on a Zipf distribution..

OSTS-Zipf(s) A variant of OSTS where the rank distribution is the Zipf distribution with parameter s..

PA-CTS Per-Arm Changepoint Thompson Sampling..

PA-CTS2 A variant of Per-Arm Changepoint Thompson Sampling that only tracks changes in the best arm..

PF Global-CTS Parameter-free Global Changepoint Thompson Sampling..

PF Global-CTS2 A variant of Parameter-free Global Changepoint Thompson Sampling that uses a heuristic to only track changes in the best arm..

PF PA-CTS Parameter-free Per-arm Changepoint Thompson Sampling..

PF PA-CTS2 A variant of Parameter-free Per-arm Changepoint Thompson Sampling that only tracks changes in the best arm..

$\nu(\mathcal{K})$ Rank distribution over arms $\mathcal{K}$..

o_t Sampled rank of OSTS at time t..

$\Psi(T)$ The recommended final decision after T steps..

$R(T)$ Expected cumulative regret received by agent up to and including time t.

r_t The runlength at time t, which is the length of time since a change-point.

$S(T)$ Expected cumulative reward received by agent up to and including time t.

γ The switch rate, which is the probability of a change occurring.

$x_{k,t}$ Reward received by pulling arm k at time t.

x_t Reward received at time t.

Abstract

Decision making using Thompson Sampling
Joseph Mellor
A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy, 2014

The ability to make decisions is a crucial ability of many autonomous systems. In many scenarios the consequence of a decision is unknown and often stochastic. The same decision may lead to a different outcome every time it is taken. An agent that can learn to make decisions based purely on its past experience needs less tuning and is likely more robust. An agent must often balance between learning the payoff of actions by exploring, and exploiting the knowledge they currently have. The multi-armed bandit problem exhibits such an exploration-exploitation dilemma. Thompson Sampling is a strategy for the problem, first proposed in 1933. In the last several years there has been renewed interest in it, with the emergence of strong empirical and theoretical justification for its use.

This thesis seeks to take advantage of the benefits of Thompson Sampling while applying it to other decision-making models. In doing so we propose different algorithms for these scenarios. Firstly we explore a switching multi-armed bandit problem. In real applications the most appropriate decision to take often changes over time. We show that an agent assuming switching is often robust to many types of changing environment. Secondly we consider the best arm identification problem. Unlike the multi-armed bandit problem, where an agent wants to increase reward over the entire period of decision making, the best arm identification is concerned in increasing the reward gained by a final decision. This thesis argues that both problems can be tackled effectively using Thompson Sampling based approaches and provides empirical evidence to support this claim.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Publications

Joseph Mellor and Jonathan Shapiro. “Thompson Sampling in Switching Environments with Bayesian Online Change Detection”. In: *AISTATS*. Vol. 31. JMLR Proceedings. JMLR.org, 2013, pp. 442–450

For my dad.

Acknowledgements

Firstly, thank you to my supervisor Dr. Jon Shapiro, whose guidance has allowed me to complete this thesis and greatly improved my approach to research. Thank you also to Dr. Mark Muldoon and Dr. Martin Brown whose encouragement, specifically in my first year, was invaluable.

Thank you to my mum, brother and Claire for all the support, especially over the last few years. I'd also like to thank my dad without whose initial encouragement I may never have embarked on this part of my life.

Thanks is also due to Mrs McKenzie, my high school mathematics teacher, who was a huge influence in inspiring my continuing interest in mathematics.

A big thank you is also due to all my friends, old and new, who have made the last few years all the more enjoyable. And lastly thank you Sandbar, I couldn't have done it without you.

Chapter 1

Introduction

The need to make decisions permeates our lives. We are continuously making them in order to affect the world around us. Decisions range from the trivial, like 'what should I have for breakfast?', to the much more serious, like 'how should a power station be controlled safely?'. Many of these decisions are now made on our behalf by automated systems. From automated stock trading systems, to cars that can navigate and drive, there are an increasing number of automated decision-making systems that already, or soon will, have an increasing effect on our lives. With the increased ubiquity of computing devices and sensors, the importance of automating decision making based on this influx of data becomes of greater interest. How should we make systems that can make these decisions for us?

The first attempts at making decision-making "expert" systems concentrated on the use of logic and formal reasoning. Such systems were brittle since feeding the system with sufficient ground truths is difficult and time consuming, and more importantly not all outcomes are entailed by a purely logic system especially when they are probabilistic or uncertain. More recently progress has been made towards more robust systems that take a different approach, which instead *learn* how to make decisions through experience. This thesis is interested in this aspect of decision-making systems, how a decision-making agent should make decisions and interact with an environment, improving their behaviour through experience. The specific decision-making problems with which we are concerned can be explained by way of examples.

Every morning I get up and prepare to travel to the University, leaving my house at roughly the same time. At the end of my road is a bus stop and, a

few minutes further down from that, a tram stop. Each day I have a 9 o'clock lecture and being studious I decide that I do not want to be late. Amongst my options are a couple of different bus routes, the tram or walking. Unfortunately I do not know the most efficient option. I decide that I will try to discover which is the best journey purely through experience, by trying commutes available to me and forming some conclusion on which will get me in on time. I might be tempted to try each of the commutes once and then from my experience of these trips decide which commute is best. Then from there on in stick forever more to a single route. Although this may appeal to some sense of habit or routine, if my aim is to be punctual it is unlikely to be a satisfactory strategy.

What is it about my morning commute problem that makes this so? The key reason is that the arrival time for my journey depends on a variety of factors potentially unknown to me. A bus could get a flat tyre or there could be leaves on the tram line on any given day, and so whether I am late using any particular route can vary from one day to the next. This variation makes my problem *stochastic* in nature. Since the outcome of my decisions are not deterministic I want to take the transport that *in general* gets me in on time most often, as to reduce the number of days I am late. That is, I want to pick the commute with highest expected success rate. Now to get a estimate of the expected success rate of a given route, through just my experience, involves taking the transport a number of times and taking an average. The more a route is taken the more accurate the estimate is likely to be. I could try allocating a fixed amount of time taking each of the transport options, observing which got me in on time most frequently and then choosing this option. If my aim is to be as punctual as possible there are two main problems with this approach. Firstly, I can never be sure my estimates are accurate enough to make a final decision. My initial journeys may end up being largely unrepresentative of future journeys, just down to dumb luck. Secondly, every time I take a (unbeknownst to me) slower-on-average route I make it more likely the number of times I am late increases. However in order to avoid taking a slow route I need to know which routes are slow (conversely which routes are fast). There is therefore a balance, or tradeoff, between choosing what I perceive as the most punctual commute (which involves exploiting the knowledge I have thus far gained), and improving the my estimate of the performance of other commutes in order to reduce my uncertainty of what is the best journey to take (exploring my options). This is known as the exploration-exploitation dilemma.

My morning commute predicament is well characterised by a simple model called the Multi-armed Bandit Problem that captures the above dilemma. Being simple the problem is a good model in which to study the automation of decision making in the presence of uncertainty. I could then use a strategy suited to the multi-armed bandit problem to guide the way in which I chose to travel each morning.

As the statistician George Box observed "all models are wrong, but some are useful". Despite being a useful model, the multi-armed bandit problem is not without its limitations. There are some assumptions made in the model that might degrade the performance of my commuting strategy and hinder me from getting to University on time as often. Factors that affect a journey, such as my bus getting a flat tyre, make the problem stochastic.

Some factors, like a flat tyre, can be seen as independent in two ways. Firstly, one bus getting a flat tyre has little to no bearing on the journey times of the other bus routes or the tram. The actions I choose are independent of each other. This is not necessarily true of all factors. Severe weather is likely to have an impact on all transport simultaneously, and road works on one route may cause more people to use an alternative route, increasing the chance you will have to wait for the next one due to overcrowding.

Secondly, the event of a flat tyre is temporally independent with respect to a given bus route. That is, given I have decided to take the route 86 bus, the chances that the 86 bus gets a flat tyre is independent of whether it got a flat tyre yesterday. The model is temporally invariant and so the probability of getting to work on time follows a stationary statistical law.

However, making this stationarity assumption does not serve my punctuality objective well. Imagine I have happily been following my multi-armed bandit strategy to choose my commutes each morning for some time comforted by the idea that I am being as punctual as I can be. It happens that this means that I more often or not take the route 86 bus. However, unfortunately for me the route follows a road with a major and aging gas pipe running under it. The road must be reduced to a single lane while maintenance and upgrades are made to the gas pipe. This severely detriments the quality of the 86 bus service for a significant amount of time. The number of times I am late by using the 86 bus now soars for the duration of the maintenance. My multi-armed bandit strategy unfortunately is not assuming that such changes occur and so this change in behaviour goes

largely ignored and I do not adapt my decision making quickly enough to the new state of affairs. I begin to get more of a reputation for tardiness. What I should be doing is noticing that the frequency of lateness has dramatically increased and adapt my journey choices to account for this. In my predicament the journey times jump starkly, or *switch* as a result of external factors like road works. I need to be able to account for the switching to ensure I make better decisions. To account for these effects we need to enrich our models. This leads to the consideration of *non-stationary switching multi-armed bandit problems*.

The tram and bus companies offer significantly discounted prices for journeys with them in terms of season tickets. A price is paid upfront for a pass or ticket that lets me travel for an extended period. The above mentioned bandit models may assume I can choose a different journey to the University every day. Again this assumption is not always suitable. I may have competing concerns. I want to arrive in time every day but I also want to take advantage of the low-price season tickets in order to save money. By using a season ticket I prevent myself from exploring other options, so I want to be as confident as possible that I'm choosing the season ticket for the right service. To do this I decide that I will allow myself a set period of time to explore my options before committing myself to a particular journey. The question then becomes how do I choose my initial journeys so that when it comes to buying a season pass I am more probable to choose the right one. This sort of decision-making dilemma again requires a different model to usefully capture the important concerns. The *best arm identification problem* is such a model.

The problems that this thesis explores are thus, the multi-armed bandit problem (with particular focus on a non-stationary switching environment) and the best arm identification problem.

The multi-armed bandit problem has been explored extensively with many solutions proposed. One of the oldest, which until recently had been largely ignored, is Thompson Sampling. It has shown empirically impressive results and has strong theoretical guarantees for the stationary multi-armed bandit problem. It also has the advantages of simplicity of implementation and a low computational cost.

This thesis proposes that the general principles behind the Thompson Sampling solution are appropriate for a wider selection of decision-making problems.

We thus propose Thompson Sampling-inspired algorithms for the switching multi-armed bandit problem and for the best arm identification problem.

1.1 Research questions

The main questions this thesis wishes to investigate are,

- What is an appropriate Thompson Sampling model for non-stationary multi-armed bandit problems?
- Does the idea of sampling to guide exploration compare favourably to other non-stationary policies?
- How do we modify Thompson Sampling so that the level of exploration is appropriate for the best arm identification problem?

1.2 Contributions

The contributions of this thesis are as follows,

Regret bounds for Optimistic Thompson Sampling A proof is given for a bound on Optimistic Thompson Sampling (an existing algorithm in the literature) using the proof techniques used by Agrawal and Goyal to bound the Thompson Sampling algorithm. This is also shown for a modification to Optimistic Thompson Sampling we call Optimism for the Underdog Thompson Sampling (OUTS).

Changepoint Thompson Sampling A class of Thompson Sampling algorithm is developed for non-stationary switching environments. Instances of the class are made for two models of switching

- **Global** Where all actions change behaviour at the same time.
- **Per-Arm** Where the behaviour of an action can change independently of the other actions.

For each model of switching two scenarios are considered,

- **Known switching rate** The switching rate is specified by a parameter.

- **Unknown switching rate** The switching rate is inferred.

A large set of experiments are also presented to demonstrate the effectiveness of the algorithms developed.

Order-Statistic Thompson Sampling A class of algorithm generalising Thompson Sampling is developed for the best arm identification problem. The following is shown,

- **Empirical results** Comparing the algorithm to others in the literature to show the effectiveness of the approach.
- **Bound on Simple Regret** A weak bound on simple regret is given for the entire class of algorithm.

Maximum Boundary of Pairs An alternative best arm identification algorithm using a Thompson Sampling step is developed and empirical results are shown to demonstrate its effectiveness.

Measure of Aggression A measure is proposed to characterise the behaviour of a bandit algorithm as opposed to characterising the problem. The main purpose of the measure is to compare different variants of the Order-Statistic Thompson Sampling strategy. However it is not limited in use to just this algorithm class.

1.3 Thesis structure

In chapter 2 we introduce much of the background material and literature review discussing the required concepts and existing solutions to the problems this thesis considers.

In chapter 3 we introduce the Thompson Sampling algorithm (as well as Optimistic Thompson Sampling). We present the proof of an upper-bound on the cumulative regret of Optimistic Thompson Sampling as a secondary contribution. This follows closely the proof structure of Agrawal and Goyal for the Thompson Sampling algorithm. We also contribute another minor modification to Thompson Sampling that we term "Optimism for the Underdog" Thompson Sampling (OUTS) and show its regret is bounded similarly, providing some empirical evidence it also marginally outperforms Thompson Sampling.

In chapter 4 we contribute a class of algorithm appropriate for switching multi-armed bandit problems. We derive a Thompson Sampling style algorithm for two types of switching model and provide a large set of empirical evaluations to ascertain their performance. This forms one of the major contributions of the thesis.

In chapter 5 we introduce another major contribution, a class of algorithm we call Ordered-Statistic Thompson Sampling. We provide a weak-bound on the probability of error for this class of algorithm and empirically show it to be an effective solution. We further contribute an alternative algorithm we term Maximum Boundary of Pairs (MBoP) and also empirically show its effectiveness.

Chapter 6 then concludes with a discussion of the work, its context and speculations concerning further work.

Chapter 2

Background

Before presenting the contributions of this thesis we must first cover the relevant background material, both for understanding and in order to place the contributions in the wider context of the literature.

The explanation of Thompson Sampling itself is left until Chapter 3. The algorithm is a Bayesian method which, as the name implies, uses sampling. Therefore Bayesian modelling and, for completeness, methods for sampling from some common distributions are given. Following this, the major theoretical results and algorithms from the stationary multi-armed bandit problem, non-stationary multi-armed bandit problem and best arm identification problem are summarised.

2.1 Bayesian modelling

There are two distinct views about how probabilities should be interpreted. The *frequentist* interpretation views the probability of an event to be the proportion of times the event occurs in repeated trials of a given experiment or process. The *Bayesian* interpretation instead views a probability as a *degree of belief* in a given event. The Bayesian perspective allows one to model the uncertainty of a given process as probabilities. A given process could be described by a space of possible hypotheses. The higher the probability for a given hypothesis the more certain we are the hypothesis is true. The Bayesian view has been employed extensively in the machine learning community. The belief of an agent can be modelled by a probability distribution. As observations from the world are seen these beliefs can then be updated. Baye's rule is a thereom of probability that can be used to perform this update process. Letting h be a hypothesis and o an observation

Baye's rule can be stated as

$$P(h|o) = \frac{P(o|h)P(h)}{P(o)}. \tag{2.1}$$

$P(h)$ is the *prior* belief in the hypothesis before observing o. $P(o|h)$ is the *likelihood* of observing o given the hypotheis h. $P(h|o)$ is the *posterior* belief in he hypothesis after observing o. This update rule naturally chains in scenarios where observations arrive sequentially; the posterior resulting from the update of one observation becomes the prior for the update from the next observation.

2.2 Choice of prior

When modelling using the Bayesian framework the practitioner is forced to specify explicitly their prior beliefs upfront in the form of the distribution $P(h)$. A question is how to choose the prior beliefs? Naturally without any knowledge about the phenomenon to be modelled we may want to make all possible hypotheses equally likely, so that no hypothesis is more likely than any other. This is known as the *principle of indifference*. Another popular method of choosing a prior is the *principle of maximum entropy* [29]. As the name implies the prior is chosen to maximise its entropy given what is known. A motivating interpretation of this is that, from an information theoretic perspective (in the sense of Shannon), by maximising the entropy the information gain between the prior and posterior distribution is also minimised.

It is also desirable for the prior (and posterior) to be closed-form expressions. This is especially true in online situations where updates occur repeatedly in a sequential fashion through time. The main benefits of a closed form solution are the reduced space and computational costs for storing and calculating the updated beliefs. Bayes rule can be viewed as a function that takes a prior belief as an argument and returns a posterior belief. The scenario being modelled determines the hypothesis space and the likelihood term $P(o|h)$, and so therefore specifies the function Baye's rule describes. Therefore for a given likelihood, it is desirable to have a closed-form input prior distribution function that leads to a closed-form output posterior distribution function. Since in sequential update settings a posterior becomes a new prior, the posterior under the action of Bayes rule should also result in a closed-form distribution, and so on. Such pairings of

likelihood functions and prior/posterior closed-form families exist. A functional form whose prior maps to a posterior with the same functional form for a given likelihood is a special case of this property. A functional form which exhibits this behaviour is called a *conjugate prior*. An example pair of a likelihood and conjugate prior is the Bernoulli distribution and the Beta distribution.

The Bernoulli distribution can be defined by a parameter p. Intuitively we might think of this parameter as modelling the probability of flipping a head from a biased coin. The likelihood is then defined as,

$$P(o|h=p) = \begin{cases} p; & \text{if } o = \text{heads} \\ 1-p; & \text{if } o = \text{tails} \end{cases} \tag{2.2}$$

The Beta distribution is the conjugate prior for the Bernoulli likelihood. Continuing the biased coin example, the Beta distribution assigns a belief (probability) to each hypothesis. The hypothesis space is candidate values of the parameter p, p is in a fixed range between 0 and 1 as is the Beta distribution. The Beta distribution has two parameters, α and β, and can be defined as,

$$P(h=p) = \frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha,\beta)}, \tag{2.3}$$

where $B(\alpha,\beta)$ is the Beta function. To see that the Beta distribution is conjugate to the Bernoulli distribution, we apply Bayes rule and show that the form of the posterior is also a Beta distribution. Consider the case where we flip a coin and observed a head, we get,

$$P(h=p|o=\text{head}) = \frac{P(o=\text{head}|h=p)P(h=p)}{P(o=\text{head})} \tag{2.4}$$

$$= \frac{p\frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha,\beta)}}{\int_0^1 \frac{qq^{\alpha-1}(1-q)^{\beta-1}}{B(\alpha,\beta)}dq} \tag{2.5}$$

$$= \frac{p^{\alpha}(1-p)^{\beta-1}}{B(\alpha+1,\beta)}. \tag{2.6}$$

We see that the posterior is also a Beta distribution. The case of $o =$ tails follows similarly.

There are quite a few conjugate priors, mostly in the exponential family of distributions. Table 2.1 lists a few; a more extensive list can be found in a technical report by Fink [20].

Likelihood	Likelihood parameters	Conjugate prior	Prior parameters	Posterior parameters
Bernoulli	θ	Beta	α, β	$\alpha + \mathbb{1}(x = 1)$, $\beta + \mathbb{1}(x = 0)$ (where $x \in \{0, 1\}$)
Normal with known variance σ^2	μ, σ	Normal	m, s	$\frac{m+\theta^2+xs^2}{\theta^2+s^2}$, $\left(\frac{1}{s^2} + \frac{1}{\sigma^2}\right)^{-1}$ (where $x \in \mathbb{R}$)
Poisson	λ	Gamma	k,θ	$k + x$, $\frac{\theta}{\theta+1}$
Uniform	θ	Pareto	$x_{\max}$, k	$\max(x, x_{\max})$, $k + 1$

Table 2.1: Table of conjugate priors

2.3 Generating random variates

The main contributed algorithms in this thesis require the ability to sample randomly from various distributions. In this section we will briefly review common methods used to sample from some of these distributions such as the Normal and Beta distribution. It is first useful to question what do we mean by random? A "true" random number generator would return a random variate according to a distribution independently from any other random variates previous generated. This may be required in applications such as security or gambling where an adversary would predict the generated variates to exploit you if this was not the case. In practice generating "true" random numbers normally involves the measurement of physical processes assumed to themselves be random. For instance the decay of a radioactive material. However, this can potentially be a time consuming process. In many other applications, such as statistical modelling, it is not required to be truly random. The process just needs to statistically appear random. Random variates can then be produced by *pseudo-random* number generators. Pseudo-random number generators are deterministic put produce outputs that to many statistical tests appear random. They start at some seed state and then update this state to produce successive variates. Due to this determinism when the generator returns to the same state it will produce the same variate. This and the finite representation of the numbers means that the generator will eventually re-enter a state and so the generator will cycle through the variates it generates for a particular seed. The longer the period until the repetition happens the better the random number generator is judged be. Another measure of a good random number generator is whether it is k-distributed.

Matsumoto and Nishimura define k-distributed as follows [52],

Definition 2.3.1 (Matsumoto and Nishimura)**.** *A pseudo-random sequence x_i of w-bit integers of period P satisfying the following condition is said to be k-distributed to v-bit accuracy: let* $\text{trunc}_v(x)$ *denote the number formed by the leading v bits of x, and consider P of the kv-bit vectors*

$$(\text{trunc}_v(x_i), \text{trunc}_v(x_{i+1}), \ldots, \text{trunc}_v(x_{i+k-1}))(0 \leq i < P). \tag{2.7}$$

Then,each of the $2^k v$ possible combinations of bits occurs the same number of times in a period, except for the all-zero combination that occurs once less often. For each $v = 1, \ldots, w$, let $k(v)$ denote the maximum number such that the sequence is $k(v)$-distributed to v-bit accuracy.

We will now go on to mention some methods for generating pseudo-random uniform variates, and then using these, how variates from some other distributions are generated.

2.3.1 Uniform variates

The uniform distribution is the simplest distribution for which to generate pseudo-random variates. A popular procedure to produce pseudo-random numbers provided in the standard library of many current programming languages and many numerical libraries [42, 30, 16, 56] is the Mersenne Twister, developed by Matsumoto and Nishimura [52]. Some benefits of the Mersenne Twister over some other pseudo-random number generators like most linear congruential generators [36] are,

- A large prime period of $2^{19937} - 1$ using only 634 words.
- 623-distributed to 32 bits (equidistributed up to 623 dimensions for 32-bit values).

2.3.2 Normal variates

In order to generate any Normal random variates we just need a method to sample variates from a standard Normal distribution ($\mathcal{N}(\mu, \sigma)$). Once we have such a procedure the variate for a standard Normal distribution can be shifted to obtain

the variate for the desired parameters. Given a standard Normal variate Z_s we can obtain the desired variate Z_d by the following transformation,

$$Z_d = Z_s\sigma + \mu. \tag{2.8}$$

There are many methods to sample Normal variates. Two such methods are the Box-Muller transform and polar method. Both assume that Uniform random variates can be generated.

Box-Muller Transform

The Box-Muller transform was introduced by Box and Muller in 1958 [11]. The procedure requires the generation of two independent uniform variates U_1 and U_2 (Uniform$(0, 1)$). It produces two Normally distributed random variates Z_1 and Z_2. It is described in Algorithm 2.1.

Algorithm 2.1 Box-Muller Transform

Generate $U_1 \sim$ Uniform$(0, 1)$.
Generate $U_2 \sim$ Uniform$(0, 1)$.

Return $Z_1 = \sqrt{-2 \ln U_1} \cos(2\pi U_2)$
and $Z_2 = \sqrt{-2 \ln U_1} \sin(2\pi U_2)$.

Polar method

The polar method is very similar to the Box-Muller transform [68]. It requires two uniform random variates U_1 and U_2 but avoids the need to compute the sine and cosine functions which can be computationally expensive. It is the method implemented in the `numpy.random` library for the Python programming language and is described in Algorithm 2.2.

2.3.3 Beta variates

Again there are many methods to produce Beta variates [17]. The best method to use depends on the hyperparameters of the distribution α and β. One method used requires the generation of two Gamma variates. This procedure is specified in Algorithm 2.3. We will not discuss how to generate Gamma variates in this thesis, but the interested reader can refer to one such method by Marsaglia and Tsang

Algorithm 2.2 Polar Method

repeat
Generate $U_1 \sim \text{Uniform}(0, 1)$.
Generate $U_2 \sim \text{Uniform}(0, 1)$.
Let $V_1 = 2U_1 - 1$.
Let $V_2 = 2U_2 - 1$.
Let $S = V_1^2 + V_2^2$.
until $S > 1$

Return $Z_1 = \sqrt{\frac{-2 \ln S}{S}} V_1$
and $Z_2 = \sqrt{\frac{-2 \ln S}{S}} V_2$.

[51]. The SciPy library for the Python programming language uses a combination of the method requiring Gamma variates and a method called Johnk's method [17] which is described in Algorithm 2.4.

Algorithm 2.3 Beta variate generation using Gamma variates

Generate $G_1 \sim \Gamma(\alpha, \theta)$.
Generate $G_2 \sim \Gamma(\beta, \theta)$.

Return $B_1 = \frac{G_1}{G_1 + G_2}$.

Algorithm 2.4 Johnk's Method

repeat
Generate $U_1 \sim \text{Uniform}(0, 1)$.
Generate $U_2 \sim \text{Uniform}(0, 1)$.
Let $Y_1 = U_1^{\frac{1}{\alpha}}$
Let $Y_2 = U_2^{\frac{1}{\beta}}$
Let $S = Y_1 + Y_2$.
until $S \leq 1$

Return $B = \frac{Y}{S}$

2.4 Concentration of measure

The decision making problems we will present are stochastic in nature, and the algorithms we propose are themselves probabilistic. In this section we will cover a

family of powerful statistical techniques that are extensively used in the analysis of probabilistic algorithms. The techniques concern *concentration of measure*. Although a function of a large number of random variables can assume a large range of possible values, with many such processes we observe that values actually tend to occupy a very narrow range. That is, they tend to concentrate in this fixed range. Concentration of measure is the phenomenon of this tendency. An example stems from a staple process in probability theory, coin flipping. Imagine flipping a fair coin for a large number of times and recording the number of heads that are observed. Then imagine repeating this process itself many times. What we would find is that the number of heads recorded would concentrate near to half the number of flips, and never stray, relatively, that far from it. We will briefly summarise and present some of the useful results in this body of work.

2.4.1 Markov's Inequality

Probably one of the most general statements for concentration of measure is Markov's inequality. Markov's inequality bounds the probability of a non-negative random variable exceeding a given value in terms of its expected value. We now formally state the result.

Theorem 2.4.1. *Let X be a non-negative random variable and let $a > 0$, then*

$$P(X \geq a) \leq \frac{\mathbb{E}[X]}{a}.$$

Proof. Denote the indicator function of an event A as $\mathbb{1}(A)$. $\mathbb{1}(A) = 1$ if event A occurs and 0 if it does not. By the definition of $\mathbb{1}(A)$ and the fact $a > 0$,

$$\mathbb{1}(A) \leq 1, \tag{2.9}$$

$$a\mathbb{1}(A) \leq a, \tag{2.10}$$

for any event A. It then follows that for the event $X \geq a$,

$$a\mathbb{1}(X \geq a) \leq a, \tag{2.11}$$

$$\leq X. \tag{2.12}$$

Now taking expectations of both sides we get,

$$\mathbb{E}\left[a\mathbb{1}(X \geq a)\right] = aP(X \geq a), \tag{2.13}$$

$$\leq \mathbb{E}\left[X\right]. \tag{2.14}$$

Since $a > 0$ it then follows that,

$$P(X \geq a) = \frac{\mathbb{E}(X)}{a}. \tag{2.15}$$

□

2.4.2 Chernoff-Hoeffding Inequality

If we think back to the coin flipping example described earlier, where we notice that after a large number of trials the amount of heads we have observed is never too far from the expected value. Chernoff-Hoeffding inequalities formalise this notion for processes described by sums of independent random variables. The inequalities bound the probability of the sum being larger (or smaller) than the expected value. The two most useful forms of the bounds are given below.

Theorem 2.4.2. *Let X_i, $i \in \{1, \ldots, N\}$, be independently random variables in $[0, 1]$ and let $X = \sum_{i=1}^{N} X_i$. Then for $t > 0$,*

$$P\left(X > \mathbb{E}\left[X\right] + t\right) \leq e^{\frac{-2t^2}{N}}, \tag{2.16}$$

$$P\left(X < \mathbb{E}\left[X\right] - t\right) \leq e^{\frac{-2t^2}{N}}. \tag{2.17}$$

Theorem 2.4.3. *Let X_i, $i \in \{1, \ldots, N\}$, be independently random variables in $[0, 1]$ with mean p_i and let $X = \sum_{i=1}^{N} X_i$. Then for $\delta \in (0, 1)$,*

$$P\left(X > (1+\delta)\mathbb{E}\left[X\right]\right) \leq e^{\frac{-\delta^2 \mathbb{E}[X]}{3}}, \tag{2.18}$$

$$P\left(X < (1-\delta)\mathbb{E}\left[X\right]\right) \leq e^{\frac{-\delta^2 \mathbb{E}[X]}{2}}. \tag{2.19}$$

Proof. Here we just proof the bound for the right tail (equation 2.18) to illustrate

the general technique for showing Chernoff-type bounds.

$$P\left(X > (1+\delta)\mathbb{E}\left[X\right]\right) = P\left(e^{\lambda X} > e^{\lambda(1+\delta)\mathbb{E}[X]}\right) \quad \text{(monotonicity of exponentiation)} \tag{2.20}$$

$$\leq \frac{\mathbb{E}\left[e^{\lambda X}\right]}{e^{\lambda(1+\delta)\mathbb{E}[X]}} \quad \text{(Markov's inequality)} \tag{2.21}$$

$$\leq \frac{\prod_{i=1}^{N}\mathbb{E}\left[e^{\lambda X_i}\right]}{e^{\lambda(1+\delta)\mathbb{E}[X]}} \quad \text{(independence of } X_i\text{)} \tag{2.22}$$

Now each X_i lies in the interval $[0,1]$. Since $e^{\lambda X_i}$ is convex then we can upper bound it by a straight line between points $(0, e^{\lambda.0} = 1)$ and $(1, e^{\lambda.1} = e^{\lambda})$. This means that,

$$e^{\lambda X_i} \leq (e^{\lambda} - 1)X_i + 1, \tag{2.23}$$

$$\mathbb{E}\left[e^{\lambda X_i}\right] \leq \mathbb{E}\left[(e^{\lambda} - 1)X_i + 1\right], \tag{2.24}$$

$$= (e^{\lambda} - 1)\mathbb{E}\left[X_i\right] + 1, \tag{2.25}$$

$$= (e^{\lambda} - 1)p_i + 1, \tag{2.26}$$

$$= e^{\lambda}p_i + 1 - p_i. \tag{2.27}$$

Therefore,

$$P\left(X > (1+\delta)\mathbb{E}\left[X\right]\right) \leq \frac{\prod_{i=1}^{N}(p_i e^{\lambda} + 1 - p_i)}{e^{\lambda(1+\delta)\mathbb{E}[X]}} \tag{2.28}$$

The arithmetic mean of non-negative values upper bounds the geometric mean of those same values. That is, for $x_i \geq 0$,

$$\frac{1}{n}\sum_{i=1}^{n} x_i \geq \left(\prod_{i=1}^{n} x_i\right)^{1/n}. \tag{2.29}$$

Applying this to Equation 2.28 we get,

$$P\left(X > (1+\delta)\mathbb{E}\left[X\right]\right) \leq \frac{\left(pe^{\lambda} + 1 - p\right)^{N}}{e^{\lambda(1+\delta)Np}}, \tag{2.30}$$

where $p = \sum_{i=1}^{N} p_i/N$.

The bound is minimised with respect to λ by differentiating with respect to λ and finding the value of λ that makes the derivative equal zero. The value of λ

is such that $e^{\lambda} = \frac{(1+\delta)(1-p)}{(1-p-\delta p)}$.

$$
\begin{aligned}
P\left(X > (1+\delta)\mathbb{E}\left[X\right]\right) &\leq \left(\left(\frac{(1-p-\delta p)}{(1+\delta)(1-p)}\right)^{(1+\delta)p} \frac{(1-p)}{(1-p-\delta p)}\right)^{N} \\
&= e^{-N(p(1+\delta)\log(1+\delta)+(1-p-\delta p)\log(\frac{1-p-\delta p}{1-p}))} \\
&= e^{-N(p(1+\delta)\log(1+\delta)-(1-p-\delta p)\log(\frac{1-p}{1-p-\delta p}))} \\
&= e^{-N(p(1+\delta)\log(1+\delta)-(1-p-\delta p)\log(1+\frac{p\delta}{1-p-\delta p}))} \\
&\leq e^{-N(p(1+\delta)\log(1+\delta)-\delta p)} && (\text{since } \log(1+x) < x) \\
&\leq e^{-\mathbb{E}[X](\frac{2\delta(1+\delta)}{2+\delta}-\delta)} && (\text{since} \log(1+\delta) > \frac{2\delta}{2+\delta}) \\
&= e^{-\mathbb{E}[X](\frac{\delta^2}{2+\delta})} \\
&= e^{-\frac{\delta^2}{3}\mathbb{E}[X]} && (\text{since } 1 \geq \delta)
\end{aligned}
$$

□

2.5 Reinforcement learning

This thesis focuses on a variety of multi-armed bandit problem. This problem can be viewed as a special simplified model within the wider Reinforcement Learning problem. Reinforcement Learning concerns learning via a process of interaction with an environment. This differs from the most common area of focus in the field of Machine Learning, Supervised Learning, where learning is done via examples provided by a supervisor. Reinforcement Learning has its history in two fields, that of Psychology and also of Dynamic Programming. The paradigm considers an agent which exists in some environment. The agent may possibly be able to sense or observe measurements about the environment and their place within it (this is not necessarily the case). The agent can perform actions in the environment, and can make decisions as to what actions to perform. How appropriate a decision was is measured by a *reward signal*, which is a scalar valued function mapping how good a decision was to a real number. When the agent performs an action it may have an effect on the environment and the agents place in it and then the environment reveals the value of the reward to the agent, which it can then use to improve future decisions. This process is summarised in Figure 2.1.

The objective is for the agent to learn a *policy* or *strategy*, that is a method of choosing actions that maximises the reward they receive. From this perspective

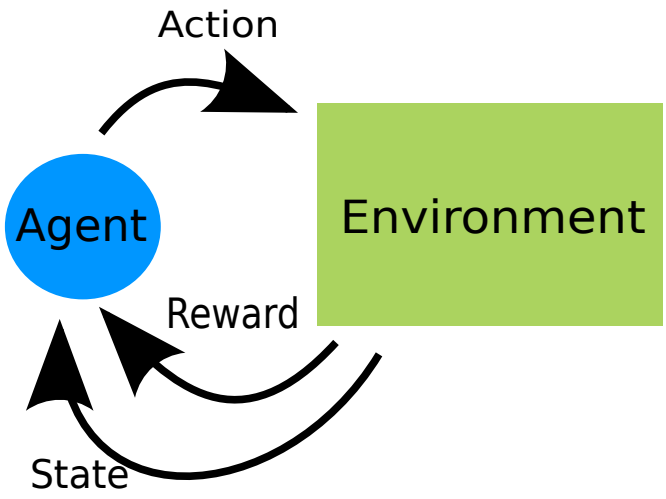


Figure 2.1: Model of Reinforcement Learning

the classic multi-armed bandit problem is Reinforcement Learning where there is not an observable state and where there is an assumption that the reward received by an action at one time step is independent but identically distributed to the reward received at another time step.

2.6 Stationary stochastic multi-armed bandits

The multi-armed bandit is a common and arguably the simplest model which exhibits the exploration-exploitation trade-off. The dilemma in sequential decision-making, between choosing actions because they are thought to be good, and choosing actions in order to learn more about their consequences. The model has been around since at least 1933 [69]. Robbins introduced his formalisation of the problem in 1952 [60].

The model takes its name from the one-arm bandit, a gambling machine with an arm that when pulled wins the gambler some money with given probability. The model considers a number of such machines, and imagines repeatedly being given the option to choose between them. The gambler wants to get the best return, and play the strategy that in expectation provides them with the most winnings.

In general we can define the problem as follows. Let there be a set of possible arms, $\mathcal{K}$, where a given arm $k \in \mathcal{K}$. At time, t, the agent chooses an arm a_t. The agent receives a reward $x_{k,t}$ by pulling arm k at time t. An agent wants to maximise the expected cumulative reward they receive,

$$S(T) = \mathbb{E}\left[\sum_{t=1}^{T} x_{a_t,t}\right]. \tag{2.31}$$

So the problem is to find a strategy that maximises this quantity. However, we more commonly consider a equivalent problem of *minimising* an alternative measure of performance called *regret*. There are many forms of regret. The most common form compares between a given strategy and the strategy that chooses the single arm that in expectation has the highest cumulative reward. The regret felt up to time, T, can be written as,

$$R(T) = \max_{k\in\mathcal{K}} \mathbb{E}\left[\sum_{t=1}^{T} x_{k,t}\right] - \mathbb{E}\left[\sum_{t=1}^{T} x_{a_t,t}\right] \tag{2.32}$$

The above formulation makes no assumptions about the rewards and the arms from which they come. Different assumptions on the rewards lead to different versions of the problem. In the *stochastic* multi-armed bandit problem we assume that the rewards associated with arm k come from some stationary distribution π_k parameterised by ρ_k with mean μ_k. The regret felt by a strategy for the stochastic multi-armed bandit can be expressed as,

$$R(T) = \sum_{t=1}^{T} \mu_{\max} - \mu_{a_t}, \tag{2.33}$$

where $\mu_{\max} = \max_{k\in\mathcal{K}} \mu_k$. The quantity $\Delta_k = \mu_{\max} - \mu_k$ is known as the gap between the best arm and arm k.

In this section we will introduce some theoretical results known about the multi-armed bandit problem, as well as highlight some empirical studies of the problem in the literature. We will then introduce some of proposed strategies for this problem. Since this thesis is about Thompson Sampling, and its application to a broader set of problem domains, we will defer formally introducing the algorithm to chapter 3, where we will give it a more thorough treatment.

2.6.1 Lower bound of regret for the stochastic multi-armed bandit

Lai and Robbins provided asymptotic lower bounds of the expected regret for the stochastic multi-armed bandit problem [41]. The result applies to any strategy for which $R(T) = o(T^a)$ for all $a > 0$ as $T \to \infty$. This intuitively means that the strategy is greedy in the limit. Kaufmann et al. call this condition *strongly consistent* (since it implies the strategy is what Lai and Robbins call *consistent*, that $\lim_{T\to\infty} \mathbb{E}\left[S(T)\right]/T = \mu_{\max}$). Let $D_{\mathrm{KL}}(\mu_i, \mu_j) = \mu_i \ln \frac{\mu_i}{\mu_j} + (1-\mu_i) \ln \frac{1-\mu_i}{1-\mu_j}$ (the Kullback-Leibler divergence between two Bernoulli distributions with parameters μ_i and μ_j). The theorem states that

$$\liminf_{T\to\infty} \frac{R(T)}{\ln T} \geq \sum_{k\in\mathcal{K}} \frac{\mu_{\max} - \mu_k}{D_{\mathrm{KL}}(\mu_k, \mu_{\max})}. \tag{2.34}$$

Letting $N_{k,T}$ be the number of times the strategy pulled arm k up to time T, the bound can be written as,

$$\liminf_{T\to\infty} \frac{\sum_{k\in\mathcal{K}} \mathbb{E}\left[N_{k,T}(\mu_{\max} - \mu_k)\right]}{\ln T} \geq \sum_{k\in\mathcal{K}} \frac{\mu_{\max} - \mu_k}{D_{\mathrm{KL}}(\mu_k, \mu_{\max})}.$$

They call any strategy that meets this lower bound with equality *asymptotically efficient* (alternatively it can be called asymptotically optimal). It also is useful to note that for a strategy that satisfies

$$\lim_{T\to\infty} \frac{\mathbb{E}\left[N_{k,T}\right]}{\ln T} \geq \frac{1}{D_{\mathrm{KL}}(\mathcal{K}, \mu_{\max})}, \tag{2.35}$$

with equality for all $k \in \mathcal{K}$ then (2.34) is satisfied with equality and the strategy is asymptotically optimal.

2.6.2 Empirical studies for stochastic multi-armed bandits

Most theory related to bandit algorithms gives asymptotic optimality guarantees. To the practitioner this may not be of much use if the performance initially for small time horizons is poor. Empirical studies may give more confidence as to an algorithms viability in this respect. There have been several papers

that have compared many of the algorithms to be mentioned in this section. Vermorel and Mohri as well as Kuleshov and Precup have empirically evaluated many algorithms in the literature for the standard stochastic multi-armed bandit problem [72, 40]. One surprising observation of such studies is that some of the simpler strategies such as *epsilon*-greedy (described later in this section) are often hard to beat. However this assumes appropriate tuning of parameters for the problem. We also mention here an empirical study by Chapelle and Li [15] that gave an empirical evaluation of Thompson Sampling, along with many other popular bandit algorithms.

2.6.3 Gittins index

An *index policy* is any strategy for which an agent can compute an *index* independently for each arm without knowledge of the other arms. The arm associated with the largest index is the arm chosen. The Gittins index is one such index policy introduced by Gittins as a dynamic allocation index [23] and further described in [12]. The Gittins index was first proposed in the context of a discounted multi-armed bandit problem.

The discounted multi-armed bandit problem has the same basic setup as the standard multi-armed bandit. However, the difference is in the objective. Instead of defining an optimal strategy as one which minimises the expected regret, an optimal strategy is instead defined as one which minimises the expected discounted regret. We can define this as,

$$R_\xi(T) = \sum_{t=1}^{T} \xi^{T-t}(\mu_{\max} - \mu_{X_t}). \tag{2.36}$$

Gittins showed that this can be solved by, at each round, pulling the arm with the highest dynamic allocation index, or Gittins index, for that round. The calculation of the indices corresponds to finding optimal stopping times for a given Markov chain. Let the reward received by pulling arm k at time t be $X_{k,t}$. The Gittins index method is Bayesian in the sense that in order to calculate an index for an arm, k, an agent must have a prior "belief" distribution, $P(\mu_k)$ over

the mean of the arm, μ_k. The Gittins index, $\kappa(P(\mu_k))$, is defined as,

$$\kappa(P(\mu_k)) = \sup_{\tau} \left[\frac{\int \mathbb{E}_{\mu_k}[X_{k,1}] + \sum_{t=1}^{\tau-1} \xi^t \mathbb{E}[X_{k,t}|X_{k,1}, \ldots, X_{k,t-1}]\, dP(\mu_k)}{\int \sum_{t=0}^{\tau} \xi^t dP(\mu_k)} \right]. \tag{2.37}$$

The indices can be found using dynamic programming. For the discounted multi-armed bandit this is the optimal solution, and completely solves the exploration-exploitation trade off for this case [75]. The solution maximises the expected cumulative discounted reward the agent receives. The limit to this solution as $\xi \to 1$ can be found and so the technique can be applied to the undiscounted stochastic multi-armed bandit problem. There are some disadvantages to the Gittins index as a strategy in the stochastic multi-armed bandit problem. Firstly the best time complexity for computing the indices is $O(|\mathcal{K}|^3)$ [57], so even for the problems where this can be directly applied, if the number of arms is reasonably large then the calculation becomes unfeasible if the decisions need to be made quickly. Secondly, Brezzi and Lai show that the Gittins index policy leads to *incomplete* learning. That is, the policy eventually fixes on pulling only a single arm, and that this arm has a non-zero probability of being sub-optimal [12]. This may at first appear inconsistent since the Gittins index is suppose to be an optimal policy. However as they state

> Such incomplete learning by the optimal policy can be attributed to the discount factor which downweights the need for acquiring information to benefit long-run future performance

Scott points out two further concerns [63]. The Gittins index policy is only optimal for scenarios in which the arms are independent and can be far from optimal when this is not the case. The geometric nature of the discounting implies that the decisions should be equally spaced in time. In many applications this is not necessarily true. It is important then to look at other policies for the multi-armed bandit problem that tackle some of these perceived problems.

2.6.4 Semi-uniform methods

ϵ-greedy

ϵ-greedy policies make an explicit trade-off between exploration and exploitation. The policy chooses the perceived best action, with probability $1 - \epsilon$. It instead

explores and picks a random action uniformly over the possible choices with probability ϵ. The algorithm is presented in Algorithm 2.5. The policy was introduced by Sutton [66]. He produced positive results in applying reinforcement learning to control problems that had previously eluded reinforcement techniques. One aspect different to previous attempts was his use of online algorithms. The online nature of the algorithms exhibit the exploration-exploitation dilemma and so required an exploration strategy. As stated ϵ-greedy will eventually over explore in terms of the multi-armed bandit problem. This is because ϵ is fixed, and so the number of times a sub-optimal arm is chosen must grow linearly with the running time of the strategy. However, ϵ can be adapted as a function of time, ϵ_t. Singh et al. showed that when $\epsilon_t = c/t$ ($0 < c < 1$), that is the exploration is decreased at a particular rate, then the strategy falls into the category of GLIE (Greedy in the limit with infinite exploration) strategies [64]. In non-stationary scenarios, which are covered later in this chapter, the proposed adaptation explores too little and so alternative adaptations must be used. A key disadvantage of ϵ-greedy, regardless of the adaptation, is that the exploration is agnostic to the quality of the arms. When exploration happens ϵ amount of the times, the worst arm is chosen equally likely as any other arm (the best, the second best).

Algorithm 2.5 ϵ-Greedy

Let $\hat{X}_{t,k}$, be the empirical mean reward for arm k at time t.

Pull each arm once.

for $t = K, \ldots, T$ **do**

Pull arm $a_t = \begin{cases} \text{argmax}_i \hat{X}_{t,i} & \text{with probability } \epsilon \\ \text{select randomly} & \text{with probability } 1 - \epsilon \end{cases}$

Update $\hat{X}_{t,a_t}$

end for

2.6.5 Softmax

We have discussed how ϵ-greedy is disadvantaged given that when it elects to explore, it chooses amongst arms with equal probability. An arm that is considered the next best is as likely to be pulled as one that is considered the worst. This suggests that a more effective strategy would be one where the rank of the arm

determined the amount the arm was pulled. Any strategy that determines the selection probability of an arm based on the weighting or ranking of their estimated reward, such that arms with higher estimates have higher selection probabilities, is known as a *softmax* strategy. Such strategies can be seen to take the greedy myopic max policy and soften it.

A popular softmax strategy is the Boltzmann exploration policy (also known as the Gibbs exploration policy) that uses the Boltzmann distribution in defining the selection probabilities of the arms. The algorithm is parametrised by $\tau > 0$, known as a temperature. The term temperature stems from the use of the Boltzmann distribution in statistic physics, where the distribution can be used to describe the proportion of particles occupying given energy levels in a physical system. The strategy is presented in Algorithm 2.6.

Algorithm 2.6 Boltzmann exploration

Let τ be an exploration parameter.
Let $\hat{X}_{t,k}$, be the empirical mean reward for arm k at time t.

Pull each arm once.

for $t = K, \ldots, T$ **do**
  Let $a_t = i$ with probability $\frac{e^{\tau \hat{X}_{t-1,i}}}{\sum_{k=1}^{K} e^{\tau \hat{X}_{t-1,k}}}$
  Pull arm a_t
  Update $\hat{X}_{t,k}$
end for

2.6.6 Optimism under uncertainty

A popular class of algorithms for the multi armed bandit are collectively known as *upper confidence bound* methods (UCB). The guiding idea for the strategy is the principle of *optimism in the face of uncertainty*. That is, when we are uncertain to the payoff of a given action/arm we should be optimistic and assume the payoff will be the highest we can statistically expect. The method then employs standard statistical tools in the form of upper confidence bounds to determine what is considered statistically expected.

In general the estimated payoff for an arm in UCB methods can be decomposed in to two parts. The first term is the empirical average payoff $\hat{X}_t$, and the second is a padding term P_t determined by the upper confidence methods. The

optimistic payoff U_t of an arm then becomes,

$$U_t = \hat{X}_t + P_t. \tag{2.38}$$

The padding function needs to be large when we are most uncertain about an arm, and get smaller as we learn about the arm's true expected payoff. The general algorithm is presented in Figure 2.7.

Algorithm 2.7 General UCB based method

Let $\hat{X}_{i,t}$ be the empirical mean reward of arm i and time t.
Let K be the number of arms
Let $n_{i,t}$ be the number of times arm i has been pulled at time t
Let $P_{i,t}$ be a padding function for arm i at time t.
Initialisation:
Play each arm once.
Main Loop:
for $t = K + 1, \ldots, T$ **do**
  Let $U_i = \hat{X}_{i,t} + P_{i,t}$.
  Pull arm $a_t = \text{argmax}_i\, U_i$.
  Update $\hat{X}_{a_t,t+1}$.
end for

Methods using confidence bounds to balance the exploration-exploitation trade off go back at least to work by Kaelbling [32]. Auer applied the ideas to more challenging settings [6] and provided the first finite-time analysis of confidence based algorithms with UCB1 [7]. This popularised the method and fits the decomposition of Equation 2.38. UCB1 assumes rewards in the interval $[0, 1]$. The padding function is $P_t = \sqrt{\frac{2 \ln t}{n_t}}$, where n_t is the number of times the arm has been pulled up until time t. The algorithm is described in Algorithm 2.8.

The expected regret of UCB1 can be bounded as follows,

$$\mathbb{E}\left[R_T\right] \leq \left[8 \sum_{i=2}^{K} \frac{1}{\Delta_i}\right] \ln T + (1 + \frac{\pi^2}{3}) \left(\sum_{i=2}^{K} \Delta_i\right), \tag{2.39}$$

as shown by Auer et al. [7]. The main focus in the area of research is on developing more sophisticated padding functions. This often relies on subtle understanding of concentration of measure. There has been a slow but steady improvement of UCB algorithms in the quest to employ new bounding techniques to improve these padding functions. The current state of the art in this arms race are algorithms

Algorithm 2.8 UCB1

Let $\hat{X}_{i,t}$ be the empirical mean reward of arm i and time t.
Let K be the number of arms
Let $n_{i,t}$ be the number of times arm i has been pulled at time t
Initialisation:
Play each arm once.
Main Loop:
for $t = K + 1, \ldots, T$ **do**
  Let $U_i = \hat{X}_{i,t} + \sqrt{\frac{2 \ln t}{n_{i,t}}}$
  Pull arm $a_t = \text{argmax}_i \, U_i$.
  Update $\hat{X}_{a_t,t+1}$.
end for

such as KL-UCB [49], which uses the Kullback-Liebler divergence, and Bayes-UCB, which employs a Bayesian model of the arms [33]. These are asymptotically efficient in that their upper bound matches the lower bound on cumulative regret.

A potential weakness of such methods is their determinism given the past history of actions and rewards. For some applications the scenario is better modelled by a bandit algorithm with delayed rewards. That is, the reward is not known immediately after an arm is pulled, and may be received some time after. In the prevailing time before the reward has been received the agent must still make decisions. In this scenario UCB methods tend to get stuck selecting the same arm and are less likely to opt to explore [15].

2.6.7 POKER

POKER, standing for the *Price Of Knowledge And Estimated Reward*, is a bandit strategy proposed by Vermorel and Mehryar [72]. It is similar to UCB style algorithms since the index of an arm is computed as the sum of two components, the empirical mean of an arm, and an added exploratory value. The arm indexes take the form,

$$I_k = \hat{\mu}_k + \mathbb{P}\left[\mu_k \geq \hat{\mu}_{\max} + \delta_\mu\right] \delta_\mu H, \tag{2.40}$$

where μ_k is the true mean and $\hat{\mu}_k$ is the empirical mean of arm k, $\hat{\mu}_{\max}$ is the largest empirical mean, and H is the remaining time horizon left in the problem. They consider δ_μ as an estimate of the 'expected reward improvement' and it is

estimated as,

$$\delta_\mu = \frac{\hat{\mu}_{\sigma(1)} - \hat{\mu}_{\sigma(\sqrt{K})}}{\sqrt{K}}, \tag{2.41}$$

where $\hat{\mu}_{\sigma(i)} \geq \hat{\mu}_{\sigma(j)}$ iff $i < j$. This is intended to be an estimate of the expected difference $\mathbb{E}[\mu_{\max} - \hat{\mu}_{\max}]$. The terms δ_μ and H are the same for all arms and serve only to globally manage the tradeoff between exploiting (based on the index being the empirical mean) and exploring. The quantity $\mathbb{P}[\mu_k \geq \hat{\mu}_{\max} + \delta_\mu]$ is thus the component of exploration specific to a given arm. It denotes the probability of an arm having a mean larger than the current largest empirical mean plus the expected reward improvement. This is estimated as,

$$\mathbb{P}[\mu_k \geq \hat{\mu}_{\max} + \delta_\mu] = \int_{\hat{\mu}_{\max}+\delta_\mu}^{\infty} \mathcal{N}\left(x; \hat{\mu}_k, \frac{\hat{\sigma}_k}{\sqrt{n_k}}\right) dx, \tag{2.42}$$

where $\hat{\sigma}_k$ is the empirical standard deviation and n_k is the number of times arm k has been pulled. Vermorel and Mehryar found this method to perform favourably in their empirical evaluation of bandit algorithms and showed that in the limit of large H the strategy converges to one that will just pull the best arm.

2.7 Non-stationary stochastic multi-armed bandits

The multi-armed bandit problem concisely formalises a tradeoff present in decision making in the presence of uncertainty. It has seen useful application or been used to model scenarios as far-ranging as clinical trials to news story placement on websites [69, 45]. The standard treatment of the problem however assumes a well-behaved stationary stochastic process from which rewards are drawn. Sutton and Barto [67] noted,

> There are many sophisticated methods for balancing exploration and exploitation for particular mathematical formulations of the n-armed bandit and related problems. However, most of these methods make strong assumptions about stationarity and prior knowledge that are either violated or impossible to verify in applications and in the full reinforcement learning problem...

The stochastic nature of sequential decision problems for which multi-armed bandits are employed are usually never fully known. A stationary process might be appropriate for a limited number of applications, however in many cases it is not. The world in which an agent acts is prone to changes and the agent should be sufficiently capable of adapting to such changes.

An example of a multi-armed bandit application is a news story placement problem. A company wishes to place a top news story on their homepage. The more people that click on the news story the more revenue is generated for the company by way of advertisements. The company has sourced a list of stories from a news vendor and must decide which to show to a user when they visit the homepage. This clearly maps to a multi-armed bandit, where a news story is an arm in the bandit problem. The users can be imagined to come from a fixed population, with each user having a fixed interest in a given news story. The task might then be to learn which news story has the best average interest (indicated by whether a user clicked through to the story or not) for the population. The assumption that this problem is stationary may be a poor one, however. The population of users visiting the homepage may not be fixed, the number and type of people visiting the website at 02:00 GMT may be significantly different than those that visit the site at 14:00 GMT. The change in population would then account for a change in average population interest in a given story. The assumption that a user has a fixed interest level in a story is also most likely flawed; if the story is topical the interest of the story will wane as it gets further in time from the event taking place, for example.

We briefly introduce to types of non-stationary behaviour that may occur that it would be useful for an agent to adapt to, before reviewing some proposed bandit algorithms designed to do just that.

2.7.1 Types of non-stationary environment

We assume that at a particular point in time the behaviour of a system can be described by a fixed statistical law. The system at this fixed time can then be modelled by a probability distribution. The probability distribution can be parameterised by a set of sufficient statistics. It is possible that the number of sufficient statistics may be denumerable but let us assume that the distribution is specified by a finite set of parameters. Let θ_t be the parameters that specify the system at time t. In a non-stationary system the behaviour of the system can

change from one time point to the next. So that it is possible that $\theta_t \neq \theta_{t+1}$. This can be characterised in two ways dependent on how stark the change is between θ_t and θ_{t+1}. The case where any change between the state of the system from one time to the next is small and gradual we call a *drifting environment*. For example the electricity used for lighting might steadily creep up as the nights draw in more quickly. The usage changes but the past allows us to make a good prediction of future usage. When the change between states is larger and less gradual we call this a *switching environment*. For instance the stock price of a pharmaceutical company might starkly fall, and a competitor's stock might rise, suddenly if a new drug does not get approval by a regulating body (for example the Medicines and Healthcare products Regulatory Agency in the UK). The price before and after may be largely uncorrelated and could be sensibly modelled as switching. An extreme type of switching environment, that is of most relevance to this thesis, is one where there is no dependence between the parameters before and after a change. Let s_t be an indicator variable denoting whether at some change occurred at time t. Either $P(\theta_{t+1}|s_t, \theta_t) = P(\theta_{t+1}|s_t)$, meaning that the system in its current state is independent from previous states, or $P(\theta_{t+1}|\theta_t, s_t) \neq P(\theta_{t+1}|s_t)$ in which case the current state is dependent on the past. The form of switching is known as a *change-point environment*. It is possible that change in the environment is locally as we describe above, either drifting or switching, but returns to the same statistical law after a long period of time, possibly in a cyclic way. This for example would reflect processes dependent on seasonal change. It may also be advantageous to model this repeating or periodic behaviour. However, when the environment changes, locally the change will either be of a drifting or switching type.

2.7.2 Non-stationary policies

Here we cover just a few multi-armed bandit algorithms specifically designed for adapting in non-stationary environments.

Discounted-UCB

UCB policies are dependent on two statistics for each arm in order to calculate their upper confidence bounds, the number of times the arm has been pulled and the empirical mean of the rewards received by that arm. The number of times

an arm has been pulled the less effect the padding function has on the UCB estimate and the less effect a new reward will have on the empirical estimate of the mean. This means the longer the algorithm has been running the more inertia the algorithm has in adapting to a change in environment. Reducing this inertia will allow the strategy to adapt to a change. This can be done by making the strategy "forgetful". An easy way to do this is to discount the rewards (and the number of times the arm has been pulled), so that past rewards are weighted less important than recent rewards. This is done by use of a discount factor ξ. The algorithm is called *Discounted UCB* and is described in Algorithm 2.9. For UCB1 we assumed that the reward was from the unit interval, but we present Discounted UCB assuming that the reward is bounded by B.

Algorithm 2.9 Discounted UCB

Parameters: ξ, η
Let B be an upper bound on the rewards received.
Let $\hat{X}_{i,t}$ be the discounted empirical mean reward of arm i and time t.
Let K be the number of arms
Let $n_{i,t}$ be the discounted number of times arm i has been pulled at time t
Initialisation:
Play each arm once.
Main Loop:
for $t = K+1, \ldots, T$ **do**
  Let $U_i = \hat{X}_{i,t} + 2B\sqrt{\frac{\eta \log \sum_{i=1}^{K} n_{i,t}}{n_{i,t}}}$
  Pull arm $a_t = \text{argmax}_i\, U_i$.
  $n_{a_t,t+1} = \xi n_{a_t,t} + 1$,
  $\hat{X}_{a_t,t+1} = \hat{X}_{a_t,t} + \frac{x_{a_t,t} - \hat{X}_{a_t,t}}{n_{a_t,t+1}}$.
  $n_{i,t+1} = \xi n_{i,t}$ for $i \neq a_t$,
  $\hat{X}_{i,t+1} = \hat{X}_{i,t}$ for $i \neq a_t$.
end for

A difficulty is in how to set the discount factor. Too small and the algorithm is too forgetful, and never manages to produce good estimates on the arms, causing the strategy to explore more than it ought to. Too large and the algorithm has too much inertia and can not adapt to the changes that occur quickly enough. Garivier et al. considered Discounted UCB in a switching environment and gave regret bounds for this case [22]. Given the number of switches that occurred as a function of time the bound could be used to select suitable values for the parameters of the algorithm. However, the bounds are only meaningful for an environment where the number of switches grows sub-linearly with time. This

may not necessarily be applicable or at least verifiable in a practical setting, but is still a significant step in the right direction for theoretical results for algorithms designed for these settings.

Windowed-UCB

To adapt UCB-style algorithms to adapt to a changing environment the inertia present in the algorithm has to be reduced. An alternative strategy to discounting the rewards, as with Discounted UCB, is to consider only rewards from a recent window of time, and forget all rewards that are too old. This leads to the strategy Sliding-Window UCB, also studied by Garivier et al. [22] and described in Algorithm 2.10.

Algorithm 2.10 Sliding-Window UCB

Parameters: τ, η
Let B be an upper bound on the rewards received.
Let $n_{i,t}(\tau) = \sum_{s=t-\tau+1}^{t} \mathbb{1}_{\{a_s=i\}}$.
Let $\hat{X}_{i,t}(\tau) = \frac{1}{n_{i,t}(\tau)} \sum_{s=t-\tau+1}^{t} x_{i,t} \mathbb{1}_{\{a_s=i\}}$.
Let K be the number of arms
Initialisation:
Play each arm once.
Main Loop:
for $t = K+1, \ldots, T$ **do**
  Let $U_i = \hat{X}_{i,t}(\tau) + B\sqrt{\frac{\eta \log \min(\tau,t)}{n_{i,t}(\tau)}}$.
  Pull arm $a_t = \operatorname{argmax}_i U_i$.
end for

Adapt-EvE

Motivated by a real-time website optimisation problem, Hartland et al. proposed the Adapt-EvE algorithm [26]. The context of the problem was to provide a community of users with the news they are most interested in. The problem was posed as a multi-armed bandit problem where the arms were the news items, with the goal to pick the most interesting news item available. The objective of Adapt-Eve was to formulate a strategy that was able to cope with the potential dynamic changes to every user's interest, which in turn would affect which news item was the most salient. The focus was not to try to model the environment or the causes of the changes, but be capable of reacting when a change could be observed via

the lack of interest in a news item. Adapt-EvE is an extension of UCBT designed to cope with such a scenario. UCBT (UCB-Tuned) is a variant of the UCB bandit strategy proposed by Auer et al. [7]. It has been found to empirically perform much better than UCB1 in many situations. UCBT estimates the variance of the rewards from each arm in order to adapt the padding function used and is presented in Algorithm 2.12. Adapt-EvE augments UCBT with a change point mechanism, namely the Page-Hinkley test which is shown in Algorithm 2.11. The Page-Hinkley test is a frequentist changepoint hypothesis test. The null hypothesis is that the sequence of rewards can be attributed to a single statistical law, otherwise a change has occurred. The test can be computed efficiently in a recursive online fashion and under some reasonable conditions can be proved to minimise the time taken to detect a change for a given false alarm rate. The Page-Hinkley test is run along with the UCBT algorithm. When a change is detected this allows the algorithm to "reset" the UCBT bandit strategy by setting both the empirical mean, $\hat{X}_{i,t}$, and times pulled, $n_{i,t}$, of each arm to zero. A perceived problem with this approach is that when a false alarm occurs all information of the state of the arms is lost. The cost of relearning the payoff of each arm sufficiently well before making good decisions again is relatively high. In order to mitigate this problem Hartland et al. proposed a mechanism to decide, on the event of a change being detected, if the best option was to acknowledge the change and reset the UCBT bandit algorithm, or to consider the change a false alarm and keep the UCBT bandit algorithm with its existing estimates. They posed this conundrum as a separate multi-armed bandit problem. When the Page-Hinkley test detects a change, a second UCBT algorithm is initialised (called the new bandit) to "compete" against the existing UCBT algorithm (called the old bandit). Another two arm bandit algorithm is run for a set period to decide between which of the two algorithms, the old or the new bandit, will be used to select an arm/news item. During this period all new changes detected are ignored. Since the mechanism is based on using a bandit algorithm to make a decision outside of the direct question of which news item is more interesting, they call the technique *meta-bandits*. The strategy is summarised in Algorithm 2.13.

2.7.3 Connection to adversarial multi-armed bandits

In the bandit problems we have considered the rewards are assumed to come from well-behaved stochastic processes. This assumption, due to the phenomenon of

Algorithm 2.11 Page-Hinkley Test

Parameters: δ,λ
Let $\hat{x}_t = \frac{1}{t}\sum_{l=1}^{t} x_l$ (the empirical mean reward)
Let $m_T = \sum_{t=1}^{T}(x_t - \hat{x}_t + \delta)$
Let $M_T = \max_{1\leq t\leq T} m_t$
Let $H_T = M_T - m_T$ (Page-Hinkley statistic)
if $H_T > \lambda$ **then**
  Change occurred.
else
  No change occurred.
end if

Algorithm 2.12 Discounted-UCBT(ρ)

Let $\hat{X}_{i,t}$ be the empirical mean reward of arm i and time t.
Let K be the number of arms
Let $n_{i,t}$ be the discounted number of times arm i has been pulled at time t
Let $\mathrm{Var}(\mu_{i,t})$ be an upper confidence bound on the reward variance of arm i at time t.
Initialisation:
Play each arm once.
Main Loop:
for $t = K+1,\ldots,T$ **do**
  Let $U_i = \hat{X}_{i,t} + \sqrt{\frac{2\log\left(\min(1/4,\mathrm{Var}(\mu_{i,t}))\sum_{i=1}^{K} n_{i,t}\right)}{n_{i,t}}}$
  Pull arm $a_t = \mathrm{argmax}_i\, U_i$.
  $n_{a_t,t+1} = \rho n_{a_t,t} + 1$,
  $\hat{X}_{a_t,t+1} = \hat{X}_{a_t,t} + \frac{x_{a_t,t} - \hat{X}_{a_t,t}}{n_{a_t,t+1}}$.
  $n_{i,t+1} = \rho n_{i,t}$ for $i \neq a_t$,
  $\hat{X}_{i,t+1} = \hat{X}_{i,t}$ for $i \neq a_t$.
end for

concentration of measure, allows an agent to learn which arm is best reasonably quickly. A question one might ask is how well can an agent perform in a bandit environment when no such statistical assumptions are made? Auer et al. considered this case [8]. They consider that the rewards of each arm are generated by an adversary. In their work they use a general definition of regret. The regret is with respect to a specified series of actions. The regret for an agent is then the reward accumulated by the agent subtracted from the rewards accumulated by the reference sequence of actions. Let the sum of rewards accumulated by a

Algorithm 2.13 Pseudo-code for Adapt-EvE

Let K be the number of arms
Initialisation:
Initialise b_{curr} to an instance of UCB-Tuned with K arms.
Initialise $d_{\text{P-H}}$ to an instance of the Page-Hinkley change detector
Initialise isMetaBanditPhase = false.
Main Loop:
for $t = K+1, \ldots, T$ **do**
  if isMetaBanditPhase = true **then**
    Decrement count.
    Let a_{meta} be the arm chosen by b_{meta} (either b_{curr} or b_{new})
    if b_{curr} chosen by b_{meta} **then**
      Pull arm a_t according to b_{curr}.
      Update bandit b_{curr} (a_t,$x_{a_t}(t)$).
      Update bandit b_{meta} with pair (1,$x_{a_t}(t)$).
    else
      Pull arm a_t according to b_{new}.
      Update bandit b_{new} with pair (a_t,$x_{a_t}(t)$).
      Update bandit b_{meta} with pair (2,$x_{a_t}(t)$).
    end if
    if count = 0 **then**
      Set isMetaBanditPhase = false
      if Arm 2 of b_{meta} pulled most **then**
        Set $b_{\text{curr}} = b_{\text{new}}$
        Reinitialise $d_{\text{P-H}}$ to an instance of the Page-Hinkley change
      end if
    end if
  else
    Pull arm a_t according to b_{curr}.
    Update bandit b_{curr} and detector $d_{\text{P-H}}$ with pair (a_t,$x_{a_t}(t)$).
    if $d_{\text{P-H}}$ detects change **then**
      Set isMetaBanditPhase = true
      Initialise b_{new} to new instance of UCB-Tuned with K arms.
      Initialise b_{meta} to new instance of UCB-Tuned with 2 arms.
      Associate arm 1 of b_{meta} to b_{curr} and arm 2 to b_{new}.
      Set count = t_{meta}
    end if
  end if
end for

series of actions, $(j_1, \ldots, j_T)$, be denoted as $S_{(j_1,\ldots,j_T)}$, such that,

$$S_{(j_1,\ldots,j_T)} = \sum_{t=1}^{T} x_{j_t,t}. \tag{2.43}$$

For a given strategy A that chooses actions $i_1, \ldots, i_T$ up until time T, let the reward accumulated be denoted as,

$$S_A(T) = \sum_{t=1}^{T} x_{i_t,t}. \tag{2.44}$$

The expected regret felt with respect to a given sequence of actions $(j_1, \ldots, j_T)$ up to time T is then,

$$S_{(j_1,\ldots,j_T)} - \mathbb{E}\left[S_A(T)\right]. \tag{2.45}$$

Notice that due to the adversarial nature of the environment the expectation is now only over any stochastic processes internal to the agent's strategy A. Auer et al. produce bounds on this regret felt by some strategies. The bound obviously depends on the arbitrary sequence of actions, $(j_1, \ldots, j_T)$, the strategy is being compared against. They introduce a measure of *hardness* in order to rank sequences of actions. The harder a sequence of actions is to compete against the higher the measure is. The hardness measure, $H(j_1, \ldots, j_T)$, is defined as,

$$H(j_1, \ldots, j_T) = 1 + |\{1 \leq l < T : j_l \neq j_{l+1}\}|. \tag{2.46}$$

We can see how this hardness measure corresponds to how frequently the arbitrary sequence of actions, that we compare a strategy against, switches between a series of actions. In this way the regret defined in this section with a given hardness has a striking resemblance to the regret felt in a switching system with a given switching rate. We then might expect an algorithm designed for such a switching system to perform reasonably in an adversarial setting. It must be noted however that the switching environment still assumes a well-behaved stochastic process generates the rewards of arms in between switches, so it is unclear how penalised a strategy that assumes this will be when the assumption fails to hold. A family of algorithms designed for the adversarial bandit setting are those designed by Auer et al. called Exp3. The basic version of this is presented in Algorithm 2.14.

2.8 Best arm identification

The stochastic multi-armed bandit captures the inherit tradeoff in sequential decision making under uncertainty. We can think of two quantities the *cost* in

Algorithm 2.14 Exp3

Let $\gamma \in (0, 1]$.
Let K be the number of arms
Initialisation:
Set $w_i(1) = 1$ for $i = 1, \ldots, K$.
Main Loop:
for $t = 1, \ldots, T$ **do**
  Let $p_i(t) = (1 - \gamma)\frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\gamma}{K}$ for $i = 1, \ldots, K$.
  Pull arm a_t s.t. $P(a_t = i) = p_i(t)$ for $i = 1, \ldots, K$.
  Receive reward $x_{a_t} \in [0, 1]$.
  Let $\hat{x}_i(t) = \begin{cases} x_i(t)/p_i(t) & \text{if } i = a_t, \\ 0 & \text{otherwise,} \end{cases}$ for $i = 1, \ldots, K$.
  Let $w_i(t + 1) = w_i(t)e^{\gamma \hat{x}_i(t)/K}$.
end for

taking an action, and the *reward* in taking an action. In the multi-armed bandit problem these two quantities are tied to one another since we consider the cost of an action to be the regret the agent feels in having chosen it. The cost an agent pays in its exploration is traded off with the reward it receives in its exploitation. The tie between the cost of exploring and the reward of exploitation need not always be so intimately linked. Bubeck et al. considered a scenario where the cost of choosing an arm was tied to resources rather than being linked to the reward [13]. In their model the cost of actions dictate the length of time for which actions can be explored. The reward is confined to the reward received by the final decision at the end of exploration. The goal of the problem they describe is to maximise the expected reward of the final decision. Equivalently the goal is to minimise the expected *simple regret* of the final decision, where the simple regret is the difference between the mean reward of the chosen action and the mean reward of the best action (best meaning having highest mean). The problem can be thought of as a period of pure exploration, followed by one purely exploitative decision. An agent must then decide on an *allocation strategy* that is used to explore the performance of arms in the exploratory phase, and a *recommendation strategy* that is used to make the final decision. The problem is known as the *best arm identification* problem. There are objectives other than the simple regret that have been studied. One such objective is to identify an ϵ-best arm. An ϵ-best arm is any arm whose mean is a distance no further than ϵ away from the mean of the best arm.

2.8.1 Are multi-arm bandit algorithms good for best arm identification?

One might assume that a strategy that is good for the stochastic multi-armed bandit problem may also be good for the best arm identification problem. A strategy for the multi-armed bandit requires an agent to pull the best arm with larger and larger probability as time elapses. In this way we might expect the probability that we pull an arm far from optimal decreases rapidly and so too should the simple regret the agent feels in performing the strategy. Another way to think of this intuition is that an upper bound on the expected cumulative regret leads to upper bounds on the expected simple regret, which is certainly true. Bubeck et al.'s investigation of the best arm identification problem showed a less obvious connection between the expected cumulative regret and the expected simple regret [13]. The main result shows that an upper bound on the expected cumulative regret using a given allocation strategy leads to a lower bound on the expected simple regret. The theorem is as follows,

Theorem 2.8.1 (Bubeck 2009)**.** *For any pair of allocation and recommendation strategies and for any function* $\epsilon : \{1, 2, \dots\} \to \mathbb{R}$ *such that for all Bernoulli distributions* $\nu_1, \dots, \nu_K$ *on the rewards, there exists a constant* $C \geq 0$ *with* $\mathbb{E}[R_T] \leq C\epsilon(T)$*, for which it holds that: for all sets of* $K \geq 3$ *distinct Bernoulli distributions on the rewards that are different from* 1*, there exists an ordering of the arms' distributions,* $\nu_1, \dots, \nu_K$ *such that for some constant* $D \geq 0$*,*

$$\mathbb{E}\left[\mu_{\sigma(1)} - x_T\right] \geq \frac{\Delta}{2} e^{-D\epsilon(T)}, \tag{2.47}$$

where $\Delta = \min_{k:\mu_{\max}-\mu_k>0} \mu_{\max} - \mu_k$*, the smallest gap between the mean of the best arm and the mean of another suboptimal arm.*

As a consequence of this theorem any strategy that is asymptotically optimal for the stochastic multi-armed bandit problem (which means the cumulative regret is $O(\ln T)$) has a lower bound on the simple regret that is polynomial in the time of the exploratory phase. By considering a uniform strategy that allocates $\lfloor T/K \rfloor$ pulls to each arm, and using a simple Chernoff inequality concentration of measure argument, it can be shown that the expected simple regret using the uniform strategy is upper bounded by a term exponential in the exploratory time. This clearly shows that any asymptotically optimal multi-armed bandit strategy

is sub-optimal for the best arm identification problem. Intuitively, strategies designed for the multi-armed bandit problem pay a cost for exploring sub-optimal arms that strategies in the best arm identification problem do not pay. This forces the strategy to exploit the best arm in order to minimise this cost, whereas the exploitation reduces the opportunity to explore.

2.8.2 Quantifying problem difficulty

A bound on the error of not identifying the best arm is dependent on the setup of the problem. The closer the means of the best arm and other suboptimal arms, the longer it will take to sufficiently reduce the error. Audibert et al. introduced two measures of hardness in this setting. These are,

$$\mathcal{H}_1 = \sum_{k=1}^{K} \frac{1}{\Delta^2_{\sigma(k)}}, \tag{2.48}$$

and,

$$\mathcal{H}_2 = \max_{k \in \{1,\dots,K\}} \frac{k}{\Delta^2_{\sigma(k)}}, \tag{2.49}$$

where $\Delta_k = \mu_{\sigma(1)} - \mu_k$ denotes the distance between the largest expected reward and that of arm k. Audibert et al. show that these two quantities are within a logarithmic factor of one another. They further present the following theorem stated below.

Theorem 2.8.2 (Audibert 2010)**.** *Let $\nu_1, \dots, \nu_K$ be Bernoulli distributions with parameters in $[p, 1-p]$, $p \in (0, 1/2)$. For any forecaster, there exists a permutation $\sigma : \{1, \dots, K\} \to \{1, \dots, K\}$ such that the probability of error of the forecaster on the bandit problem defined by $\hat{\nu}_1 = \nu_{\sigma(1)}, \dots, \hat{\nu}_K = \nu_{\sigma(K)}$ satisfies*

$$P_e \geq \exp\left(-\frac{(5 + o(1))T}{p(1-p)\mathcal{H}_2}\right), \tag{2.50}$$

where the $o(1)$ term depends only on K, p and T and goes to 0 when T goes to infinity.

This theorem can be used to justify the use of $\mathcal{H}_2$ (and therefore $\mathcal{H}_1$) as a measure of hardness for a problem. This is because for any Bernoulli armed problem instance whose rewards have non-zero variance can be lower bounded by

a function of $\mathcal{H}_2$, and so this measure of hardness determines how well one can hope to do in solving the best arm identification problem.

A good strategy should have a sample complexity of order $\mathcal{H}_2$. By sample complexity we mean the order of the number of samples needed to have a good chance of identifying the best arm.

2.8.3 Race algorithms

Race algorithms are a class of algorithms suitable for the best arm identification problem. The basic algorithm makes use of upper and lower bound estimates of the average payoff of each arm. The strategy proceeds in rounds, with the initial round starting with all arms considered. At the end of each round the arms whose best average payoff (their upper bound estimate) are worse than the best arm's worst average payoff (its lower bound estimate) are eliminated from further consideration. The procedure continues until there is one remaining arm which is then considered the best arm. In this way, the arms can be considered to be in a "race" to become the candidate best arm, and thus the name of racing algorithms. The upper and lower bounds are based on concentration of measure results. A popular choice is Hoeffding bounds leading to Hoeffding racing [55], however Bernstein inequalities have also been used [48].

2.8.4 Gap algorithms

Gap-based algorithms use ideas from upper confidence bound based strategies. In upper confidence bound strategies the best arm is estimated not as a empirical mean of rewards received, but as an upper confidence estimate. This is an application of the principle of optimism under uncertainty. In the best arm identification problem Gabillon et al. argued that estimating the mean of the best arm is not the more salient information [21]. What is more crucial is estimating the difference between the mean of the best arm and second best arm, which they call the gap. In a similar way, that an UCB estimate for an arms mean is an optimistic estimate, the gap-based strategies produce optimistic estimates for what the gap is, and pull the least pulled arm associated with that estimate. Gabillon et al. called their approach *unified gap-based exploration* (UGapE). The meta-algorithm can be parametrised for two separate scenarios that were considered. The first when the exploration phase has a fixed known time period called

fixed budget. In this case the bound is on the performance measure of interest (for example the expected simple regret). The second when an agent has a predetermined level of risk they agree to incur and they explore until they have identified the best arm up to the corresponding confidence for the risk. This is called *fixed confidence*. The fixed confidence will equate to a given expected simple regret and so the bound is on the expected number of trials before this confidence is reached.

Hoffman et al. extended the idea of gap-based exploration and considered it from a Bayesian perspective [27]. The confidence intervals used in their proposed algorithm, BayesGap, are taken from Bayesian estimates rather than frequentist approaches as with Gabillon. In order to use confidence bounds on Bayesian models they generalise the procedure to cope with asymmetry of the bounds around the empirical mean. BayesGap is presented in Algorithm 2.15. The variant given assumes the rewards of the arms are Bernoulli distributed. This means that the arm beliefs are modelled as Beta distributions with parameters α and β.

Algorithm 2.15 BayesGap

Let $U_i(t, \gamma_{\text{bgap}})$ be the upper confidence of arm i at time t
Let $L_i(t, \gamma_{\text{bgap}})$ be the lower confidence of arm i at time t

Exploratory Phase:

for $t = 1, \ldots, T$ **do**
Let $B_k(t) = \max_{i \neq k} U_i(t, \gamma_{\text{bgap}}) - L_k(t, \gamma_{\text{bgap}})$
$s_k(t) = U_k(t, \gamma_{\text{bgap}}) - L_k(t, \gamma_{\text{bgap}})$
$J(t) = \arg\min_k B_k(t)$
$j(t) = \arg\min_{k \neq J(t)} U_k(t, \gamma_{\text{bgap}})$
Pull arm $a_t = \text{argmax}_{k \in \{j(t), J(t)\}} s_k(t)$.
Let $\alpha_{t+1,a_t} = \alpha_{t,a_t} + \mathbb{1}(x_{a_t}(t) = 1)$
$\beta_{t+1,a_t} = \beta_{t,a_t} + \mathbb{1}(x_{a_t}(t) = 0)$
Let $\alpha_{t+1,j} = \alpha_{t,j}$
$\beta_{t+1,j} = \beta_{t,j}$ for $j \in \{1, \ldots, K\} \setminus \{a_t\}$.
end for

Final Decision:

Let $\Psi(T) = J(\arg\min_{t \leq T} B_{J(t)}(t))$

2.8.5 Upper confidence bound algorithms

Bubeck et al. considered UCB-style algorithms and produced bounds on the simple regret for such a method assuming a given time horizon for the exploratory phase [13]. Audibert et al. also considered an UCB type algorithm they called UCB-E (Upper Confidence Bound Exploration) instead producing bounds on the probability of error [5]. The padding function, rather than being of the form $C\sqrt{\frac{\ln t}{n_{i,t}}}$ (where C is a constant, t the total pulls so far, and $n_{i,t}$ the number of pulls of arm i up to time t), is instead of the form $\sqrt{\frac{a}{n_{i,t}}}$ for some constant a. Since the padding function no longer contains a factor that grows with t, a period of inactivity in pulling an arm no longer increases the UCB estimate for that arm. This causes the algorithm to focus more on arms that have not been pulled, than with for example UCB1 and therefore increases the exploration in the algorithm. The larger the constant a the more the algorithm explores. The algorithm is described in Algorithm 2.16.

Audibert et al. show that the probability of error in identifying the best arm after T pulls, e_T is bounded from above as,

$$e_T \leq 2TKe^{-2a/25}, \tag{2.51}$$

for $0 < a \leq \frac{25}{36}\frac{T-K}{\mathcal{H}_1}$. This bound requires knowledge of $\mathcal{H}_1$, a measure of hardness for the problem discussed in section 2.8.2. This could be estimated online during the running of the algorithm, however Audibert et al. did not prove this and in fact believe that for the case of UCB-E such a strategy would often lie far away from the desired rate of convergence. However, they propose an alternative UCB-E algorithm for which they can use more appropriate online estimates of the hardness of the problem. They call this adaptive UCB-E.

2.8.6 Successive Rejects

Successive Rejects was proposed by Audibert et al. [5]. It resembles a race algorithm. The time horizon, T, of the exploratory phase is assumed to be known in advance. The exploration is split into $K - 1$ phases. The algorithm starts considering all arms, in the first phase the arms are pulled in a round-robin fashion an equal number of times. At the end of the phase, the worst performing arm is discarded and no longer pulled from then on in. The next phase continues with

Algorithm 2.16 UCB-E

Let $\hat{X}_{i,t}$ be the empirical mean reward of arm i and time t.
Let K be the number of arms
Let $n_{i,t}$ be the number of times arm i has been pulled at time t
Let $a > 0$.
Initialisation:
Play each arm once.
Main Loop:
for $t = K+1, \ldots, T$ **do**
  Let $U_i = \hat{X}_{i,t} + \sqrt{\frac{a}{n_{i,t}}}$
  Pull arm $a_t = \text{argmax}_i U_i$.
  Update $\hat{X}_{a_t,t+1}$
end for

the remaining arms being pulled round-robin until the end of the phase. The procedure is repeated until the algorithm is left with a single arm. The arm that remains is chosen as the recommendation in the final decision. Algorithm 2.17 describes the procedure. The length of the phases are carefully chosen through knowledge of T, to produce a bound on the probability of error that is optimal up to a logarithmic factor in K. This bound is,

$$P_e \leq \frac{K(K-1)}{2} \exp\left(-\frac{T-K}{\overline{\log}(K)\mathcal{H}_2}\right), \tag{2.52}$$

where $\overline{\log}(K) = \frac{1}{2} + \sum_{i=2}^{K} \frac{1}{i}$. The time horizon needs to be of a scale similar to $\overline{\log}(K)\mathcal{H}_2$ to identify the best arm, whereas an optimal solution would only need a time horizon with a scale of $\mathcal{H}_2$.

2.9 Summary

We have presented the core background knowledge that underpins the themes in this thesis. Firstly we have covered the basic mathematics required such as Bayesian modelling and some classic concentration of measure results. The aim of the thesis is to propose extensions to Thompson Sampling, a stochastic multi-armed bandit strategy, to problem settings beyond the standard model for which it was designed. To put the algorithm in context we have defined the stochastic multi-armed bandit problem, described competing strategies that have been proposed for the problem and covered some of the theoretical results known about

Algorithm 2.17 Successive Rejects

Let $\mathcal{K}_1 = \{1, \ldots, K\}$,
$\overline{\log}(K) = \frac{1}{2} + \sum_{i=2}^{K} \frac{1}{i}$,
$n_0 = 0$
Let $n_k = \lceil \frac{n-K}{\overline{\log}(K)(K+1-k)} \rceil$ for $k \in \{1, \ldots, K-1\}$.

Exploratory Phase:

for phase $k = 1, \ldots, K-1$ **do**
Pull arm i $n_k - n_{k-1}$ times, for $i \in \mathcal{K}_k$.
$\mathcal{K}_{k+1} = \mathcal{K}_k \setminus \{\arg\min_{i \in \mathcal{K}_k} \hat{X}_{i,n_k}\}$ (in case of a tie randomly select an arm to remove from set $\arg\min_{i \in \mathcal{K}_k} \hat{X}_{i,n_k}$)
end for

Final Decision:

Let $\Psi(T)$ be the remaining element in $\mathcal{K}_K$

the problem. The two problems for which we propose Thompson Sampling style algorithms are the non-stationary switching multi-armed bandit problems and the best arm identification problem. Again, known results for these two problem settings along with competing strategies for them were presented to further put the work in this thesis in a fuller context.

Chapter 3

Thompson Sampling in Multi-Armed Bandit Problems

This chapter introduces Thompson Sampling. The algorithm and the idea behind it form a unifying theme to the original contributions presented in later chapters. Thompson Sampling falls in the category of a probability matching algorithm, like Softmax, and is Bayesian in nature. The origins of Thompson Sampling stem from the procedure's inventor, William R. Thompson [69]. Thompson was interested in the general problem of research planning. He was concerned with being able to utilise a small numbers of observations in order to steer actions taken before more data could be collected. This was in the context of a perceived objection to argument based on small numbers of observations at the time. Thompson posed his problem in terms of deciding between two treatments in a clinical trial. One of the treatments would be administered to a patient in the trials population and the effect could be observed. These observations could then be incorporated into the decision-making process to improve the odds of the most effective treatment being administered to further patients.

Although an old technique, predating many popular approaches in the literature such as UCB, it had been surprisingly absent in the bandit and machine learning literature. However, it has recently seen something of a revival [63, 24, 15] and is now considered a state-of-the-art baseline bandit algorithm. It has been given several names including *posterior sampling*, *Bayesian sampling* and *Bayesian learning automaton*, however Thompson Sampling is more prevalent and so we will use this term to describe the procedure.

3.1 Thompson Sampling

Thompson Sampling, as previously mentioned, is a randomised probability matching strategy for the multi-armed bandit problem. We mean this in the sense that, for each decision, the probability of an arm being pulled matches the probability that the arm is in fact the optimal arm, given all past observations of arm pulls. The algorithm is Bayesian in that the probability that an arm is the best is a Bayesian estimate. Being more precise, let θ be the parameter vector defining behaviour of a particular multi-armed bandit problem. The probability that an arm k is optimal (it is equal to the optimal arm k^*) is,

$$P(k = k^*) = \int_\theta \mathbb{1}(k = k^*|\theta)P(\theta)d\theta. \tag{3.1}$$

An arm is thus pulled with the probability $P(k = k^*)$. The algorithm can be viewed as forming a decision based on a one-step Monte-Carlo sample estimate of the probability of an arm being the best.

The integral above may not have a closed form solution. The integral may be approximated by quadrature. However, this is undesirable as the running cost of each decision will depend on the difficulty of the multi-armed bandit problem. The insight used in Thompson Sampling is that instead we can sample from a distribution defined by $P(k = k^*)$ by simply first sampling a candidate θ from the distribution $P(\theta)$. Given a candidate θ we can then just pull the arm that is best given this candidate ($\mathbb{1}(k = k^*|\theta)$). $P(\theta)$ is initially specified as a prior and then later inferred using Bayes rule as rewards from arm pulls are observed. The general Thompson Sampling algorithm for a K-armed bandit is therefore described by the following pseudo-code.

Initialise $P(\mu_1, \ldots, \mu_K)$, the prior belief of the mean payoffs of arms $1, \ldots, K$.
Let H_t be the history of action,reward pairs (a_τ, x_τ) for $1 \leq \tau \leq t$.
Initialise $H_1 = \{\}$.

for $t = 1, \ldots, T$ **do**
  Sample $\theta_i, \ldots, \theta_K \sim P(\mu_1, \ldots, \mu_K|H_t)$.
  Pull arm $a_t = \text{argmax}_i\, \theta_i$
  Receive reward x_t
  Let $H_{t+1} = H_t \cup (a_t, x_t)$.

end for

Thompson Sampling for Bernoulli-armed bandit problems

For much of the work in this thesis we will, for simplicity, restrict ourselves to considering multi-armed bandit problems with Bernoulli rewards (the reward received is either a zero or a one). We will thus derive the Thompson Sampling algorithm for this case (the most common in the literature). For the multi-armed bandit with Bernoulli rewards there are a finite set of arms $k \in 1, \ldots, K$. The payoffs for arm k are distributed Bernoulli with parameter μ_k, the expected reward of the arm. The bandit problem can thus be specified by the set of parameters $\{\mu_i : i = 1, \ldots, K\}$, therefore $\theta = (\mu_1, \ldots, \mu_K)$. The probability of an arm is now defined as

$$P(k = k*) = \int_{\mu_1} \ldots \int_{\mu_K} \mathbb{1}(k = k^* | \mu_1, \ldots, \mu_K) P(\mu_1, \ldots, \mu_K) d\mu_1 \ldots d\mu_K. \quad (3.2)$$

The arms are independent and so we can write this integral as

$$P(k = k*) = \int_{\mu_1} \ldots \int_{\mu_K} \mathbb{1}(k = k^* | \mu_1, \ldots, \mu_K) P(\mu_1) \ldots P(\mu_K) d\mu_1 \ldots d\mu_K. \quad (3.3)$$

We can therefore infer the distributions, $P(\mu_1), \ldots, P(\mu_K)$, separately. When making a decision a sample is drawn from each $P(\mu_k)$ to estimate the true mean μ_k of arm k. The arm with highest corresponding estimate is then pulled. When the arms are Bernoulli it makes sense to model $P(\mu_k)$ as a Beta distribution. This is because the Beta distribution is a conjugate prior to the Bernoulli distribution. When we pull arm k and receive a reward, x, then need to update the belief $P(\mu_k)$ (i.e. infer $P(\mu_k|x)$). Since $P(\mu_k)$ is a Beta distribution, which in conjugate to the Bernoulli distribution, then $P(\mu_k|x)$ is also a Beta distribution. This makes the algorithm incredibly efficient as for the case of a Beta distribution only the counts of number of successes ($x = 1$) and number of failures ($x = 0$) for each arm need to be stored. Algorithm 3.1 presents the Thompson Sampling algorithm for the stationary Bernoulli multi-armed bandit problem and Figure 3.1 shows the evolution of the posterior Beta distributions as the strategy is performed on a two armed bandit problem.

Of course a similar derivation could be performed for any bandit problem whose reward distribution had a corresponding conjugate prior. For example an

Algorithm 3.1 Thompson Sampling for Bernoulli Bandits

Let $\alpha_{1,k} = 1$,
$\beta_{1,k} = 1$ for $k \in \{1, \ldots, K\}$.

for $t = 1, \ldots, T$ **do**
Sample $\theta_i \sim \text{Beta}(\alpha_{t,i}, \beta_{t,i})$, for $i \in \{1, \ldots, K\}$..
Pull arm $a_t = \text{argmax}_i\, \theta_i$
Let $\alpha_{t+1,a_t} = \alpha_{t,a_t} + \mathbb{1}(x_{a_t}(t) = 1)$
$\beta_{t+1,a_t} = \beta_{t,a_t} + \mathbb{1}(x_{a_t}(t) = 0)$
Let $\alpha_{t+1,j} = \alpha_{t,j}$
$\beta_{t+1,j} = \beta_{t,j}$ for $j \in \{1, \ldots, K\} \setminus \{a_t\}$.
end for

alternative would be if the reward of the arms were distributed as Gaussian (Normally) with a known variance but unknown mean. The corresponding conjugate prior from which we would sample from (and whose hyperparameters would be stored and updated via observations) would also be a Normal distribution.

Normal Thompson Sampling in the Bernoulli-armed bandit

As we have stated, other conjugate prior pairs can be used other than the Bernoulli-Beta pair. The Bernoulli-Beta pair is picked because the reward for each arm is assumed to be a Bernoulli random variable. However it is useful to investigate how the performance of a Thompson Sampling strategy is affected when the underlying assumptions are wrong. In order to see this an experiment is performed. The bandit algorithm is tested on a Bernoulli-armed bandit, however a Thompson Sampling algorithm that assumes that the rewards are distributed as a Gaussian with known variance is used instead. The strategy is described in Algorithm 3.2. The known variance of the reward is set to 1/4 (the maximum variance of a Bernoulli trial), the mean of the prior to 1/2 and the variance of the prior to 1. Figure 3.2 shows the results of three different problem instances. We can see that surprisingly the Normal Thompson Sampling strategy outperforms the Beta Thompson Sampling strategy on these three problems. It is unclear as to whether this is true more broadly.

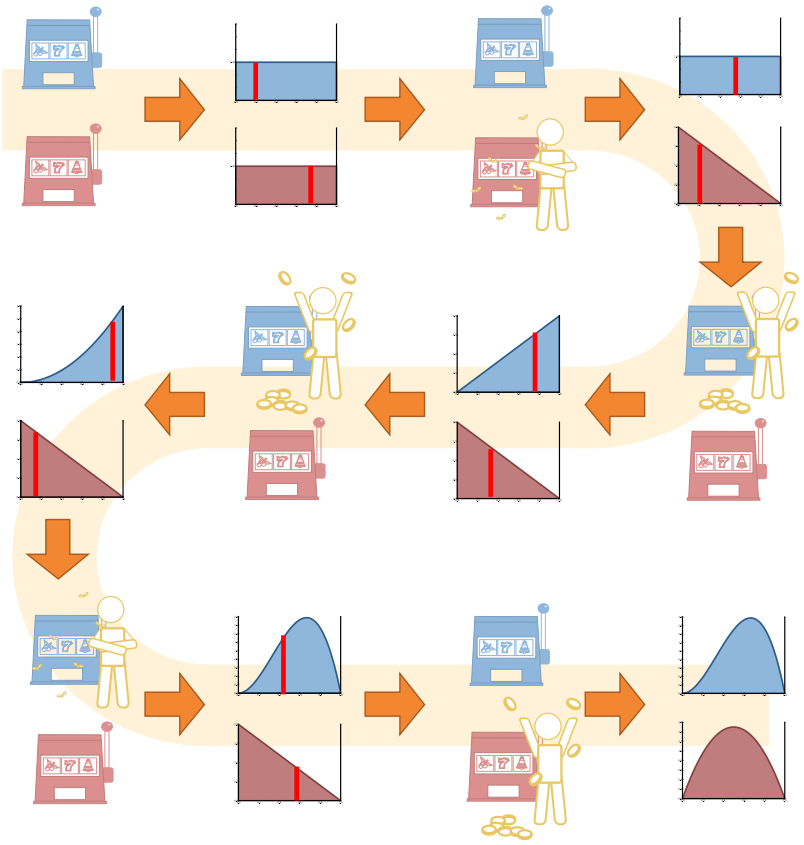


Figure 3.1: The evolution of the posterior Beta distributions for a two armed Bernoulli bandit problem. The agent starts with no knowledge of the arm means and so the distributions are uniform (Beta(1, 1)). As more observations are made the distributions become more concentrated so that it becomes more likely that a sample estimate is draw close to the true mean of a given arm.

3.1.1 Perceived advantages

Computationally efficient

For many of the standard multi-armed bandit problems, including the stochastic Bernoulli multi-armed bandit problem, the relevant posterior distribution that models the expected arm payoffs can be modelled in closed form. Due to conjugate priors updates to beliefs lead to simple update rules to a small fixed set of parameters.

Algorithm 3.2 Thompson Sampling for Normal Bandits (with known variance σ^2)

Let $\mu_{1,k} = 0$,
$s^2_{1,k} = 1$ for $k \in \{1, \ldots, K\}$.

for $t = 1, \ldots, T$ **do**
Sample $\theta_i \sim \mathcal{N}(\mu_{t,i}, s^2_{t,i})$, for $i \in \{1, \ldots, K\}$..
Pull arm $a_t = \text{argmax}_i\, \theta_i$
Let $\mu_{t+1,a_t} = \frac{\mu_{t,a_t}\sigma^2 + x_{a_t} s^2_{t,a_t}}{\sigma^2 + s^2_{t,a_t}}$
$s^2_{t+1,a_t} = \frac{s^2_{t,a_t}\sigma^2}{s^2_{t,a_t} + \sigma^2}$
Let $\mu_{t+1,j} = \mu_{t,j}$
$s^2_{t+1,j} = s^2_{t,j}$ for $j \in \{1, \ldots, K\} \setminus \{a_t\}$.
end for

A principle for decision making under uncertainty

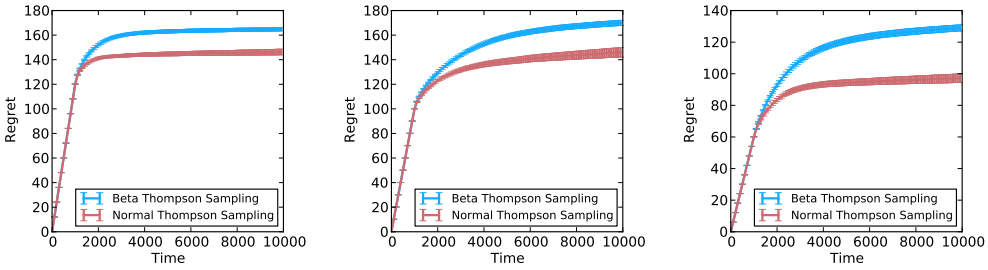
Like UCB algorithms use the principle of "optimism under uncertainty" to motivate a class of algorithm, Thompson Sampling too provides a principle in which to tackle decision making under uncertainty. The principle is to model the uncertainty directly using Bayesian methods, and to manage exploration by adaptively producing estimates that reflect this uncertainty via random sampling. This thesis is focused on extending the general principle to decision making models beyond the multi-armed bandit problem.

Applicable to complicated bandit problems

Bandit problems that have arms with interdependence, or arms with complicated reward distributions can also be tackled straightforwardly with Thompson Sampling. Even if distributions can not be represented exactly in closed form, Bayesian modelling techniques such as Markov Chain Monte Carlo, variational techniques or particle filtering can be applied to form the posterior distributions necessary from which to sample from.

Resilience to delayed rewards

In the multi-armed bandit problem we assume that the reward associated with a given action is observed instantaneously by the agent. In some contexts this is unrealistic. An agent may have to wait a period of time before learning the



(a) 5-armed bandit with arm means of 0.76,0.7,0.64,0.6 and 0.5

(b) 5-armed bandit with arm means of 0.55,0.5,0.5,0.4 and 0.3

(c) 5-armed bandit with arm means of 0.3,0.25,0.25,0.2 and 0.2

Figure 3.2: Three plots showing the expected cumulative regret for Beta Thompson Sampling and Normal Thompson Sampling for a Bernoulli-armed bandit. Surprisingly the results show that the Normal Thompson Sampling strategy outperforms the Beta Thompson Sampling despite the conjugate prior being more appropriate for a Bernoulli reward distribution in some sense.

effect of their decision. In the intervening time the agent can not simply stop the world and wait for the information to arrive. The rewards are therefore delayed. The agent still needs to interact with the environment they are in and make further decisions. One claimed advantage of Thompson Sampling by Chapelle and Li is that it should be resilient to delayed rewards [15]. The reason for this is down to its randomised nature. Since each action is randomised the strategy will continue to explore actions regardless of receiving no new information. This appears to be the case, however, of itself it is not a particularly strong claim about the algorithm. Many algorithms can also claim this advantage, including UCB. For instance Joulani et al. showed that the regret of UCB in a environment with delayed feedback is only greater than that of the undelayed case by an additive term linear in the expected largest number of unseen rewards [31]. Their paper also summarises results for different delayed scenarios showing that for the adversarial setting the upper-bound on reward for the delayed setting is different to the non-delayed case by a *multiplicative* term linear in the delay. It is less clear if a randomised approach based on Thompson Sampling will behave better than an approach based on UCB in this setting.

3.1.2 Existing theoretical analysis

Compelling empirical evidence had lead to a resurged interest in Thompson Sampling as a decision-making strategy. Initially only weak theoretical results on Thompson Sampling were produced. May et al. for instance showed that the Thompson Sampling strategy was greedy in the limit with infinite exploration (GLIE) [53]. However frequentist methods such as UCB had enjoyed much firmer theoretical grounding. Variants of UCB had strong upper bounds on their expected cumulative regret. These upper bounds were shown to match the lower bounds derived by Lai and Robbins for the stochastic multi-armed bandit problem [41]. This lack of analysis, on an equal footing to other bandit algorithms, led to call for the discovery of tight upper bounds on regret for Thompson Sampling as an important open problem in the machine learning community [44]. Agrawal et al. provided the first serious attack on a regret bound but it was not optimal in the sense of matching the bounds of Lai and Robbins [2]. In the same year as the open problem was announced, Kaufman et al. did show that Thompson Sampling was asymptotically optimal for the Bernoulli stochastic multi-armed bandit problem [34]. This was later improved by Agrawal et al. with asymptotically optimal problem-independent regret bounds [3]. This answered positively the open problem and showed that Thompson Sampling, a Bayesian algorithm, was efficient and could compete against other strategies even when measured by frequentist performance measures. The results show that the Thompson Sampling problem-dependent bound for the K-armed stochastic bandit problem is,

$$\mathbb{E}\left[R_T\right] \leq (1+\epsilon)\sum_{i=2}^{K}\frac{\ln T}{d(\mu_i,\mu^*)}\Delta_i + O\left(\frac{K}{\epsilon^2}\right), \tag{3.4}$$

where $d(\mu_i,\mu^*)$ is the KL divergence between the mean of arm i and the best arm. The problem-independent bound is such that for any problem the regret is bounded as,

$$\mathbb{E}\left[R_T\right] \leq O(\sqrt{KT\ln T}). \tag{3.5}$$

Later Korda et al. showed that Thompson Sampling achieves the lower bound on regret for all one-dimensional exponential family bandits [38]. The result assumes that the prior distribution is chosen as the Jeffreys prior, an uninformative prior. This puts the Thompson Sampling strategy on an equal theoretical standing to state of the art upper confidence bound strategies such as KL-UCB.

Other lines of theoretical investigation have extended this work. Russo and Roy later showed that there is a connection between the expected cumulative regret bound of a UCB algorithm and the upper bound on *Bayes Risk* for Thompson Sampling. For the finite armed Bernoulli bandit the Bayes Risk is defined as,

$$\text{BayesRisk}(T) = \sum_{t=1}^{T} \mathbb{E}\left[\max_{k \in \mathcal{K}} \mu_k - \mu_{a_t}\right]. \tag{3.6}$$

The expectation is taken over the prior distribution over the arm means $\mu_1, \ldots, \mu_K$ of the problem, unlike the regret which is defined with respect to known arm means. This allows one to easily take advantage of existing bounds found for the regret of UCB algorithms and apply them to Thompson Sampling. One big advantage of this approach is that bounds can be produced for bandit problems where the arms are interdependent and also for the contextual bandit problem. The bounds they produce depend on the amount of dependence amongst the arms and this is captured by a notion referred to as *margin dimension* [62]. Li argues that a downside to this approach is its reliance on a correct prior from which to take the expectation [43]. Instead Li takes a different approach. By noticing connection between Thompson Sampling and exponentiated updates Li generalises Thompson Sampling to the expert-learning framework [43].

3.2 Optimism in Thompson Sampling

A question that arises is what is the tradeoff between exploration and exploitation that Thompson Sampling makes. May et al. [53] tried to separate the two aspects of the algorithm. To do this they defined the *exploitative value* of an action to be the expected payoff of an action conditioned on the rewards seen to date by the agent. The estimated value of an action could be seen as the value of the sample drawn from the posterior distribution for the action. With these the exploratory value of an action could then by found by subtracting the exploitative value from the estimated sample value. They observed that this exploratory value could sometimes be negative and so there would be no value from an exploration point of view to pull the arm. The exploratory value of an action is only negative when the sample estimate drawn from the posterior is less than the exploitative value of the arm (pictured in Figure 3.3). This corresponds to a sample drawn from the left tail of the posterior distribution. They reasoned that no advantage was

to be gained by sampling from the left-hand tail of the arm distributions when performing Thompson Sampling, and proposed Optimistic Thompson Sampling. In Thompson Sampling samples are drawn from the posterior distributions of each arm, that is $\theta_i(t) \sim P(\mu_i)$. Instead Optimistic Thompson Sampling draws samples such that $\theta_i(t) = \max(\mathbb{E}[\mu_i], s_i(t))$ where $s_i(t) \sim P(\mu_i)$. In other words if a sample from the posterior distribution is less than the mean of the distribution, then we take the sample to be the mean. This ensures that the exploratory value of an action is always positive. Algorithm 3.3 more formally presents the algorithm specifically for the Bernoulli bandit. May et al. observed empirically that Optimistic Thompson Sampling performed (in some cases much) better than standard Thompson Sampling.

Algorithm 3.3 Optimistic Thompson Sampling for Bernoulli Bandits

Let $\alpha_{1,k} = 1$,
$\beta_{1,k} = 1$ for $k \in \{1, \ldots, K\}$.

for $t = 1, \ldots, T$ **do**
Sample $\theta_i \sim \text{Beta}(\alpha_{t,i}, \beta_{t,i})$, for $i \in \{1, \ldots, K\}$..
Pull arm $a_t = \text{argmax}_i \max\left(\theta_i, \frac{\alpha_{t,i}}{\alpha_{t,i}+\beta_{t,i}}\right)$
Let $\alpha_{t+1,a_t} = \alpha_{t,a_t} + \mathbb{1}(x_{a_t}(t) = 1)$
$\beta_{t+1,a_t} = \beta_{t,a_t} + \mathbb{1}(x_{a_t}(t) = 0)$
Let $\alpha_{t+1,j} = \alpha_{t,j}$
$\beta_{t+1,j} = \beta_{t,j}$ for $j \in \{1, \ldots, K\} \setminus \{a_t\}$.
end for

Here we adapt the proof of Agrawal and Goyal [3] to produce a regret bound for Optimistic Thompson Sampling. Since asymptotically Thompson Sampling attains the lower bound of regret for the multi-armed bandit problem then we can not expect any asymptotic improvement. However, the modification to the proof does remove some constant terms in the bound. This corroborates the findings of May, Korda, Lee, and Leslie [53].

3.2.1 Optimistic Thompson Sampling

For this section we switch notation to be the same as that used by Agrawal and Goyal, to allow easier comparison with their proof. The algorithm remains the same as stated in Algorithm 3.3. Let $k_i(t)$ be the number of plays of arm i. Let

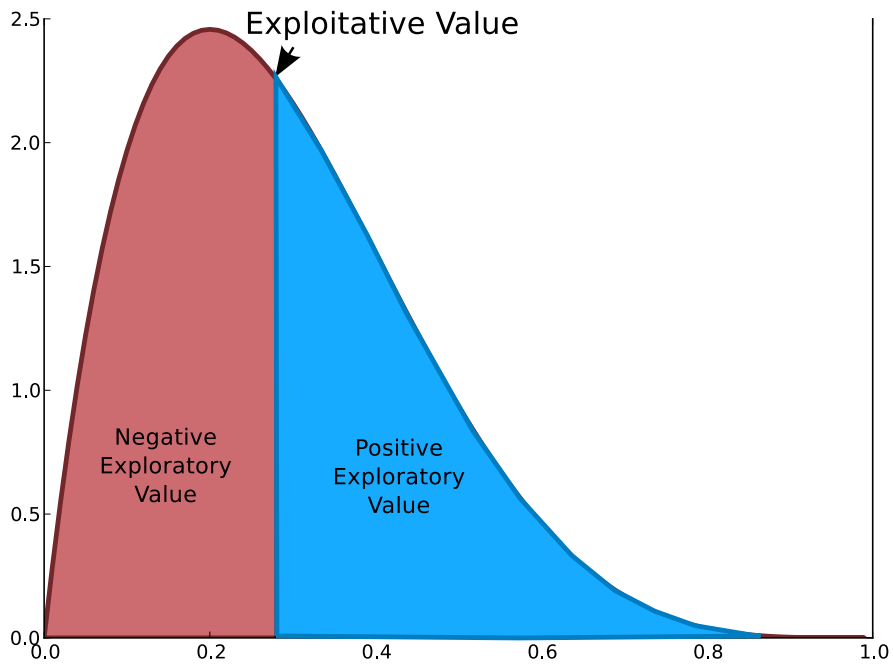


Figure 3.3: This image shows the area of the posterior distribution for an arm (in this case distributed Beta$(2, 5)$) from which a sample can be drawn to give a negative exploratory value and the region that corresponds to a positive exploratory value.

$i(t)$ denote the arm played at time t and with out loss of generality let arm 1 be the optimal arm. Let $S_i(t)$ be the number of successes of arm i.

Let μ_i be the expected payoff of arm i, and $\hat{\mu}_i(t)$ be the estimated mean of the Beta distribution the algorithm models arm i with (That is $\hat{\mu}_i(t) = (S_i(t) + 1)/(k_i(t) + 1)$). Define constants x_i and y_i such that $\mu_i < x_i < y_i < \mu_1$. Let $\theta_i(t)$ be the sample generated from the Beta distribution for the i^{th} arm at time t.

Let $E_i^{\mu}(t)$ be the event that $\hat{\mu}_i(t) \leq x_i$, and $E_i^{\theta}(t)$ be the event that $\theta_i(t) < y_i$.

The Optimistic Thompson Sampling algorithm can be described as follows,

1. For all arms i, sample $\theta_i(t) \sim \text{Beta}(1 + S_i(t), 1 + k_i(t) - S_i(t))$
2. Pick the arm a such that $a = \text{argmax}_i \max\left(\theta_i(t), \frac{\alpha_i(t)}{\alpha_i(t)+\beta_i(t)}\right)$
3. Receive reward $X_a(t) \in \{0, 1\}$
4. Update arms hyperparameters, $S_a(t + 1) = S_a(t) + X_a(t)$ and $k_a(t + 1) = k_a(t) + 1$

5. For unplayed arms j update hyperparameters, $S_j(t+1) = S_j(t)$ and $k_a(t+1) = k_a(t)$

6. Repeat

3.2.2 Proof sketch

In this section we outline the steps to bound the expected regret of Optimistic Thompson Sampling. The proof techniques used below are all the same as those that Agrawal and Goyal apply to ordinary Thompson Sampling. The only contribution here is, for completeness, to apply their techniques to the Optimistic Thompson Sampling case. As with their proof, to bound the regret we just need to bound the expectation of pulling a suboptimal arm, i,

$$\mathbb{E}\left[k_i(T)\right] = \sum_{t=1}^{T} \Pr(i(t){=}i). \tag{3.7}$$

This is because the expected regret can be written in terms of the number of times each suboptimal arm is pulled, as follows,

$$\mathbb{E}\left[R_T\right] = \sum_{k=2}^{K} \Delta_k \mathbb{E}\left[k_i(T)\right] \tag{3.8}$$

The decomposition is the same as that used by Agrawal and Goyal [3],

$$\sum_{t=1}^{T} \Pr(i(t) = i) = \sum_{t=1}^{T} \overbrace{\Pr(i(t) = i, E_i^{\mu}(t), E_i^{\theta}(t))}^{\text{A: well estimated AND sampled}} \tag{3.9}$$

$$+ \sum_{t=1}^{T} \overbrace{\Pr(i(t) = i, E_i^{\mu}(t), \overline{E_i^{\theta}(t)})}^{\text{B: well estimated, high sample}} \tag{3.10}$$

$$+ \sum_{t=1}^{T} \overbrace{\Pr(i(t) = i, \overline{E_i^{\mu}(t)})}^{\text{C: poorly estimated}}. \tag{3.11}$$

Here $E_i^{\mu}(t)$ is the event that $\hat{\mu}_i < x_i$ at time t ($\hat{\mu}_i$ being the empirical mean of arm i) and $E_i^{\theta}(t)$ is the event that $\theta_i < y_i$ (θ_i being the sample estimate of arm i). Remembering x_i and y_i are defined such that $\mu_i < x_i < y_i < \mu_1$.

We will then consider each of the terms separately.

Term A

Term A, the probability that the sub-optimal arm is pulled when well estimated and well sampled, is bounded by an expression in terms of the optimal arm being sampled sufficiently highly. Let $P_i(t) = \Pr(i(t)=i, E_i^{\mu}(t), E_i^{\theta}(t)|\mathcal{F}_{t-1})$, where $\mathcal{F}_{t-1}$ is the filtration for the history of action-reward pairs until time $t-1$. Lemma 3.2.1 is used to bound this term.

Lemma 3.2.1. *For all $t \in [1, T]$, and $i \neq 1$,*

$$P_i(t) \leq \frac{(1-p_{i,t})}{p_{i,t}} \Pr\left(i(t)=1, E_i^{\mu}(t), E_i^{\theta}(t)|\mathcal{F}_{t-1}\right),$$

where $p_{i,t} = \Pr(\theta_1(t) > y_i|\mathcal{F}_{t-1})$.

Proof. Either $\mathcal{F}_{t-1}$ is such that $E_i^{\mu}(t)$ is true, or it is not. In the case where it is not the the bound is trivially true since in that case $P_i(t) = 0$. For the case where $\mathcal{F}_{t-1}$ is such that $E_i^{\mu}(t)$ is true it suffices to prove that,

$$\Pr\left(i(t)=i|E_i^{\theta}(t), \mathcal{F}_{t-1}\right) \leq \frac{(1-p_{i,t})}{p_{i,t}} \Pr\left(i(t)=1|E_i^{\theta}(t), \mathcal{F}_{t-1}\right). \tag{3.12}$$

Let $M_i(t)$ be the event that $\theta_i(t) \geq \theta_j(t), \forall j \neq 1$ (The event that the sampled estimate of arm i exceeds that of all other suboptimal arms at time t). Equation 3.12 holds if the following two inequalities hold,

$$\Pr\left(i(t)=1|E_i^{\theta}(t), \mathcal{F}_{t-1}\right) \geq p_{i,t} \Pr\left(M_i(t)|E_i^{\theta}(t), \mathcal{F}_{t-1}\right), \tag{3.13}$$

$$\Pr\left(i(t)=i|E_i^{\theta}(t), \mathcal{F}_{t-1}\right) \leq (1-p_{i,t}) \Pr\left(M_i(t)|E_i^{\theta}(t), \mathcal{F}_{t-1}\right). \tag{3.14}$$

Starting with the left hand side of Equation 3.13 we have,

$$\Pr\left(i(t)=1|E_i^{\theta}(t), \mathcal{F}_{t-1}\right) \geq \Pr\left(i(t)=1, M_i(t)|E_i^{\theta}(t), \mathcal{F}_{t-1}\right), \tag{3.15}$$

$$= \Pr\left(M_i(t)|E_i^{\theta}(t), \mathcal{F}_{t-1}\right) \Pr\left(i(t)=1|M_i(t), E_i^{\theta}(t), \mathcal{F}_{t-1}\right). \tag{3.16}$$

Since events $M_i(t)$ and $E_i^{\theta}(t)$ occur, it holds that for all $j \neq i$ and $j \neq 1$ that

$\theta_j(t) \leq \theta_i(t) \leq y_i$. It then follows that,

$$\Pr\left(i(t)=1|M_i(t), E_i^{\theta_i}(t), \mathcal{F}_{t-1}\right) \geq \Pr\left(\theta_1(t) > y_i|M_i(t), E_i^{\theta}(t), \mathcal{F}_{t-1}\right), \quad (3.17)$$

$$= \Pr\left(\theta_1(t) > y_i|\mathcal{F}_{t-1}\right), \quad (3.18)$$

$$= p_{i,t}. \quad (3.19)$$

Equation 3.18 uses the fact that, conditioned on $\mathcal{F}_{t-1}$, $\theta_1(t)$ is independent of $M_i(t)$ and $E_i^{\theta}(t)$. This combined with Equation 3.16 leads to the inequality in Equation 3.13.

From the definition of $E_i^{\theta}(t)$ the joint event $E_i^{\theta}(t), i(t)$ only occurs if $\theta_1(t) < y_i$. Therefore,

$$\Pr\left(i(t)=i|E_i^{\theta}(t), \mathcal{F}_{t-1}\right) \leq \Pr\left(\theta_1(t) \leq y_i, \theta_i \geq \theta_j(t), \forall j \neq 1|E_i^{\theta}(t), \mathcal{F}_{t-1}\right), \quad (3.20)$$

$$= \Pr\left(\theta_1(t) \leq y_i|\mathcal{F}_{t-1}\right) \Pr\left(\theta_i(t) \geq \theta_j(t), \forall j \neq 1|E_i^{\theta}(t), \mathcal{F}_{t-1}\right), \quad (3.21)$$

$$= (1-p_{i,t}) \Pr\left(M_i(t)|E_i^{\theta}(t), \mathcal{F}_{t-1}\right), \quad (3.22)$$

which proves the inequality in Equation 3.14. □

Letting $\Gamma_t = \frac{(1-p_{i,t})}{p_{i,t}}$, the proof by Agrawal and Goyal [3] proceeds using this bound as follows,

$$\sum_{t=1}^{T} P_i(t) \leq \sum_{t=1}^{T} \Gamma_t \Pr(i(t)=1, E_i^{\mu}(t), E_i^{\theta}(t)|\mathcal{F}_{t-1}) \quad (3.23)$$

$$= \sum_{t=1}^{T} \mathbb{E}\left[\Gamma_t \mathbb{I}(i(t)=1, E_i^{\theta}(t), E_i^{\mu}(t))\right] \quad (3.24)$$

$$= \sum_{k=0}^{T} \mathbb{E}\left[\Gamma_{\tau_k+1} \sum_{t=\tau_k+1}^{\tau_{k+1}} \mathbb{I}(i(t)=1)\right], \quad (3.25)$$

where τ_k denotes the k^{th} time the optimal arm was pulled. The last line (Equation 3.25) uses the fact that $p_{i,t}$ only changes when the optimal arm is pulled.

Agrawal and Goyal [3] bound the term $\sum_{k=1}^{T-1} \mathbb{E}\left[\frac{1}{p_{i,\tau_k+1}} - 1\right]$ with their Lemma

4. The proof proceeds as follows,

$$\sum_{k=1}^{T-1} \mathbb{E}\left[\frac{1}{p_{i,\tau_k+1}} - 1\right] \tag{3.26}$$

$$\leq \frac{24}{(\mu_1 - y_i)^2} + \sum_{j=0}^{T-1} \Theta\left(e^{\frac{-(\mu_1-y_i)^2 j}{2}} + \frac{e^{-D(\mu_1,y_i)j}}{(j+1)(\mu_1-y_i)^2} + \frac{1}{e^{\frac{(\mu_1-y_i)^2 j}{4}} - 1}\right), \tag{3.27}$$

where $D(\mu_1, y_i) = y_i \log \frac{y_i}{\mu_1} + (1 - y_i) \log \frac{1-y_i}{1-\mu_1}$. The expression defining $D(\mu_1, y_i)$ is equivalent to the KL divergence between two Bernoulli distributions, one with parameter μ_1 and the other with parameter y_i.

The above bound is derived by first bounding

$$\mathbb{E}\left[\frac{1}{p_{i,\tau_k+1}}\right] = \sum_{s=0}^{j} \frac{f_{j,\mu_1}(s)}{F_{j+1,y_i}(s)}, \tag{3.28}$$

where $f_{j,\mu_1}(s)$ is the pdf of a binomial and $F_{j,y_i}(s)$ is a cdf of a binomial.

The sum in Equation 3.28 can be split in to several sums as follows,

$$\sum_{s=0}^{j} \frac{f_{j,\mu_1}(s)}{F_{j+1,y_i}(s)} = \sum_{0}^{\lfloor y_i j \rfloor - 1} \frac{f_{j,\mu_1}(s)}{F_{j+1,y_i}(s)} + \sum_{\lfloor y_i j \rfloor}^{\lfloor y_i j \rfloor} \tag{3.29}$$

$$+ \sum_{\lceil y_i j \rceil}^{\lfloor \mu_1 j - \frac{(\mu_1 - y_i)}{2} j \rfloor} \frac{f_{j,\mu_1}(s)}{F_{j+1,y_i}(s)} + \sum_{\lceil \mu_1 j - \frac{\mu_1 - y_i}{2} j \rceil}^{j} \frac{f_{j,\mu_1}(s)}{F_{j+1,y_i}(s)}. \tag{3.30}$$

For regular Thompson Sampling, Agrawal et al. bound these sums as follows,

$$\text{Sum}\left(0, \lfloor y_i j \rfloor - 1\right) \leq \Theta\left(e^{-Dj} \frac{1}{j+1} \frac{1}{(\mu_1 - y_i)^2}\right) + \Theta\left(e^{-2(\mu_1-y_i)^2 j}\right), \tag{3.31}$$

$$\text{Sum}\left(\lfloor y_i j \rfloor, \lfloor y_i j \rfloor\right) \leq 3e^{-Dj}, \tag{3.32}$$

$$\text{Sum}\left(\lceil y_i j \rceil, \lfloor \mu_1 j - \frac{(\mu_1 - y_i)}{2} j \rfloor\right) \leq \Theta\left(e^{-\frac{(\mu_1-y_i)^2 j}{2}}\right), \tag{3.33}$$

$$\text{Sum}\left(\lceil \mu_1 j - \frac{\mu_1 - y_i}{2} j \rceil, j\right) \leq 1 + \frac{1}{e^{\frac{(\mu_1-y_i)^2 j}{4}} - 1}. \tag{3.34}$$

At this point we can reduce the additive components in terms of $\mu_1 - y_i$ from the bound due to how Optimistic Thompson Sampling behaves. We know that

in Optimistic Thompson Sampling a sample for arm i is certain to exceed y_i as long as the mean of the Beta distribution for arm i exceeds this value. This means that if $s > y_i j$ then $p_{i,\tau_j+1} = 1$, the consequence of this is that the sums $\text{Sum}(\lceil y_i j \rceil, \lfloor \mu_1 j - \frac{(\mu_1 - y_i)}{2} j \rfloor)$ and $\text{Sum}(\lceil \mu_1 j - \frac{\mu_1 - y_i}{2} j \rceil, j)$ can be discarded. The bound for $\text{Sum}(\lceil \mu_1 j - \frac{\mu_1 - y_i}{2} j \rceil, j)$ in the proof by Agrawal and Goyal [3] required that $j \geq 8/(\mu_1 - y_i)$, the case for $j < 8/(\mu_1 - y_i)$ treated separately, leading to additive term $24/(\mu_1 - y_i)^2$. The bound for Optimistic Thompson Sampling therefore does not contain this additive constant. Therefore for Optimistic Thompson Sampling the following lemma holds,

Lemma 3.2.2. *Let τ_j be the time at which the jth trial of the optimal arm happens, then,*

$$\mathbb{E}\left[\frac{1}{p_{i,\tau_j+1}}\right] \leq 1 + \Theta\left(\exp\left(\frac{-(\mu_1 - y_i)^2 j}{2}\right) + \frac{\exp\left(-D(\mu_1, y_i) j\right)}{(j+1)(\mu_1 - y_i)^2}\right). \tag{3.35}$$

This gives us a bound for term A of,

$$\sum_{k=1}^{T-1} \mathbb{E}\left[\frac{1}{p_{i,\tau_k+1}} - 1\right] \tag{3.36}$$

$$\leq \sum_{j=0}^{T-1} \Theta\left(e^{\frac{-(\mu_1 - y_i)^2 j}{2}} + \frac{e^{-D(\mu_1, y_i) j}}{(j+1)(\mu_1 - y_i)^2}\right). \tag{3.37}$$

We can see that this is less than the bound derived in equation 3.37 for the original proof for Thompson Sampling.

Term B

Term B is bounded by lemma 3 in the proof by Agrawal and Goyal. The term captures the event that a sample, θ, from a sub-optimal arm, i, is larger than the problem-defined constant y_i, when the estimated mean of the arm is *below* the problem-defined constant x_i. Note that the estimated mean $\bar{\mu}_i < x_i < y_i$. This is therefore a statement about the probability of sampling from the right-hand side of the posterior distribution for arm i. The modification Optimistic Thompson Sampling makes is only to the left-hand tail of the distribution. Any sample that is drawn from the left-hand tail is replaced by the mean. Therefore on the event that Term B applies Optimistic Thompson Sampling it behaves identically to Thompson Sampling. Lemma 3 then applies to Optimistic Thompson Sampling

in the same way. The lemma is given below.

Lemma 3.2.3.

$$\sum_{t=1}^{T} \Pr(i(t) = i, E_i^{\mu}(t), \overline{E_i^{\theta}(t)}) \leq \frac{\ln T}{D(x_i, y_i)} + 1. \tag{3.38}$$

Proof. Firstly we bound the probability $\Pr\left(i(t) = i, \overline{E_i^{\theta}(t)} | E_i^{\mu}(t), \mathcal{F}_{t-1}\right)$. We bound this term as follows,

$$\Pr\left(i(t) = i, \overline{E_i^{\theta}(t)} | E_i^{\mu}(t), \mathcal{F}_{t-1}\right) \leq \Pr\left(\theta_i(t) > y_i | \hat{\mu}_i(t) \leq x_i, \mathcal{F}_{t-1}\right). \tag{3.39}$$

In Optimistic Thompson Sampling,

$$\theta_i(t) \sim \max(\mathbb{E}\left[\mu_i\right], s_i(t)),$$

where $s_i(t) \sim \text{Beta}\left(\hat{\mu}_i(t)k_i(t) + 1, (1 - \hat{\mu}_i(t))k_i(t) + 1\right)$. On the event $E_i^{\mu}(t)$ a sample $s_i^*(t) \sim \text{Beta}\left(x_i k_i(t) + 1, (1 - x_i)k_i(t) + 1\right)$ is likely to be at least as big since $\hat{\mu}_i(t) \leq x_i$. Therefore,

$$\begin{aligned}
\Pr\left(\theta_i(t) > y_i | \hat{\mu}_i(t) \leq x_i, \mathcal{F}_{t-1}\right) &= \Pr\left(\max(\mathbb{E}\left[\mu_i\right], s_i^*(t)) > y_i\right), && (3.40)\\
&= 1 - F^{\text{Beta}}_{x_i k_i(t)+1, (1-x_i)k_i(t)+1}(y_i) && (3.41)\\
&= F^{B}_{k_i(t)+1, y_i}(x_i k_i(t)), && (3.42)\\
&\leq F^{B}_{k_i(t), y_i}(x_i k_i(t)), && (3.43)\\
&\leq e^{-k_i(t) D(x_i, y_i)}, && (3.44)
\end{aligned}$$

where $F^{\text{B}}_{n,p}(.)$ is the cumulative density function of a Binomial distribution with parameters n, p and $F^{\text{Beta}}_{\alpha,\beta}(.)$ is the cumulative density function of a Beta distribution with parameters α, β.

Therefore, for t where $k_i(t) > \frac{\ln T}{D(x_i, y_i)}$, it then follows that

$$\Pr\left(i(t) = i, \overline{E_i^{\theta}(t)} | E_i^{\mu}(t), \mathcal{F}_{t-1}\right) \leq \frac{1}{T}. \tag{3.45}$$

Letting τ be the largest time until $k_i(t) \leq \frac{\ln T}{D(x_i, y_i)}$ we can then bound term B

as follows,

$$\sum_{t=1}^{T} \Pr\left(i(t)=i, \overline{E_i^{\theta}(t)}, E_i^{\mu}(t)\right) \leq \sum_{t=1}^{T} \Pr\left(i(t)=i, \overline{E_i^{\theta}(t)} | E_i^{\mu}(t)\right), \tag{3.46}$$

$$= \mathbb{E}\left[\sum_{t=1}^{T} \Pr\left(i(t)=i, \overline{E_i^{\theta}(t)} | E_i^{\mu}(t) \mathcal{F}_{t-1}\right)\right], \tag{3.47}$$

$$= \mathbb{E}\left[\sum_{t=1}^{\tau} \Pr\left(i(t)=i, \overline{E_i^{\theta}(t)} | E_i^{\mu}(t) \mathcal{F}_{t-1}\right)\right. \tag{3.48}$$

$$\left. + \sum_{t=\tau+1}^{T} \Pr\left(i(t)=i, \overline{E_i^{\theta}(t)} | E_i^{\mu}(t) \mathcal{F}_{t-1}\right)\right], \tag{3.49}$$

$$\leq \mathbb{E}\left[\sum_{t=1}^{\tau} \Pr\left(i(t)=i, \overline{E_i^{\theta}(t)} | E_i^{\mu}(t) \mathcal{F}_{t-1}\right) + \sum_{t=\tau+1}^{T} \frac{1}{T}\right], \tag{3.50}$$

$$\leq \mathbb{E}\left[\sum_{t=1}^{\tau} I(i(t)=i)\right] + 1, \tag{3.51}$$

$$\leq \frac{\ln T}{D(x_i, y_i)} + 1. \tag{3.52}$$

□

Term C

Term C is bounded by lemma 2 in the proof by Agrawal and Goyal. The term is the event that a sub-optimal arm is over-estimated. By over-estimated we mean that the empirical mean of the arm, i, is above the problem-dependent constant x_i. The empirical mean of a suboptimal arm is the same in Optimistic Thompson Sampling as it is in Thompson Sampling, and so the bound is the same. The lemma is as follows,

Lemma 3.2.4.

$$\sum_{t=1}^{T} \Pr(i(t)=i, \overline{E_i^{\mu}(t)}) \leq \frac{1}{D(x_i, \mu_i)} + 1. \tag{3.53}$$

Proof. Let τ_k be the time at which the kth trial of arm i happens ($\tau_0 = 0$). Then

it follows that,

$$\sum_{t=1}^{T} \Pr\left(i(t)=i, \overline{E_i^{\mu}(t)}\right) \leq \mathbb{E}\left[\sum_{k=1}^{T}\sum_{t=\tau_k+1}^{\tau_{k+1}} I(i(t)=i) I(\overline{E_i^{\mu}(t)})\right], \tag{3.54}$$

$$= \mathbb{E}\left[\sum_{k=0}^{T-1} I(\overline{E_i^{\mu}(\tau_k+1)}) \sum_{t=\tau_k+1}^{\tau_{k+1}} I(i(t)=i)\right], \tag{3.55}$$

$$= \mathbb{E}\left[\sum_{k=0}^{T-1} I(\overline{E_i^{\mu}(\tau_k+1)})\right], \tag{3.56}$$

$$\leq 1 + \mathbb{E}\left[\sum_{k=1}^{T-1} I(\overline{E_i^{\mu}(\tau_k+1)})\right], \tag{3.57}$$

$$\leq 1 + \sum_{k=1}^{T-1} \exp\left(-kD(x_i,\mu_i)\right), \tag{3.58}$$

$$\leq 1 + \frac{1}{D(x_i,\mu_i)}. \tag{3.59}$$

□

Final bound

We thus end up with the same asymptotic bounds as with Thompson Sampling, and see that we achieve a problem-dependent bound of,

$$\mathbb{E}\left[R_T\right] \leq \sum_{i\neq 1}(1+3\varepsilon)\frac{\ln T}{D(\mu_i,\mu_1)}\Delta_i + O\left(\frac{K}{9\varepsilon^2}\right), \tag{3.60}$$

and a problem-independent bound of,

$$\mathbb{E}\left[R_T\right] \leq \Theta\left(\sum_{i\neq 1}\frac{\ln T}{\Delta_i}\right), \tag{3.61}$$

which are asymptotically efficient. However, we can see from the bound for term A, that some additive lower order terms are removed. This supports the empirical findings that Optimistic Thompson Sampling shows a slight improvement over conventional Thompson Sampling.

3.2.3 Optimism for the underdog

We here consider a modification to Optimistic Thompson Sampling that will be useful to us in Chapter 5. In the language of May et al. a greedy strategy acts purely on the basis of the exploitative value of the actions, only picking the arm which in expectation can be exploited for most reward. However if we only ever picked the arm with highest exploitative value we may never learn which arm is truely the best. Therefore the exploratory value is added to enable exploration of alternative arms. We can then view the exploratory value as an added encouragement to pull a given arm. In Optimistic Thompson Sampling this "encouragement" is applied to all arms, even the arm with highest exploitative value. However, this arm is the de facto choice so when we exploit we are in effect exclusively exploring this arm. The question is whether we should be adding this exploratory value to the current exploitative arm, or only the rest of the arms (perceived as underdogs in the competition to be picked as "best"). We call this modification *Optimism for the Underdog Thompson Sampling* (OUTS). The sample estimate for the arm with highest empirical mean is taken to be the empirical mean itself, otherwise the sample estimate is the same as Optimistic Thompson Sampling for all other arms. OUTS is described in Algorithm 3.4.

Algorithm 3.4 Optimism for Underdogs Thompson Sampling for Bernoulli Bandits

Let $\alpha_{1,k} = 1$,
$\beta_{1,k} = 1$ for $k \in \{1, \dots, K\}$.

for $t = 1, \dots, T$ **do**
Find arm $b = \text{argmax}_{k \in \{1,\dots,K\}} \frac{\alpha_{t,k}}{\alpha_{t,k}+\beta_{t,k}}$
$\theta_b = \frac{\alpha_{t,b}}{\alpha_{t,b}+\beta_{t,b}}$
Sample $\theta_i \sim \text{Beta}(\alpha_{t,i}, \beta_{t,i})$, for $i \in \{1, \dots, K\} \setminus \{b\}$..
Pull arm $a_t = \text{argmax}_i \max\left(\theta_i, \frac{\alpha_{t,i}}{\alpha_{t,i}+\beta_{t,i}}\right)$
Let $\alpha_{t+1,a_t} = \alpha_{t,a_t} + \mathbb{1}(x_{a_t}(t) = 1)$
$\beta_{t+1,a_t} = \beta_{t,a_t} + \mathbb{1}(x_{a_t}(t) = 0)$
Let $\alpha_{t+1,j} = \alpha_{t,j}$
$\beta_{t+1,j} = \beta_{t,j}$ for $j \in \{1, \dots, K\} \setminus \{a_t\}$.
end for

The reasoning in applying the proof techniques of Agrawal and Goyal to Optimistic Thompson Sampling also apply for bounding the regret of this modified

algorithm. Empirically we found that this modification was an improvement on the original Thompson Sampling algorithm, however it appeared that Optimistic Thompson Sampling was still more sample efficient. This leads to the assumption that added exploratory value is useful in all arms. Figure 3.4 shows an experiment comparing the three methods on a 6-armed Bernoulli bandit problem. The experiment was run 300 times and the average regret was plotted as a function of time with error bars marking the standard error. Although this result backs the hypothesis that optimism is advantageous, further investigation is required to give a fuller picture as to what extent this is true in general.

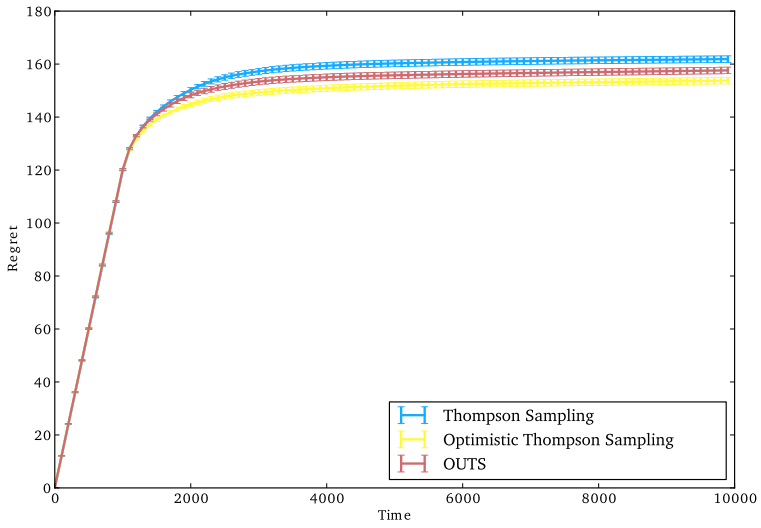


Figure 3.4: The regret of Thompson Sampling, Optimistic Thompson Sampling and OUTS on a 6-armed Bernoulli bandit problem. We can see both Optimistic Thompson Sampling and OUTS performing better than conventional Thompson Sampling.

Chapter 4

Applying Thompson Sampling to Non-Stationary Bandits

4.1 Introduction

The assumption made in the multi-armed bandit problem discussed in the previous chapter is that the distributions from which the environment draws rewards are stationary. That is, the expected payoff of an arm at a given round is temporally invariant. This assumption rarely holds in practice. Take the web ad placement problem as an example. The effectiveness (as measured by click through rates) of an ad can change in all manner of ways. The ad may be for barbecues, it is likely to be successful in summer and less so in winter. This sort of change is seasonal, and periodic. Some changes are not periodic, for example, the summer for a given year might be unusually cold with high rainfall making the ad less effective, or there could be a once-in-a-generation prolonged heatwave making the ad desirable for longer.

The above examples show why a system, and therefore actions based on it, might change. For some examples, the parameters of the system a change depends on are *known*. For example, seasonal change in demand for barbecues could be learned by a decision-making agent if they knew the date. The decision effects could be directly modelled as a function of this parameter. This approach is taken with Contextual bandits. In other scenarios, given parameters might be known to affect the outcome of decision but are not available to the agent. It may be plausible to model and infer these *known unknowns* to aid the decision-making process. However real world systems can often be complex, and not fully

understood. The response of decisions could be influenced by variables that are not known, or properly accounted for. This falls into the category of *unknown unknowns*. Finally the system may change independent of external parameters. It is desirable for an agent to be able to adapt in the face of all of these sources of change.

There are two main ways in which an action response can be affected over time. The action response can *drift*. An example might be a system where the mean response of an action changes continuously as a function of time. A characteristic of a drifting system is that the recent history of the behaviour of the system has a reasonable correlation with the present mean response. The second type of change is when the system exhibits *switching* behaviour. In this case the mean response can change in such a way that the present is uncorrelated with the recent past history.

This chapter focuses on non-stationary multi-armed bandit problems. Specifically on Thompson Sampling in these environments. The main contribution in the chapter is a class of algorithm, Change-point Thompson Sampling, designed with a switching environment in mind.

4.1.1 Model of dynamic environment

Real decision-making scenarios take place in environments that change over time. We therefore want to design algorithms that assume the environment changes over time. Our proposal is that the model of a switching environment is a useful one on which to base a decision-making agent that can adapt appropriately to changes that happen in real-world applications. We assume abrupt switching defined by a hazard function, $h(t)$, such that,

$$\mu_i(t) = \begin{cases} \mu_i(t-1) & \text{with probability } h(t) \\ \mu_{\text{new}} \sim U(0,1) & 1 - h(t). \end{cases} \tag{4.1}$$

The algorithms developed are designed with two such models in mind. The first model we will refer to as the *Global Switching* model. This model switches at a constant rate (γ); when a change point happens *all* arms change their expected rewards. The second model will be referred to as *Per-Arm Switching*. In this model change points occur independently for each arm, such that the times when arms switch are uncorrelated from each other (see Figure 4.1 for a pictorial

representation of each of the two models).

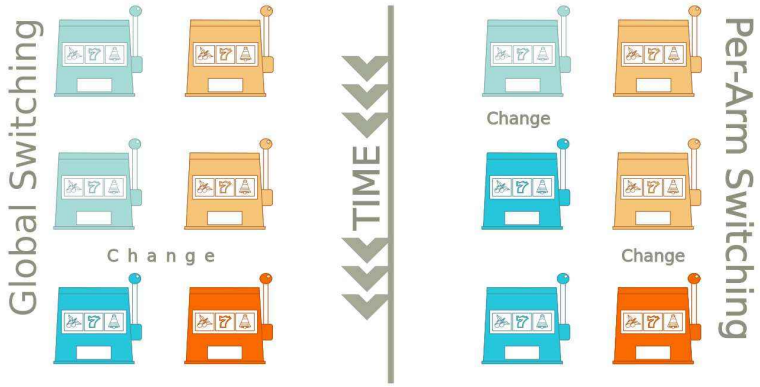


Figure 4.1: Global and Per-Arm Switching Models

4.2 Motivation: examples of switching

We propose that a switching environment is a good model for a wide variety of decision-making scenarios. In this section, in order to motivate our work we briefly explore a few scenarios that we deem should exhibit switching behaviour.

4.2.1 Game playing

A common type of decision-making scenario is that of game playing. Two or more agents must compete against, and sometimes cooperate with, each other in order to gain reward within a game. For example in the card game Poker there are a number of agents who play together in a game. Cards are dealt to each of the agents in a series of rounds such that each agent is unaware of the cards the others have been dealt. An agent must decide whether to bid, call or fold in order maximise the size of their winnings. The strategy that leads to largest winnings for an agent depends not only on the hands of themselves and all other agents, but on the strategies of the other competitors (how the other agents decide to bid, call or fold). The strategies of the competing agents might be assumed to be stationary, such that given the same hand the competitor behaves to the same statistical law. A common situation however, is that the competitor is learning by experience and adapting its strategy based on the results of previous games. This means that the competitor agent's strategy will change (given the

same observable state the expected decisions will be different) during the time of play. This of itself is an argument for an agent to be adaptive to non-stationary environments in game playing scenarios, but we conjecture that the strategies of some types of competitor is likely to be switching in nature.

Q-Learning agents

We consider playing some game against a reinforcement learning agent. A popular category of reinforcement learning is *value-function* based learning. A value function maps states or state-action pairs to estimates of the future discounted rewards for either being in a given state (in the case of mapping from just states) or choosing an action in a given state (in the case of mapping from state-action pairs). Value-function based agents then derive a policy (mapping a state to a distribution over the probability of taking an action) based on the current state of the value function. When new observations are collected via interacting in the environment the value-function is updated, which in turn may alter the policy that the agent uses to choose actions. A popular value-function based algorithm is Q-Learning. Let the environment the agent acts in be a space of states $s \in \mathcal{S}$, and let the agent have a number of actions $k \in \mathcal{K}$ available to it. When the agent performs an action a_t in state s_t they receive a reward, x_t, and move from state s_t to state $s_{t+1} \in \mathcal{S}$. The value-function, denoted $Q(s_t, a_t)$, learns an estimate of the expected discounted reward, $\sum_{t=1}^{\infty} \xi^{t-1} x_t$, that the agent will receive by choosing action a_t in state s_t. The Q-Learning algorithm updates this estimate via the following update equation,

$$Q(s_t, a_t) = (1-\alpha)\overbrace{Q(s_t, a_t)}^{\text{past estimate}} + \overbrace{\alpha}^{\text{learning rate}}\left(x_t + \overbrace{\xi \max_{k \in \mathcal{K}} Q(s_{t+1}, k)}^{\text{discounted estimate of future rewards}}\right), \tag{4.2}$$

where α is the learning rate and ξ is the discount factor, both parameters to be set for the algorithm. It is assumed that both $\alpha, \xi \in (0, 1)$. The smaller α the less influence a new observation has on the updated estimates (learning happens more slowly). The smaller ξ the more the value-function estimates depend on immediate rewards and so the more likely the agent is to think only of short term gain. Alone this does not specify how the agent should act, the agent needs a strategy of how to use the information in the value function in order to form a policy. A simple example of a policy is the greedy policy. The greedy policy

chooses the action in a given state that has the largest value in the value function.

The observation is that for appropriate (and we argue typical) values of α and ξ an agent's greedy policy does not change on every update of the value function when learning to play a game. Instead there is often a period of time where the value function is updated without changing the greedy action. Eventually after repeated rounds the value function changes such that the greedy action changes.

Rock, Paper, Scissors

By way of an example we will demonstrate this by considering a Q-Learning agent playing Rock, Paper, Scissors in self-play. By self-play we mean an agent competes against another agent that uses the same learning strategy, in this case Q-Learning. Rock,Paper,Scissors is a two-player game. The agents choose one of three actions (*rock*, *paper* or *scissors*) without knowledge of their opponent's choice. They then reveal their choices to one another simultaneously. The winner, who gains a reward of one, is decided by the two agents' choices in the following way; paper beats rock, rock beats scissors and scissors beats paper. In the case of a tie (they choose the same action) then both players receive no reward. The game *payoff matrix* for Rock,Paper,Scissors can be specified as follows,

		Player 1		
		Rock	Paper	Scissors
Player 2	Rock	0,0	1,0	0,1
	Paper	0,1	0,0	1,0
	Scissors	1,0	0,1	0,0

This game can be repeated for many rounds in an iterated fashion. The player at the end who has won the most number of round is declared the winner.

To make a Q-Learning agent to play this game we need to map the game to a series of states and associated actions. For Rock, Paper, Scissors this is straightforward since there is no observable difference between one round and the next, and so there is just a single state. Since there is one state in our notation we will omit the state in the value function, so $Q(s_t, a_t)$ becomes $Q(a_t)$. The actions in the game are just the choices *rock*, *paper* and *scissors*. The value function for the agent can then be thought of as a lookup table with three values $Q(rock)$, $Q(paper)$ and $Q(scissors)$, which are then updated as described in Equation 4.2. The value function is initialised with all values randomly selected between zero and one.

When two such agents are played against each other we can observe periodic switching in the policy of the agents. One agent will gain the upper hand and win a round. The entry of the value function associated with the action that caused them to win is updated, it will either increase or stay the same. This means that they will play the same action again next time. They will continue to do so until the value associated with the dominant action reduces to be below the value of one of the other actions. The time this takes depends on the opponents strategy and the agent's parameters α and ξ. Figure 4.2 shows the evolution of the value function of one of the agents in self play. The Q-values for each action are superimposed on the same graph. The switching in strategy can be clearly seen. The Q-value for playing paper initially increases as this is a strategy continues to beat the opponent's strategy. Eventually the opponent learns this and switches strategy, and so the Q-value for the player shown goes down. This continues until the Q-value for playing paper gets sufficiently low that playing scissors is considered a better option. The Q-value for scissors then increases until the opponent changes strategy again, and so on.

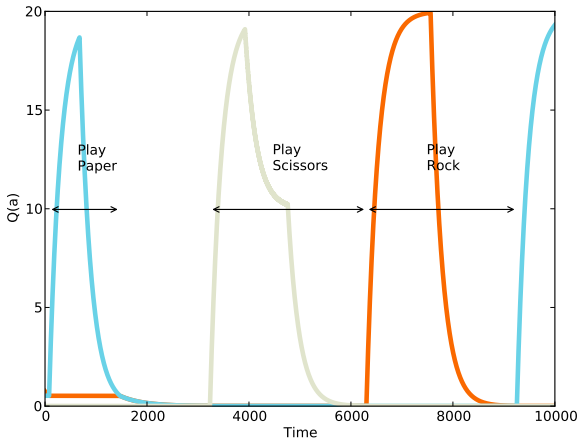


Figure 4.2: The evolution of the value function, $Q(a_t)$, for a Q-Learning agent in self-play in the game Rock,Paper,Scissors. The value-function was randomly initialised with values between zero and one, $\alpha = 0.1$ and $\xi = 0.95$. The switching of policy from action to another can be seen. In this self-play setting the switching appears to be periodic with a characteristic time scale dependent on α and ξ.

If the time for such an agent switching policy is sufficiently long then a decision-making agent that could respond to their switching behaviour might be able to exploit them.

4.2.2 Financial scenarios

The bandit model for decision-making has been employed in economics to model problems in financial markets. One of the first using the bandit model was due to Rothschild [61]. Rothschild was interested in the process by which companies learn the market demand for their goods and services. The problem he considered is a single company gauging the market demand. The actual demand is given by a probability distribution over the consumer valuations observable to the company. The demand is assumed to be one of a finite set of possible values each with a prior probability of being the true demand. The company can then set its prices in a sequence of rounds and observe consumer valuations in order to learn what the true demand is. The problem of Rothschild was assumed to be stationary. Keller et al. considered a similar problem to Rothschild, but one key difference is they assume the unknown demand is subject to changes over time and is therefore non-stationary [35]. Sorensen has also considered bandit-like models in the context of Venture Capital funding, where a Venture Capitalist must weigh up the continued investment in a number of different start-up companies [65]. Some research on the application of bandit problems to financial markets was summarised by a survey by Bergemann et al. [9]. This body of work shows the interest from the economics research community in the multi-armed bandit problem.

We have already appealed to a general argument about the changing nature of real world problems in order to motivate our work, and indeed we believe that financial markets are subject to such changes. However what is the evidence that financial markets exhibit switching behaviour? Preis et al. have studied the dynamics of financial markets and have hypothesised and provided evidence for switching behaviour [59]. Since switching behaviour is observed in such markets by extension we believe that bandit algorithms that account for this switching behaviour are worthy of exploration.

4.2.3 Networks

Networks and graph structure arise in many different applications from computer networks to road networks. A network is a series of nodes connected by edges. For instance in a logistic problem for a haulage problem, the nodes might represent various distribution centres and customer sites, and the edges represent the possible routes between them. In a computer network, the nodes are instead

computers and the edges represent communication channels between them. Such networks are prone to various types of failure. In road networks, changes in traffic load throughout the day, car accidents and road repairs can all lead to changes in the network. In computer networks, hardware is prone to failure causing nodes to become unreachable, fundamentally changing network structure. We propose that such unpredictable changes constitute a sudden switching change to the network behaviour. We believe therefore that decision-making systems operating in environments dependent on the behaviour of networks should be adaptable to sudden abrupt changes as with a switching environment. An example of a network related problem for which multi-armed bandit algorithms have been used is in information collection by wireless sensor networks proposed by Tran-Thanh et al. [70]. Wireless sensor networks are a collection of densely deployed, spatially decentralised autonomous sensor devices communicating through a wireless communication network. The task of such systems is often in monitoring physical conditions of the environment in which the sensor network is deployed. The measurements of interest may be anything including temperature, noise, pressure, seismic and magnetic information. The devices are often cheap, with both low energy and computational power. In order to collect information effectively the nodes of the network must balance their low computational/energy budget between routing information collected through the network (either transmitting or receiving), or sampling new information via their sensors. The balance between the different tasks constitutes an energy allocation for the device. Tran-Thanh et al. proposed modelling learning the best energy allocation for a device as a multi-armed bandit problem. They employed an Exp3 variant [8] for their work, since it makes no assumptions of the changing nature of the problem. However, we feel that there is a case that this type of problem could also be suitably modelled by a switching system.

4.3 Bayesian online change-point detection

The model of the bandit environment we are concerned with is that of a switching system. That is, at given time points, known as change-points, the statistical law from which arm rewards are drawn changes (the probability of some reward before a change-point will be different than the probability of the same reward after the change-point). If we knew when these change-points happened it could

help us to improve the agent's decision-making ability as we could ensure we only incorporate data relevant to agent's current decision. There exists a large literature on change-point techniques, for example the Page-Hinkley test used in Adapt-EvE (see section 2.7.2). There are a few requirements that we desire for our change-point detection mechanism in order for it to be useful for a Thompson Sampling inspired bandit algorithm. These are:

- *online* - The mechanism needs to be efficient to compute, scaling reasonably (both in space and time complexity) with the number of data points.
- *Bayesian* - The mechanism needs to produce a posterior distribution of when change-points occurred. In order to remain a Thompson Sampling strategy we need to be able to sample from a posterior distribution.

It turns out that these conditions can be met by an inference algorithm. Rather than ask the question "did a change-point occur at time t?" the key is to ask "how long ago was it since a change-point occurred?". The time elapsed since the last change-point occurred is called the *runlength*. Fearnhead and Liu [19] as well as Adams and MacKay [1] have independently done work on calculating the online posterior of the runlength. Fearnhead and Liu applied the method to a segmenting the genome into Isochores. Adams and MacKay applied the method to problems like tracking geophysical data and financial market data. They show exact inference on the runlength can be achieved by a simple message passing algorithm. Let x_t be the reward at time t so that $D_t = x_t \cup D_{t-1}$ is the history of past rewards. Let r_t be the runlength at time t. The inference procedure can be easily derived as follows.

$$P(r_t|x_{t-1}, D_{t-2}) = \frac{P(r_t, x_{t-1}, D_{t-2})}{P(x_{t-1}, D_{t-2})} \tag{4.3}$$

The numerator can then be expressed as

$$P(r_t, x_{t-1}, D_{t-2}) = \sum_{r_{t-1}} P(r_t, r_{t-1}, x_{t-1}, D_{t-2}) \tag{4.4}$$

$$= \sum_{r_{t-1}} P(r_t, x_{t-1}|r_{t-1}, D_{t-2}) P(r_{t-1}, D_{t-2}) \tag{4.5}$$

$$= \sum_{r_{t-1}} \overbrace{P(r_t|r_{t-1})}^{\text{switching rate}} \overbrace{P(x_{t-1}|r_{t-1}, D_{t-2})}^{\text{reward likelihood}} P(r_{t-1}, D_{t-2}). \tag{4.6}$$

The derivation just applies the rules of probability up to and including Equation 4.5. One assumption is made in Equation 4.6, that the runlength is only dependent on the runlength at the previous timestep. This forms a simple message passing algorithm because r_t can only take values depending on r_{t-1}. In fact $r_t = r_{t-1} + 1$ when switching does not occur and $r_t = 0$ when it does. $P(r_t|r_{t-1})$ is defined by a hazard function $h(t)$. Figure 4.3 pictorially represents the message passing scheme.

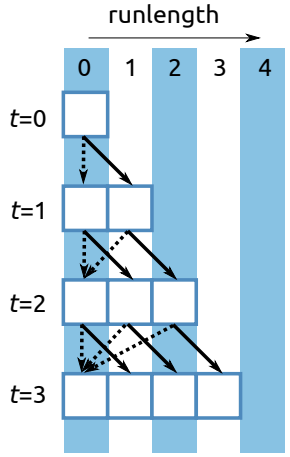


Figure 4.3: Message passing for runlength inference. Boxes represent runlengths in the runlength distribution, and arrows represent messages that are passed between them in the update algorithm. At the top of the picture the runlength distribution is shown starting as a single runlength (here we initialise the algorithm such that we assume a change-point has happened, so the runlength distribution just contains the runlength of zero). After one time step the runlength distribution grows to two runlengths (zero and one). Either a change occur and so the runlength remains at zero (shown by a message from zero to zero) or a change did not happen and the runlength increments to one. The picture shows how the runlength grows by one after each time step. The update step has a cost (both in time and space) linear in the number of possible runlengths.

The switching rate and the reward likelihood thus both play a role in the learning process. When the switching is constant the switching rate represents a fixed factor that affects the probability of a given runlength, irrespective of the data seen. The reward likelihood for this case is the term dependent on the incoming data and so is where learning can be seen to occur.

4.3.1 Computational complexity

Unfortunately the exact inference has space and time requirements that grow linearly in time for each step. The space requirements are linear because at each time step the support set of the posterior runlength distribution (the values of time for which the runlength has non-zero probability) increases by one, which means we have to store information for an extra value of the runlength at every step. The update is also linear in time, as the message passing algorithm requires an update to each runlength in the support. This not ideal for our use-case. We want to allow our agent to adapt to changes throughout its use. If the cost of the algorithm is linear in time, this means the time to make decisions will continue to get longer as time progresses, and that eventually memory resources for the agent will be saturated. The algorithm needs to be modified. The runlength distribution needs to be made approximate in order to reduce the requirements. Adams and MacKay suggest a simple thresholding technique to eliminate runlengths with small probability mass associated with them. This means we can know in expectation how much memory this algorithm will require. However this is only in expectation; the memory requirements might deviate significantly from this, also causing the running time to increase. An alternative with hard guarantees on memory requirements is desirable. Fearnhead and Liu suggest a much more sophisticated particle filter resampling step to maintain a finite sample of the runlength distribution, which has the benefit that we can be certain on the upper limit of space the algorithm requires.

4.3.2 Stratified optimal resampling

The runlength distribution is a discrete distribution. Any approximation for the distribution needs to be compatible with an update inference step. The easiest approach to this, similarly to the Adams and MacKay approach, is to discard some of the runlengths in the runlength distribution. As stated the problem with using a threshold on the probability, under which a runlength is discarded, is that the memory requirements are harder to specify such that the system remains within its resource limits. Liu and Fearnhead's approach is a particle filter technique to discard runlengths in the runlength distribution. The method is called *Stratified Optimal Resampling*. A particle filter is a Monte-Carlo method for approximately estimating a sequential Bayesian model. Particles are used to represent points

in the distribution to be estimated and are assigned weights that correspond to their approximate probabilities. The number of particles can grow at each time step and so occasionally some particles need to be thrown away. This leaves us to assign new weights to the remaining particles. This procedure is called resampling. Since the distribution is discrete each particle is used to represent a separate runlength. The procedure is designed so the practitioner can specify a maximum limit to the particles stored. When this maximum limit is reached the procedure reduces the number of particles down to another pre-specified number. Let the maximum number of particles to be stored be N. Once this is reached let M be the number of particles for the distribution to be reduced to (by discarding $N - M$). We can write the (possibly approximate) runlength distribution when it reaches N particles as the set of probabilities $p_1, \ldots, p_N$. We can think of the approximate runlength distribution of M particles as an N particles distribution $q_1, \ldots, q_N$ such that $N - M$ of the q_i are zero. Stratified Optimal Resampling is optimal in the sense that:

- $\mathbb{E}[q_i] = p_i$ for all i (the distributions remain the same in expectation)
- $\sum_{i=1}^{N}(p_i - q_i)^2$ is minimised.

Assume that the particles are ordered by their runlength such that for example the runlength associated with p_1 is smaller than the runlength associated with p_2. Since $(p_1, \ldots, p_N)$ is a distribution $\sum_{i=1}^{N} p_i = 1$. Firstly Stratified Optimal Resampling (SOR) finds the unique solution α to the expression

$$\sum_{i=1}^{N} \min(1, \frac{p_i}{\alpha}) = M. \tag{4.7}$$

Any particle p_i in the original distribution that is greater than or equal to α is kept, such that $q_i = p_i$ when $p_i \geq \alpha$. This stage of SOR will have kept $A \leq M$ particles. The next stage is a stratified resampling step. The algorithm for this was proposed by Carpenter et al. [14]. A random variable u is drawn uniformly from the interval $[0, \alpha]$. The algorithm proceeds in order through the remaining $N - A$ particles. u is reduced by the probability of a particle ($u = u - p_i$). If u becomes zero or lower then keep the particle but now assigned with a new probability such that $q_i = \alpha$. If the particle is kept update u to $u = u + \alpha$ and continue to the next particle. The full procedure is summarised in Algorithm 4.1.

Algorithm 4.1 Stratified Optimal Resampling

Require: Distribution $\{p_i : 0 < i \leq N\}$, of N particles with ordering $\sigma(i)$ s.t. $\sigma(i) < \sigma(j)$ for $i < j$
Require: Parameter $M < N$
  Find α s.t. $\sum_{i=1}^{N} \min\left(1, \frac{p_i}{\alpha}\right) = M$
  Initialise u by drawing uniformly from $[0, \alpha]$.
  for $i = N$ **do**
    if $p_i \geq \alpha$ **then**
      $q_i = p_i$
    else
      $u = u - p_i$
      if $u \leq 0$ **then**
        $q_i = \alpha$
        $u = u + \alpha$
      else
        $q_i = 0$
      end if
    end if
  end for

The particles with probability $p_i > \alpha$ contribute 1 to the sum $\sum_{j=1}^{N} \min(1, \frac{p_j}{\alpha})$, and so are kept with their probability unchanged. Given that there are A particles saved in this way we can rearrange Equation 4.7 to get,

$$\sum_{i:p_i/\alpha<1} p_i = (M - A)\alpha. \tag{4.8}$$

In order so that $\sum_{i=1}^{N} q_i = 1$, the remaining $M-A$ particles that are not discarded are therefore given weight α.

A consequence of ordering the particles is that the maximum Kolmogorov Smirnov distance between the distribution $\{p_i\}$ and $\{q_i\}$ is bounded by α. The Kolmogorov Smirnov distance is defined as

$$\text{KSD} = \max\left\{\max_i \left|\sum_{j=1}^{i} (p_j - q_j)\right|\right\},$$

where the first maximisation is over realisations of $q_1, \ldots, q_N$. This is a metric describing the maximum distance between the cumulative density of the original distribution, $\{p_i\}$, and the resampled one, $\{q_i\}$.

The expression $\sum_{i=1}^{N} \min(1, \frac{p_i}{\alpha}) = M$ can be solved for α via a quick-select method on the probabilities p_i. A pivot is found (via a median of medians approach for example) and the particles are sorted into two bins, *left* and *right*, with those in *left* having probability less than the pivot, and those in *right* having probability greater than the pivot. A candidate solution for α is found by assuming that $\min(1, p_i/\alpha) = 1$ for all p_i in *right*, leading to a simple rearrangement of Equation 4.7 to find α. Given this value of α we can evaluate $\sum_{i=1}^{N} \min(1, \frac{p_i}{\alpha})$. If it equals M then we have found the correct value of α, but if it is not we can use whether it is less than or greater than M to steer whether the next pivot is to be found in *left* or *right*. In this way the Stratified Optimal Resampling step have a time complexity of $O(N)$ (N being the number of particles).

This algorithm allows us to give a fixed bound on the time and space requirements of inferring the runlength distribution. We can set N to a value based on the resources that are available and the specific time constraints of the application.

4.4 Switching Thompson Sampling

In order to perform Thompson Sampling we wish to sample from $P(\theta_t|D_{t-1})$, which is the probability of the arm model given the data so far. θ_t is a tuple of all the parameters defining the distribution of the arms. In our case this is the means $(\mu_1, \ldots, \mu_K)$. In a switching system the arms model θ_t is only dependent on the data since the last switching occurred, but we do not know when this happened. If we did we could just do the same Bayesian update as with the standard Bernoulli case to arrive at the distribution of our model. Since we do not know the runlength r_t we can introduce it as a latent variable and marginalise it out. Taking D_{t-1} as the history of rewards and arm pulls seen so far, we can write this as

$$P(\theta_t|D_{t-1}) = \sum_{r_t} \underbrace{P(\theta_t|D_{t-1}, r_t)}_{\substack{\text{posterior of} \\ \text{model given data}}} \overbrace{P(r_t|D_{t-1})}^{\substack{\text{probability} \\ \text{of runlength}}} . \tag{4.9}$$

Now to sample from $P(\theta_t|D_{t-1})$ we just need to sample from the $P(r_t|D_{t-1})$ (the runlength distribution) and then given that runlength, sample from $P(\theta_t|D_{t-1}, r_t)$ to arrive at our arm model θ_t. The Bayesian online change-point detection algorithm [19, 1] explained in section 4.3 provides a mechanism to infer $P(r_t|D_{t-1})$. Sampling from $P(\theta_t|D_{t-1}, r_t)$ is done in the same way as conventional Thompson Sampling.

The message-passing algorithm for Bayesian online change-point detection requires knowledge of the likelihood, $P(x_t|\mu_i, D_{t-1}, r_t)$, of a reward from an arm given the past history up until the last change-point. Between change-points we assume Bernoulli arms in our model, and can model our belief for the arm mean between change-points as a Beta distribution. We can integrate over this distribution to get a closed form estimate of $P(x_t|\mu_i, D_{t-1}, r_t)$.

4.5 Proposed inference models

We have shown we can perform Thompson Sampling in a switching system by splitting the procedure into a stage that samples the runlength since a switch occurred and a stage that samples from the arm model given this runlength.

We introduce two models of switching in a multi-armed bandit problem, *Global Switching* and *Per-Arm Switching*. The Global Switching and Per-Arm Switching models are appealing due to their simplicity. Only one runlength distribution needs to be inferred for Global Switching, which does not depend on the number of arms the bandit has. The Per-Arm model can store runlengths for each arm independently, and the space requirements grow linearly with respect to the number of arms. They are presented below. Models with more complicated inter-arm dependencies can quickly become intractable.

4.5.1 Global switching

In global switching there is a single change point process across all of the arms since when one arm switches distribution so do all other arms. This means that the data from every arm pull contributes to the posterior of the single runlength distribution. Effectively to sample from the posterior of the full bandit model, we first need to sample from the runlength distribution, this gives us an estimate of the runlength, which tells us how much data from the past our arms can use.

Once the global runlength is sampled, we then proceed by sampling individually from the posterior distributions of the arms, given only the data since the last changepoint (determined by the runlength). The arm with the corresponding maximum sample is then pulled. We only need to store the posterior probabilities of the given runlengths and the hyperparameters for the arm posteriors associated with those runlengths. We will call the runlength distribution the Change Point model, and the set of hyperparameters associated with each runlength for a given action the Arm model. The global switching model for a two arm bandit problem is represented graphically in Figure 4.4 where a single runlength distribution is shared between two arm models.

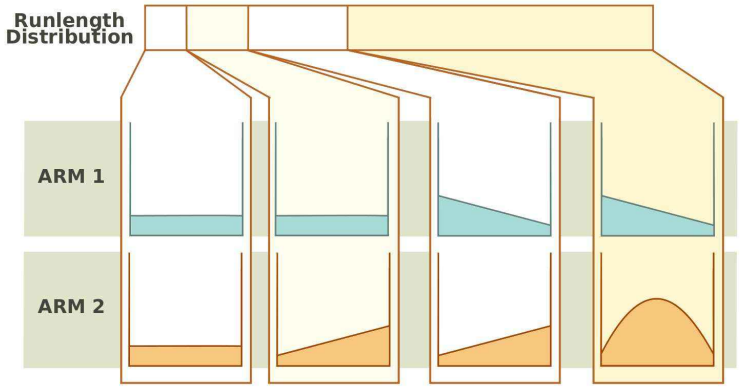


Figure 4.4: Global Change-Point Thompson Sampling: Here only one runlength distribution needs to be inferred across all the arms. When choosing an arm a sample is first drawn from this single distribution. This selects the arm distributions for which the regular Thompson Sampling algorithm is applied.

The Change Point model is an approximation of the runlength distribution storing a probability for at most N runlengths as defined in Section 4.3.2. Let w_i^t be the probability of having a runlength of i at time t. We consider a bandit where the arm rewards are assumed to come from a Bernoulli distribution so the hyperparameters stored are the 2 parameters for the Beta distribution. Let $\alpha_{i,j}^t$ and $\beta_{i,j}^t$ be the hyperparameters for a runlength of i at time t for arm j. At any point in time t there is a set of runlengths $R_t \subset \mathbb{N}$, $|R_t| \leq N$, where for every $r \in R_t$ there exists quantities w_r^t, $\alpha_{r,j}^t$ and $\beta_{r,j}^t$. When $|R_t| = N$ then a resampling step is performed in order to reduce the number of runlengths stored. For ease of notation let $\{w\}^t$ be the set of runlength probabilities at time t and let $\{\alpha\}_j^t$ and $\{\beta\}_j^t$ be the sets of hyperparameters for arm j at time t. Similarly let $\{\alpha\}^t$

and $\{\beta\}^t$ be the set of all hyperparameters at time t. The algorithm is presented in pseudocode in Algorithm 4.2 and Figure 4.5 shows the updating life cycle of particles diagrammatically.

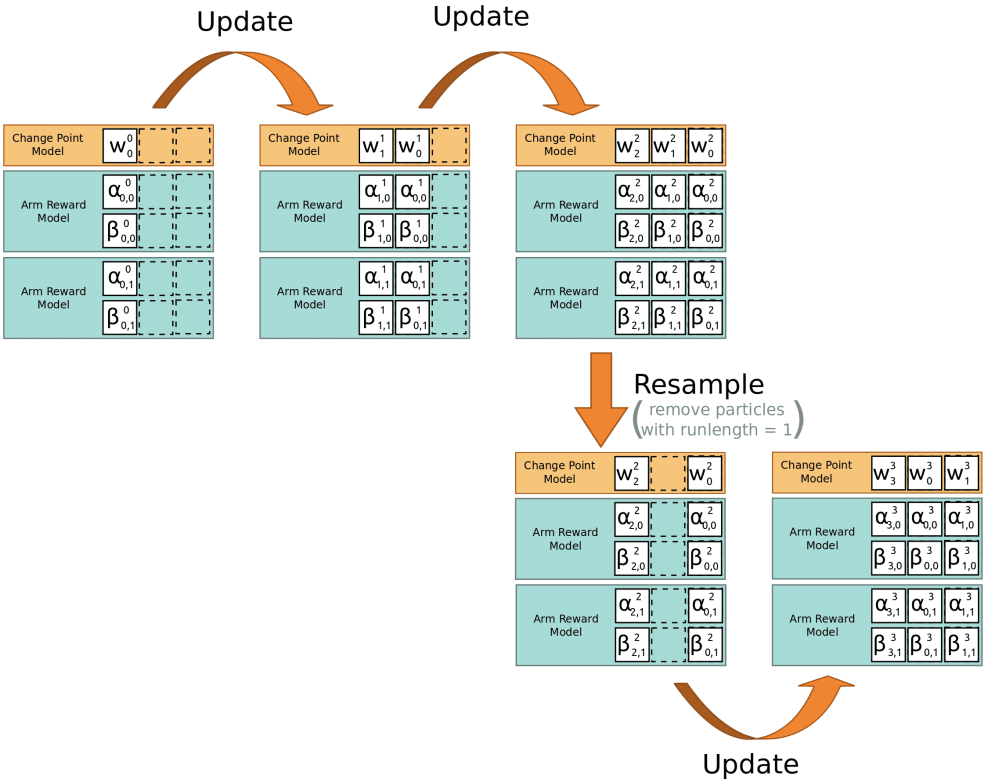


Figure 4.5: Global Change-Point Thompson Sampling: Here we show the evolution of the hyperparameters as we update the algorithm over time. Each column of hyperparameters being thought of as a particle. When the number of particles reaches its maximum the algorithm performs a resampling step.

We will refer to this algorithm as *Global Change-Point Thompson Sampling* (Global-CTS).

4.5.2 Per-arm switching

The difference in implementation with respect to global switching is that now there is a runlength distribution for each arm. That is, for each arm j we have a different set of runlength probabilities $w^t_{i,j} \in \{w\}^t_j$. In the per-arm switching model at a timestep t we update the Change Point model associated with the arm that was pulled at t much like via the update equations sketched in 4.6.

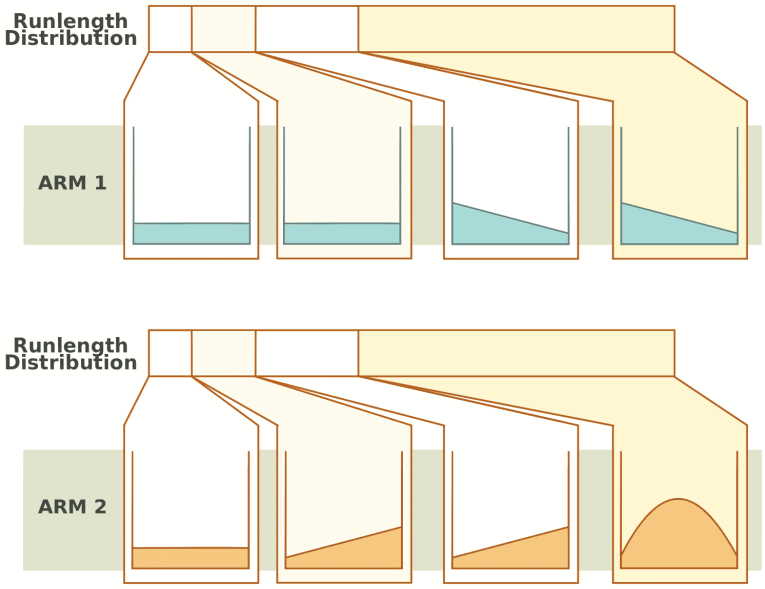


Figure 4.6: Per-Arm Change-Point Thompson Sampling: Unlike Global-CTS (see Figure 4.4), where there was as single runlength distribution, PA-CTS has a separate runlength distribution for every arm that needs to be inferred.

The Change Point models associated with arms not pulled at t are updated differently since the runlength for these arms is independent of the reward we received for the arm we actually pulled. The reward likelihood term disappears in the update equations for the runlength distribution of unpulled arms. This is shown in Equation 4.12. Since we normalise the distribution at each step we can ignore the factor $P(x_{t-1})$. This is shown as follows,

$$P(r_t, x_{t-1}, D_{t-2}) = P(x_{t-1})P(r_t, x_{t-1}, D_{t-2}) \tag{4.10}$$

$$\propto \sum_{r_{t-1}} P(r_t, r_{t-1}, D_{t-2}) \tag{4.11}$$

$$\propto \sum_{r_{t-1}} P(r_t|r_{t-1}, D_{t-2})P(r_{t-1}, D_{t-2}). \tag{4.12}$$

We will refer to this algorithm as *Per-Arm Change-Point Thompson Sampling* (PA-CTS).

Algorithm 4.2 Global Change-Point Thompson Sampling

Global-CTS$(N, \gamma, \alpha_0 = 1, \beta_0 = 1)$
  Let $t = 0$ {Initialise time}
  Let $w_0^t = 1$, and add to $\{w\}^t$ {Initialise runlength distribution}
  For all arms j, let $\alpha_{0,j}^t = \alpha_0$ {Initialise hyperparameters}
  For all arms j, let $\beta_{0,j}^t = \beta_0$
  while Interacting **do**
    Let $a_t =$ **SelectAction**$(\{w\}^t, \{\alpha\}^t, \{\beta\}^t)$
    Let $x_t =$ **PullArm**(a_t)
    Let $\{w\}^{t+1} =$ **UpdateChangeModel**$(\{w\}^t, \{\alpha\}_{a_t}^t, \{\beta\}_{a_t}^t, a_t, x_t, \gamma)$
    Let $\{\alpha\}^t, \{\beta\}^t =$ **UpdateArmModels**$(\{\alpha\}^t, \{\beta\}^t, a_t, x_t)$
    if $\left|\{w\}^{t+1}\right| = N$ **then**
      ParticleResample$(\{w\}^{t+1}, \{\alpha\}^{t+1}, \{\beta\}^{t+1})$
    end if
    $t = t + 1$
  end while
end

UpdateChangeModel$(\{w\}^t, \{\alpha\}_{a_t}^t, \{\beta\}_{a_t}^t, a_t, x_t, \gamma)$
  if $x_t = 1$ **then**
    Let likelihood$_i = \frac{\alpha_{i,a_t}^t}{\alpha_{i,a_t}^t + \beta_{i,a_t}^t}$, for all i s.t. $w_i^t \in \{w\}^t$
  else
    Let likelihood$_i = \frac{\beta_{i,a_t}^t}{\alpha_{i,a_t}^t + \beta_{i,a_t}^t}$, for all i s.t. $w_i^t \in \{w\}^t$
  end if
  Let $w_{i+1}^{t+1} = (1 - \gamma) *$ likelihood$_i * w_i^t$, for all i s.t. $w_i^t \in \{w\}^t$
  Let $w_0^{t+1} = \sum_i \gamma *$ likelihood$_i * w_i^t$
  Normalise $\{w\}^{t+1}$
  return$\{w\}^{t+1}$
end

UpdateArmModels $(\{\alpha\}^t, \{\beta\}^t, a_t, x_t)$
  if x_t=1 **then**
    Let $\alpha_{i+1,a_t}^{t+1} = \alpha_{i,a_t}^t + 1$, for all i s.t. $\alpha_{i,a_t}^t \in \{\alpha\}_{a_t}^t$
  else
    Let $\beta_{i+1,a_t}^{t+1} = \beta_{i,a_t}^t + 1$, for all i s.t. $\beta_{i,a_t}^t \in \{\beta\}_{a_t}^t$
  end if
  Let $\alpha_{0,j}^{t+1} = \alpha_0$, for all arms j {Set Prior for runlength 0}
  Let $\beta_{0,j}^{t+1} = \beta_0$, for all arms j
  return$\{\alpha\}^{t+1}, \{\beta\}^{t+1}$
end

ParticleResample$(\{w\}^{t+1}, \{\alpha\}^{t+1}, \{\beta\}^{t+1})$
  Find set to discard $d \in D$ using Stratified Optimal Resampling on $\{w\}^{t+1}$
  Discard all w_d^{t+1}, α_d^{t+1}, β_d^{t+1}
end

SelectAction$(\{w\}^t, \{\alpha\}^t, \{\beta\}^t)$
  Pick i with probability w_i^t
  Let sample$_j \sim Beta(\alpha_{i,j}^t, \beta_{i,j}^t)$ for all arms j.
  return $\max_j sample_j$
end

4.6 Learning the switching rate

Both Wilson et al. [76] and Turner et al. [71] have proposed methods for learning the hazard function from the data. The method of Wilson et al. can learn a hazard function that is piecewise constant via a hierarchical generative model. Turner et al. can learn any parametric hazard rate via gradient descent, but from initial investigations appeared to not perform particularly well if the hazard rate is adapted at every time step. Although the more general hierarchical model could be investigated for our purposes a constant switching rate was assumed which was learned using the approach of Wilson et al. The hierarchical model could be used in the same general framework but is out of the scope of this thesis.

For the simplest case where we consider a single constant switch rate, Wilson et al. model whether a change point occurred as a Bernoulli variable. Using a Beta prior with hyper-parameters as the number of times the system has switched, $\mathbb{a}_t$, and has not switched, $\mathbb{b}_t$, we can infer the switch rate. We now compute the joint distribution $P(r_t, \mathbb{a}_t | x_{t-1}, D_{t-2})$ as opposed to the original distribution $P(r_t | x_{t-1}, D_{t-2})$. The message passing proceeds in a very similar fashion as before, except now the number of particles also grows quadratically rather than linearly. In the global switching model the algorithm now keeps track of sets of particles $w^t_{r,\mathbb{a}_t}$, $\alpha^t_{r,i,\mathbb{a}_t}$ and $\beta^t_{r,i,\mathbb{a}_t}$ associated with a runlength r, learning rate hyperparameter $\mathbb{a}$, arm i and time t. The updates are as follows.

$$
\begin{aligned}
w^0_{0,0} &= 1 \\
w^{t+1}_{r+1,\mathbb{a}} &= \frac{t-\mathbb{a}+1}{t+2} \frac{\alpha^t_{r,i,\mathbb{a}}}{\alpha^t_{r,i,\mathbb{a}} + \beta^t_{r,i,\mathbb{a}}} w^t_{r,\mathbb{a}} \text{if reward} = 1 \\
w^{t+1}_{r+1,\mathbb{a}} &= \frac{t-\mathbb{a}+1}{t+2} \frac{\beta^t_{r,i,\mathbb{a}}}{\alpha^t_{r,i,\mathbb{a}} + \beta^t_{r,i,\mathbb{a}}} w^t_{r,\mathbb{a}} \text{if reward} = 0 \\
w^{t+1}_{0,\mathbb{a}+1} &= \frac{\mathbb{a}+1}{t+2} \frac{\alpha^t_{r,i,\mathbb{a}}}{\alpha^t_{r,i,\mathbb{a}} + \beta^t_{r,i,\mathbb{a}}} w^t_{r,\mathbb{a}} \text{if reward} = 1 \\
w^{t+1}_{0,\mathbb{a}+1} &= \frac{\mathbb{a}+1}{t+2} \frac{\beta^t_{r,i,\mathbb{a}}}{\alpha^t_{r,i,\mathbb{a}} + \beta^t_{r,i,\mathbb{a}}} w^t_{r,\mathbb{a}} \text{if reward} = 0
\end{aligned}
$$

Wilson et al. also propose a resampling technique for the hierarchical case. However we again use the resampling algorithm of Liu and Fearnhead to manage the space requirements of the algorithm.

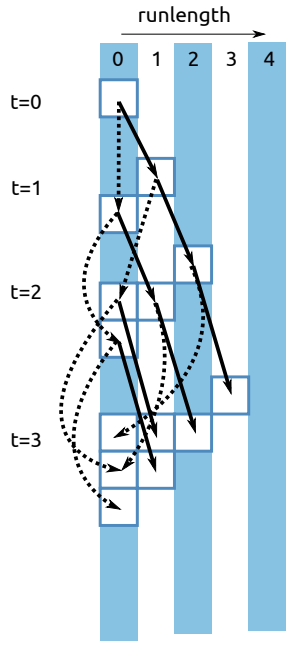


Figure 4.7: Message passing for inferring both a runlength and a constant switch rate. The dotted and solid lines representing messages when a switch does and does not occur respectively. It can be seen how the number of particles grows quadratically

In the Global Switching model there is only 1 runlength distribution, and so only 1 switching rate to learn. This leads naturally to an algorithm *Parameter-free Global Change-Point Thompson Sampling* (PF Global-CTS). With Per-Arms there are many possibilities: there could be a single switching rate for each of the independent arms, or each arm could have a separate switching rate. For flexibility as well as simplicity we assume each arm has a separate switching rate and call this algorithm *Parameter-free Per-Arm Change-Point Thompson Sampling* (PF PA-CTS). It is noted that it may well be possible to improve learning rates for the case of a shared switching rate, but this has not been further investigated.

4.7 Tracking changes in the best arm

The algorithms presented so far attempt to track changes in all arms, irrespective of whether they are pulled. For an arm not pulled, the data is not updated, but r_t is, which leads to a high variance. As a motivating principle in the development of Adapt-EvE, Hartland et al. argued that it is only important to track whether

the perceived best arm has changed. We can modify the algorithms to track the perceived best arm.

This is simplest in the Per-Arm Switching model. Since each arm is treated independently, we only update the runlength and hyperparameters of particles associated with the arm which was pulled. For the Global Switching model, this does not work, because the arms share a runlength model, which is updated at each pull. We need hyperparameters for unpulled arms at runlength 0. We use hyperparameter values associated with a pulled arm in the recent past, by putting those discarded during resampling in a queue. When the algorithm comes to set the prior parameters associated with runlength zero (in the **UpdateArmModels** section of Algorithm 4.2) α_{0,a_t}^{t+1} and β_{0,a_t}^{t+1} are set as before to α_0 and β_0 respectively. However for all arms not pulled, j, values for $\alpha_{0,j}^{t+1}$ and $\beta_{0,j}^{t+1}$ are pulled from the queue.

We can apply the same method from Wilson to infer the switching rate for these architectures as well. The algorithms with the modification described in this section will be denoted by having a "2" appended to the algorithm name (For example Global-CTS2 as opposed to Global-CTS).

4.8 Summary of algorithms

The general framework of using online Bayesian changepoint detection with Thompson Sampling was applied to two models of the environment (Global and Per-Arm), two assumptions about the switching rate (constant but known and constant but unknown), a heuristic modification was also investigated for tracking changes in only the best arm. This leads to a set of eight instances of the general algorithm. These instances are summarised in Table 4.1.

Table 4.1: A summary of the instances of Changepoint Thompson Sampling procedure that were evaluated.

	Known constant switch rate	Unknown constant switch rate
Global switching model (all arm reward distributions change together)	Global-CTS Global-CTS2	PF Global-CTS PF Global-CTS2
Per-arm switching model (arm reward distributions change independently)	PA-CTS PA-CTS2	PF PA-CTS PF PA-CTS2

4.9 Practicalities in estimating $P(x_{t-1}|r_{t-1}, D_{t-2})$

The Bayesian change detection message passing scheme is described in equation 4.6. It comprises several factors including the switching rate and the reward likelihood. The reward likelihood, $P(x_{t-1}|r_{t-1}, D_{t-2})$ is the probability of getting a reward x_{t-1} when the runlength is r_{t-1} from the past history of observations D_{t-2}. When an arm at time $t-1$ is considered to be a Bernoulli random variable with parameter p, as in our specific model, then $P(x_{t-1} = 1|r_{t-1}, D_{t-2}) = p$ and $P(x_{t-1} = 0|r_{t-1}, D_{t-2}) = 1 - p$. We store a belief distribution over the value of p for each runlength as the hyperparameters of a Beta distribution. In order to apply the message passing an estimate for p is required from the Beta distributions.

The Bayesian way to form an estimate for $P(x_{t-1}|r_{t-1}, D_{t-2})$ is to integrate over a model with respect to our beliefs. Our beliefs in this case are represented by a Beta distribution with parameters α and β. When we integrate over this distribution we find that the estimate is the mean of the Beta distribution, $p = \frac{\alpha}{\alpha+\beta}$. There is a potential practical problem with this approach when the payoff probabilities are close to 1/2 (assuming we set the prior distributions using the principle of indifference ,where $\alpha, \beta = 1$). To see this, let us consider an example of inferring the runlength distribution for a stationary process of rewards from a single arm. The arm rewards then come from a single distribution, with parameter p. In this scenario we would like the inference to give a high probability to larger runlengths. This will partly depend on the fixed switching rate that is assumed,

but for a given switching rate we would like most of the probability mass for the runlength distribution to be associated with runlengths that are as large as possible. Let us say the true parameter $p = 1/2$ then consider what happens to the runlength distribution as it evolves in time due to the inference algorithm. The reward likelihood associated with a runlength of zero in this scenario is $P(x_{t-1}|r_{t-1} = 0, D_{t-2}) = 1/2$. For any non-zero runlength the expectation of the reward likelihood is $\mathbb{E}\left[P(x_{t-1}|r_{t-1} \neq 0, D_{t-2})\right] = 1/2$ also. However, this is only in expectation, and there will be variance associated with it. The consequence of this is that shorter runlengths maybe "preferred" by the inference algorithm to longer runlengths. However, when the true parameter p is far from the mean of the prior Beta distribution (associated with a runlength of zero) the larger the runlength the closer the mean is to p (and with less variance in the estimate). In this case the inference will behave as we would like and the runlength distribution will favour higher runlengths (the meaning of higher being dependent on the switching rate).

There are some alternatives to the Bayesian approach described above for how we may choose this estimate. These include,

- Estimate p with the mode of the Beta distribution.
- Estimate p with a sample from the Beta distribution.

We will consider these alternatives to see if they have the same potential problem.

Estimate p with the mode of the Beta distribution

We might choose to estimate p by the maximum likelihood estimate, namely the mode of the Beta distribution. This approach has a problem however. If the observed rewards are such that $x_{t-1} \neq x_{t-2}$ then the reward likelihood, $P(x_{t-1}, r_{t-1} = 1, D_{t-2})$, associated with a runlength of 1, is equal to zero. This is also true for any runlength, r where the last r rewards are different to the current observed reward. Since the message passing when incrementing a runlength is multiplicative this can have the effect of making much of the runlength distribution zero, and hinder learning.

Estimate p with a sample from the Beta distribution

We can sample from the Beta distribution associated with an arm for a given runlength in order to get an estimate for p. This is much in the same way as we

get an estimate for the mean of an arm in Thompson Sampling. By sampling from the Beta distribution, the estimate for p will have high variance for small runlengths regardless of what the true value of p is. This may potentially mitigate the perceived problem with estimating the reward likelihood using the mean. However, conversely when the true value of p is far from the mean of the prior, the introduction of variance to the estimate of the reward likelihood for lower runlengths might be less efficient than using the mean as an estimate.

Sample or Mean estimate?

To investigate which method is to be preferred some experiments were conducted. Firstly both methods were used in inferring the runlength for a one-armed bandit with a switching Bernoulli reward distribution. The sample method and the mean method were compared in two scenarios. The first scenario is where the arm switched between Bernoulli distributions with parameters far from 1/2. In this experiment we expect the Mean estimate approach to perform better. The results of this experiment can be seen in 4.9. The experiment shows that the mean method is much more confident in the true runlengths than the sample method. The experiment was repeated but for a switching environment where switching occurred between Bernoulli distributions with parameters closer to 1/2. The results are shown in Figure 4.8. We can see that in this case that the sample method appears to be more confident in the true runlengths. However the differences are less stark than the first experiment where it is observed that the sample method is more confident that no changes at all have occurred in comparison to the mean method, when in fact changes have occurred. From this we propose that using the mean method (where we use the mean to estimate the reward likelihood) is more appropriate.

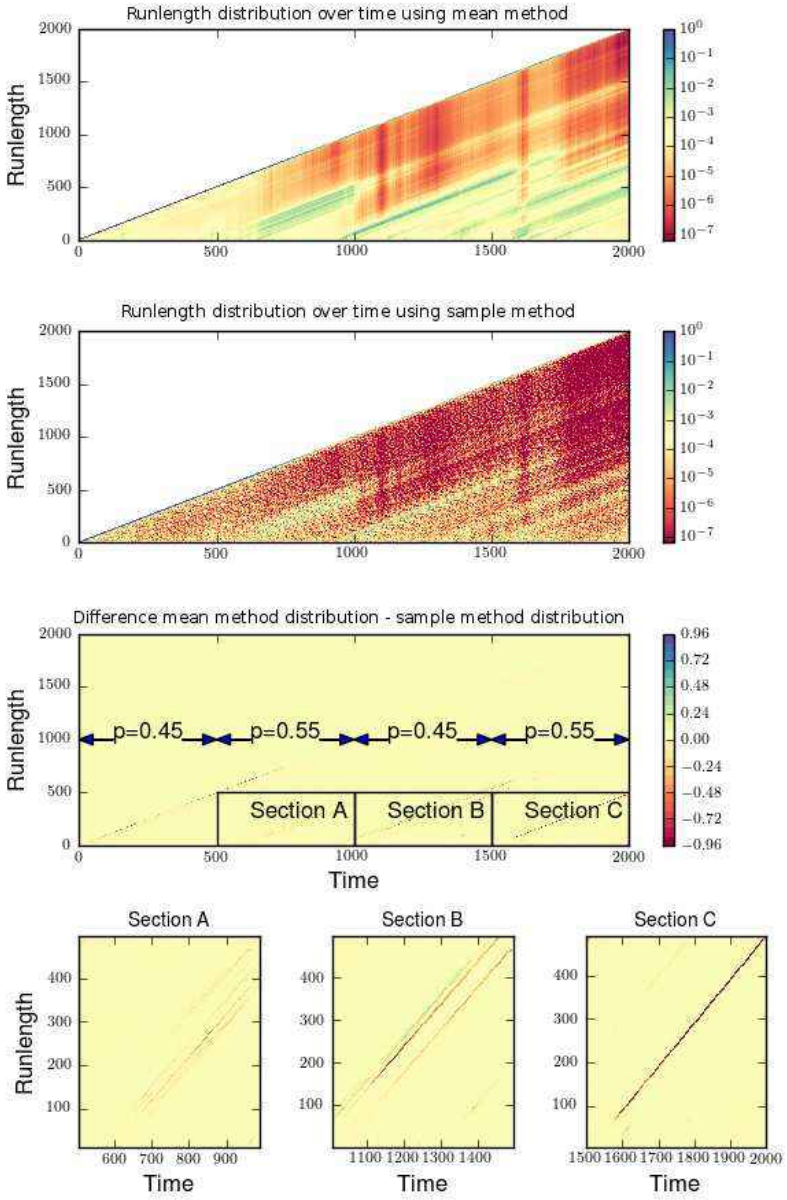


Figure 4.8: Two versions of the Bayesian change point algorithm were run in a switching environment. Let X_t denote the observed Bernoulli random variable at time t and $X_{t:\tau}$ denotes the set of variables observed between t and τ. X_t were drawn independently, with $X_{1:500}$ and $X_{1000:1500}$ drawn from the same Bernoulli distribution ($p = 0.45$) and $X_{500:1000}$ and $X_{1500:2000}$ drawn from another ($p = 0.55$). The top two plots show the evolution of runlength distributions over time for the two methods. We can clearly see both identifying switches. The next plot shows the difference in distribution of the two methods and mark where p changes. Section A,B and C are magnified portions of this plot. These were plotted to highlight the difference of the two methods. We can see that when the Bernoulli parameters switch in a range close to a half, the method that uses the Sampled estimate for the likelihood term in the message passing algorithm is more confident in a runlength close to the true value than the mean estimate of the same term.

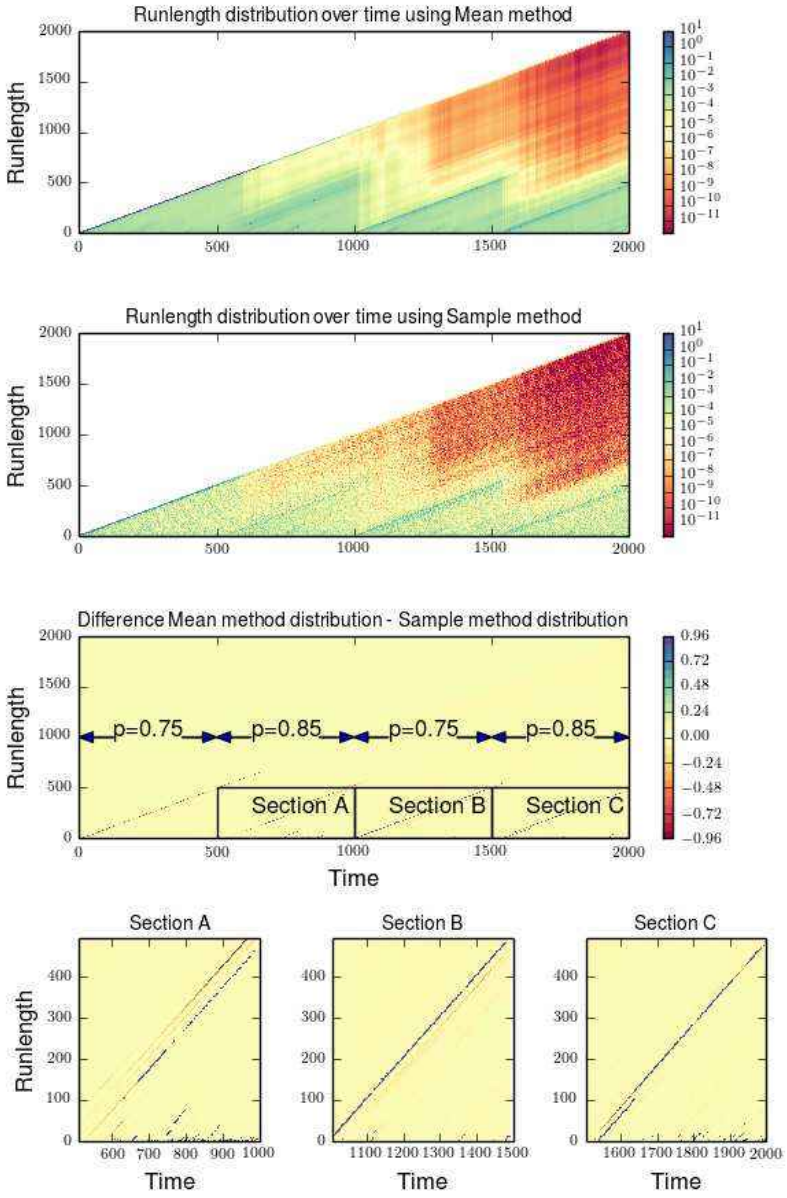


Figure 4.9: The same experiment as in Figure 4.8 were performed. The Bernoulli parameter was this time switched between $p = 0.75$ and $p = 0.85$. The gap between the two parameters remaining the same, but critically the values being further away from $p = 0.5$. Since we are choosing a uniform prior (Beta$(1, 1)$) for our belief in the p parameter then the mean estimate coincidences with this value for a runlength of 0. We expect if the true parameter is nearer to this value, learning is harder when using the mean estimate. Conversely when the true value is far away for this critical point it will perform better. We can see for the difference plot shows the mean method is more confident about the true runlengths (blue line in section A, B and C). This leads to the conclusion that when not in a regime near the critical value the mean estimate should be preferred.

4.10 Experiments: global switching model

We first compare the algorithms in an environment with a constant global switching rate. Global-CTS and PF Global-CTS were designed for this environment and so a-priori we would expect them to perform the best.

The first set of experiments were a single run of the algorithms working in an instance of this environment type with 2 arms. The payoff probabilities of the Bernoulli bandit were changed with a probability of $\gamma = 0.001$, the switch rate of the environment. The agents were run for a time horizon of $t = 10000$. The switch rate parameter for algorithms such as Global-CTS and PA-CTS were set to this value.

Figures 4.10 and 4.11 plot example heatmaps of the runlength distributions of some of the algorithms. At a particular time, the graphs show the runlength distribution. In the case of the PA-CTS and PF PA-CTS there are 2 plots for each algorithm, corresponding to the runlength distribution for each arm. The pay off of the 2 arms has been superimposed over the top of the plots so that it can be seen how the runlength distribution matches up with the changes in the environment.

From the heatmap figures we can see the change point prediction works when applied to a bandit problem. As expected the change point distribution looks to be more accurate for the Global-CTS and PF Global-CTS algorithms which use the Global Switching model, this is because each data point can contribute to the posterior runlength distribution. The PA-CTS also performs reasonably well even though the amount of data that has influence on each posterior is reduced.

For PF PA-CTS, learning the separate switching rates appears to significantly decrease the certainty for a particular runlength.

An experiment comparing the algorithms in this setting was performed. Each run was over a period of 10^6 time steps and the experiment was repeated 100 times. The results are displayed in Table 4.2. All parameters were set as for the PASCAL challenge test run. The environment constant switching rate was 10^{-4}, the same as the switch rate parameter for the algorithms.

Global-CTS performs the best in the environment, which is not surprising since the environment fits the algorithms model. PF Global-CTS performs well in this too, which suggests that learning the hazard rate for this model may be feasible.

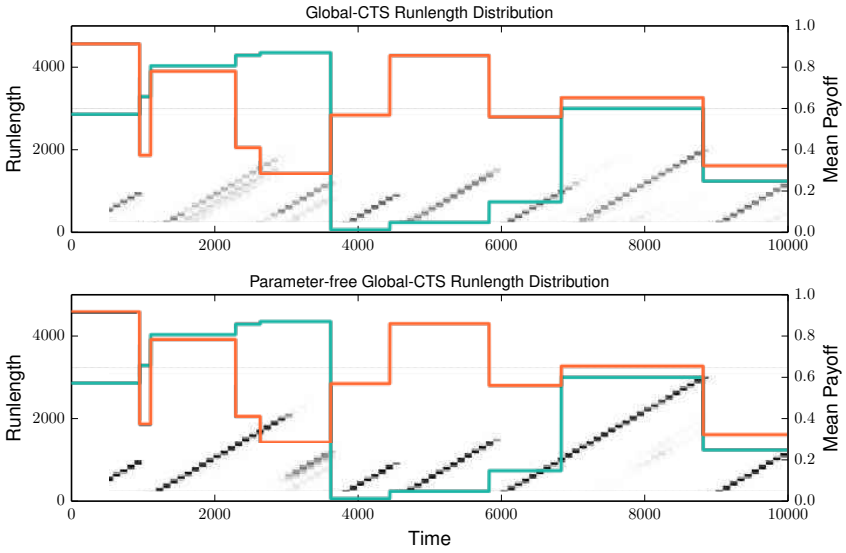


Figure 4.10: Runlength Distribution for Global-CTS and PF Global-CTS in Global Switching Environment. The mean payoffs of the arms are super-imposed over the distribution.

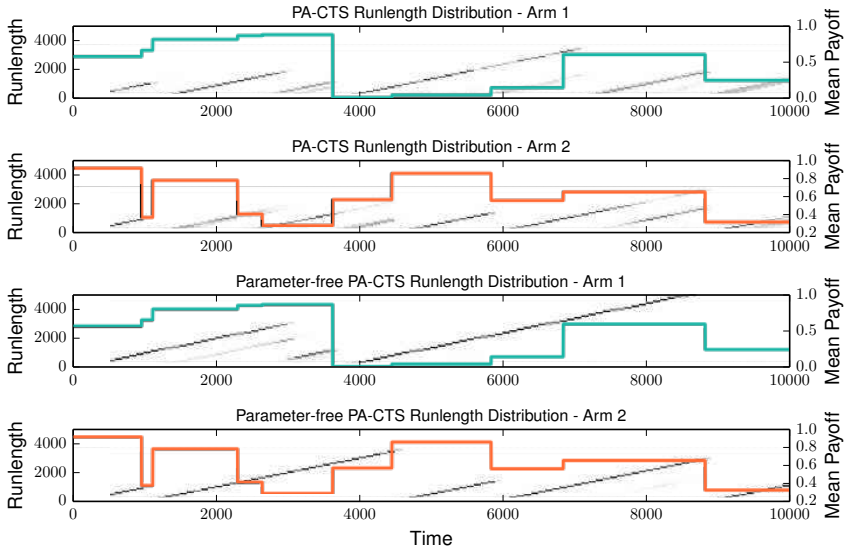


Figure 4.11: Runlength Distribution for PA-CTS and PF PA-CTS in Global Switching Environment. The mean payoffs of the arms are super-imposed over the distribution.

Table 4.2: Results against Global Switching Environment (given as number of mistakes$\times 10^{-3}\pm$ Std. Error)

Name	Regret	Name	Regret
Global-CTS	$\mathbf{5.9 \pm 0.07}$	Global-CTS2	30.5 ± 1.07
PA-CTS	12.1 ± 0.10	PA-CTS2	49.6 ± 1.70
PF Global-CTS	6.7 ± 0.08	PF PA-CTS	29.4 ± 0.95
PF Global-CTS2	10.3 ± 0.20	PF PA-CTS2	25.6 ± 0.86
UCB	178.3 ± 8.20	DiscountedUCB	15.5 ± 0.27

4.11 Experiments: Per-arm switching model

The next environment was a switching system where the switching for each arm was independent of every other arm. PA-CTS and PF PA-CTS were designed with this situation in mind and again a-priori may be expected to perform better. An experiment comparing the algorithms was performed with 10^6 iterations and then repeated 100 times. The results are shown in Table 4.3. As expected the PA-CTS algorithm performs best in this environment. PF PA-CTS, the algorithm corresponding to PA-CTS that learns the hazard rate suffers much more regret, which would appear to indicate for the particular model the parameters are not being learned quickly enough. The algorithms designed for a Global Switching environment also perform reasonably in this sort of environment.

Table 4.3: Results against Per-Arm Switching Environment (given as number of mistakes$\times 10^{-3}\pm$ Std. Error)

Name	Regret	Name	Regret
Global-CTS	13.8 ± 0.20	Global-CTS2	37.9 ± 1.02
PA-CTS	$\mathbf{13.0 \pm 0.11}$	PA-CTS2	67.1 ± 1.23
PF Global-CTS	13.8 ± 0.17	PF PA-CTS	30.8 ± 0.79
PF Global-CTS2	15.8 ± 0.28	PF PA-CTS2	38.1 ± 0.83
UCB	175.1 ± 7.47	DiscountedUCB	16.8 ± 0.28

The two algorithms that matched the models for the global and per-arm switching models were Global-CTS and PA-CTS respectively, plots showing the evolution of the probability that each strategy pulled the optimal arm are shown in Figure 4.12.

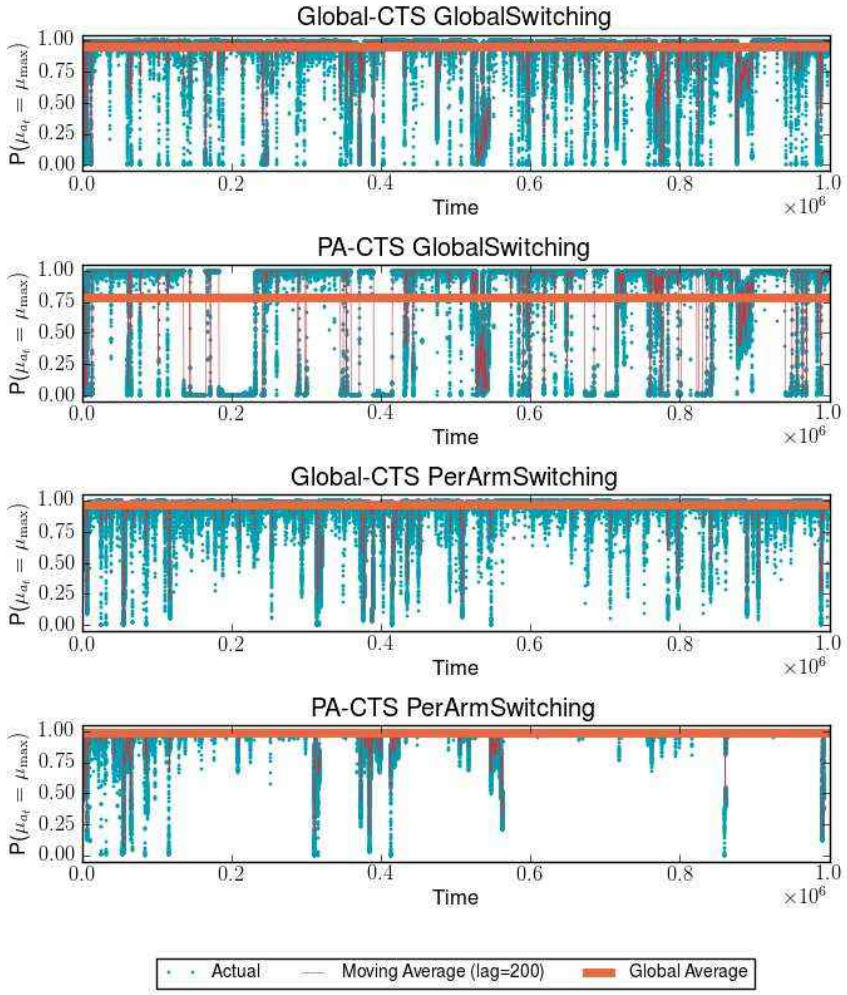


Figure 4.12: The probability that Global-CTS and PA-CTS pulled the optimal arm in single runs of the global and per-arm switching experiments.

4.12 Experiment: Bernoulli-armed bandit with random normal walk

Our model was shown to achieve good performance in environments known to be Bernoulli and switching. This was largely expected since this is the sort of environment for which they were designed. To investigate how our model responded to a more foreign setting, where the actual and assumed dynamics did not match, another simulated environment was investigated. In this environment

at time, t, each arm, i, was Bernoulli with probability of success $\theta_i(t)$. At each time step the success rate of the arm was allowed to drift as a truncated normal walk. That is the probability of success for an arm $\theta_i(t) \in [0, 1]$ conditional on $\theta_i(t-1) \in [0, 1]$ is,

$$P(\theta_i(t)|\theta_i(t-1)) = \frac{\exp\left(\frac{-(\theta_i(t-1)-\theta_i(t))^2}{\sigma^2}\right)}{\int_0^1 \exp\left(\frac{-(\theta_i(t-1)-x)^2}{\sigma^2}\right) dx}. \tag{4.13}$$

Table 4.4 shows a comparison of the algorithms. The experiment was run 100 times, where each run had a period of 10^6. The variance of the random walk was set to $\sigma^2 = 0.03$.

Table 4.4: Results against Bernoulli Bandit with Truncated Normal Walk (given as number of mistakes$\times 10^{-3}\pm$ Std. Error)

Name	Regret	Name	Regret
Global-CTS	97.9 ± 0.10	Global-CTS2	134.1 ± 0.23
PA-CTS	107.1 ± 0.16	PA-CTS2	148.9 ± 0.35
PF Global-CTS	116.6 ± 0.13	PF PA-CTS	117.0 ± 0.11
PF Global-CTS2	100.9 ± 0.10	PF PA-CTS2	$\mathbf{94.8 \pm 0.13}$
UCB	194.5 ± 3.78	DiscountedUCB	162.4 ± 0.47

In this sort of environment it appears that our algorithms perform better than the benchmark algorithms with PF PA-CTS2 achieving the smallest regret.

4.13 Experiments: PASCAL EvE challenge

The PASCAL Exploration vs. Exploitation Challenge 2006 was a competition in a multi-armed bandit problem [28]. The challenge revolved around website content optimisation, whereby the options available corresponded to different content to present to a user on a website. The challenge is a good general test for the algorithms presented in this paper as to perform well it was required for the bandit algorithms to be able to work in non-stationary environments. The challenge had 6 separate environments in which the algorithms needed to perform;

1. Frequent Swap (FS) - The best option in this environment would frequently swap.

2. Long Gaussians (LG) - The best option changed over long periods of time.

3. Weekly Variation (WV)- The response rates of options varied sinusoidally over 2 timescales, with the longer timescale dominating.

4. Daily Variation (DV) -The response rates of options varied sinusoidally over 2 timescales, with the shorter timescale dominating.

5. Weekly Close Variation (WCV) - Similar to Weekly Variation except the response rates are grouped closely together.

6. Constant (C) - The idealised stationary environment.

These environments are artificially generated, where the dynamics of the expected payoffs resemble either periodic Gaussian, Sinusoidal or constant signals. Figure 4.13 displays a selection of these environments to better visualise the way in which each environment changes. The bandit problem was a Bernoulli bandit problem, the payoff was either 0 or 1. The parameter determining the probability of success of the Bernoulli arm was the quantity that was varied in each of the six environments.

Hartland et al. won this competition with the Adapt-EvE algorithm [26]. The Adapt-EvE algorithm's most prominent feature is its use of the Page-Hinkley change-point detection mechanism. This is used to determine when to reset an underlying bandit algorithm UCB-Tuned, a variant of the UCB strategy. The details of the algorithm are discussed in Section 2.7.2.

Since the CTS algorithms also use a change-point mechanism it is interesting to compare their performance of them to Adapt-EvE. The challenge also provides an environment for which the algorithm was not directly designed and so will hopefully indicate some robustness in it's strategy. We were unable to implement a version of Adapt-EvE that replicated the performance reported by Hartland et al., so here we are simply replicating the results published.

Table 4.5 shows a comparison of the Change-Point Thompson Sampling algorithms (Global-CTS, PA-CTS, PF Global-CTS, PF PA-CTS) against Adapt-EvE Meta-Bandit and Meta-p-Bandit [26]. The comparison also features the algorithm "DiscountedUCB", which was submitted by Thomas Jaksch to the same competition and performed comparably to Adapt-EvE. The code for this algorithm was available and so has been included for comparison in all other environments. For

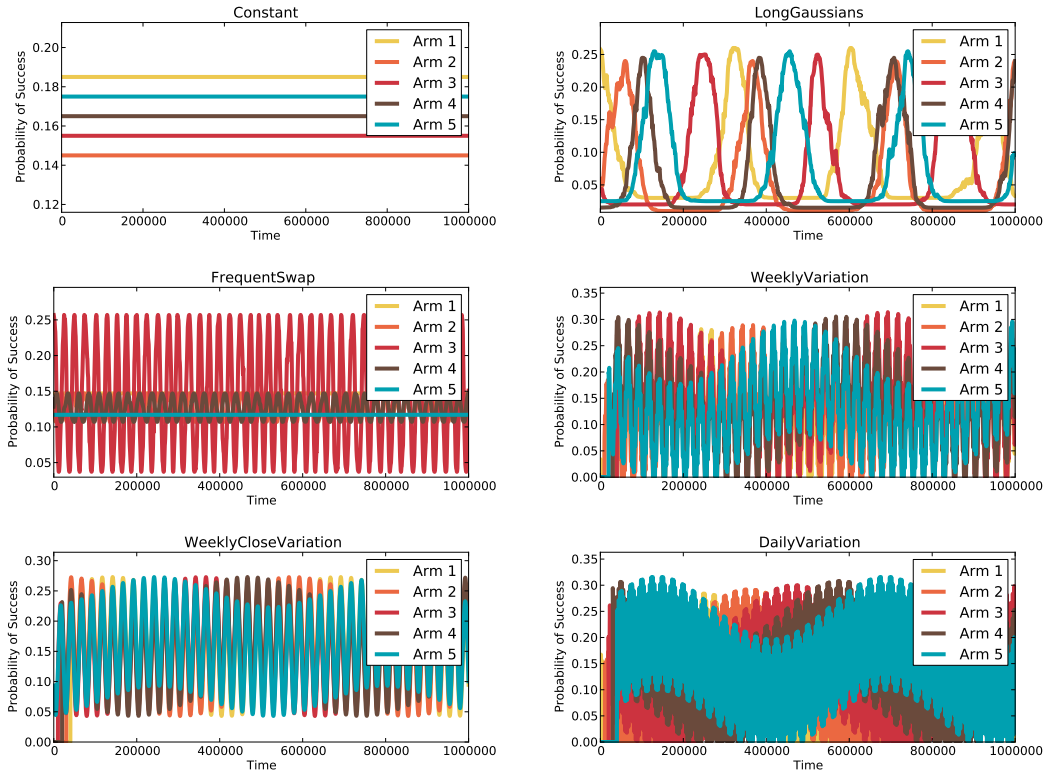


Figure 4.13: The PASCAL EvE Challenge 2006 test environments. The mean payoff is plotted as a function of time for the environments specified with 5 arms.

each environment in the PASCAL challenge we mark the lowest regret achieved by an algorithm in Table 4.5 in bold.

Here we can see that Global-CTS2 does the best out of our models, followed closely by both PA-CTS and PA-CTS2. The strategies do not quite perform as well as the results reported for Adapt-EvE but achieve remarkably low regret despite the PASCAL challenge being far from the exact model we had in mind when designing the algorithms. The PASCAL challenge environments have a periodic nature and drift rather than switch. The heuristic that Global-CTS2 uses, where Beta distributions from existing runlengths are used to set the priors for arms not pulled, may benefit from this periodicity.

Table 4.5: Results against PASCAL EvE Challenge 2006 (given as number of mistakes$\times 10^{-3}$)

	Global-CTS	Global-CTS2	Adapt-EvE Meta ρ
WCV	8.9 ± 0.4	6.9 ± 0.4	5.5 ± 0.9
FS	27.9 ± 2.4	12.5 ± 1.3	10.6 ± 1.3
C	$\mathbf{0.6 \pm 0.1}$	1.0 ± 0.2	3.2 ± 0.3
DV	17.1 ± 0.3	6.6 ± 0.3	6.1 ± 0.7
LG	4.4 ± 0.4	3.4 ± 0.5	4.3 ± 1.4
WV	8.2 ± 0.3	5.3 ± 0.5	5.1 ± 0.9
Total	67.2	35.8	**34.7**
	PA-CTS	PA-CTS2	Adapt-EvE Meta
WCV	$\mathbf{4.2 \pm 0.8}$	6.2 ± 0.4	5.4 ± 0.8
FS	13.7 ± 1.6	15.1 ± 1.7	14.0 ± 1.9
C	3.2 ± 0.4	2.0 ± 0.3	2.5 ± 0.5
DV	$\mathbf{4.5 \pm 1.5}$	4.9 ± 0.5	6.2 ± 0.7
LG	9.4 ± 2.9	3.7 ± 0.7	4.8 ± 1.6
WV	4.7 ± 1.7	5.4 ± 0.5	4.8 ± 0.8
Total	39.6	37.4	37.7
	PF Global-CTS	PF Global-CTS2	DiscountedUCB
WCV	8.9 ± 0.4	9.0 ± 0.3	5.3 ± 0.5
FS	28.2 ± 2.7	14.8 ± 1.2	$\mathbf{10.1 \pm 1.1}$
C	0.3 ± 0.2	0.8 ± 0.2	5.5 ± 0.5
DV	17.6 ± 0.3	16.0 ± 0.3	7.9 ± 0.9
LG	4.4 ± 0.4	4.0 ± 0.3	$\mathbf{2.9 \pm 0.4}$
WV	8.5 ± 0.3	8.4 ± 0.3	$\mathbf{4.0 \pm 0.4}$
Total	67.9	53.1	35.7
	PF PA-CTS	PF PA-CTS2	Random
WCV	12.8 ± 0.7	10.4 ± 0.4	25.7 ± 0.3
FS	23.1 ± 1.2	23.0 ± 1.9	49.1 ± 0.5
C	15.8 ± 0.4	1.9 ± 0.2	20.0 ± 0.1
DV	15.1 ± 1.0	24.2 ± 0.3	57.2 ± 0.3
LG	14.4 ± 2.1	8.2 ± 0.5	112.1 ± 9.1
WV	12.1 ± 1.1	11.7 ± 0.4	57.2 ± 0.3
Total	93.2	79.3	321.3

In the previous chapter we noted that, surprisingly, Thompson Sampling based on a Normal prior outperformed the Beta prior version for experiments using a Bernoulli bandit. To investigate if this observation carried across to a non-stationary environments we ran the same experiments PASCAL again, but this time with versions of Changepoint Thompson Sampling assuming rewards which were Normally distributed with known variance (set to 0.25). The findings are displayed in Table 4.6. Again the lowest regret for each of the six environments is marked in bold.

We find that in many instances the Normal prior is detrimental to the performance of the general strategy. However the normal variant of PA-CTS shows an increase in performance, so much so that it also outperforms the results reported for Adapt-EvE and DiscountedUCB. We suspect this may have to do with the rate at which a change becomes more probable. A change in the Global-CTS strategies has a larger impact on the entire strategy than with the Per-Arm model where a change is local to a given arm and so the estimates of other arms are unaffected.

Figure 4.14 shows the probability of pulling the best arm for the Normal PA-CTS strategy as a function of time for a single run of each environment in the PASCAL challenge. This can be compared to Figure 4.15 which shows the same for the DiscountedUCB strategy.

Table 4.6: Results against PASCAL EvE Challenge 2006 for Normal variants of CTS.

	Normal Global-CTS	Global-CTS	
WCV	10.5 ± 0.4	9.0 ± 0.4	
FS	29.7 ± 2.0	27.5 ± 2.2	
C	$\mathbf{0.6 \pm 0.1}$	$\mathbf{0.6 \pm 0.1}$	
DV	21.0 ± 0.3	18.8 ± 0.3	
LG	6.0 ± 0.4	4.9 ± 0.4	
WV	10.6 ± 0.4	9.1 ± 0.5	
Total	78.3	69.8	
	Global-CTS2	Normal Global-CTS2	DiscountedUCB
WCV	6.9 ± 0.4	7.6 ± 0.4	5.3 ± 0.5
FS	12.0 ± 1.2	14.0 ± 1.4	$\mathbf{10.0 \pm 1.1}$
C	1.0 ± 0.1	0.9 ± 0.1	5.5 ± 0.4
DV	7.1 ± 0.4	8.3 ± 0.3	7.7 ± 0.8
LG	3.9 ± 0.6	4.2 ± 0.5	$\mathbf{3.0 \pm 0.4}$
WV	5.5 ± 0.5	6.0 ± 0.4	4.1 ± 0.5
Total	36.4	41.1	35.6
	PA-CTS	Normal PA-CTS	
WCV	$\mathbf{3.8 \pm 0.6}$	3.9 ± 0.5	
FS	12.6 ± 1.9	11.3 ± 1.3	
C	3.1 ± 0.3	4.0 ± 0.3	
DV	3.5 ± 0.7	$\mathbf{3.6 \pm 0.5}$	
LG	8.9 ± 3.2	6.9 ± 1.8	
WV	4.0 ± 0.9	$\mathbf{3.7 \pm 0.7}$	
Total	35.9	**33.4**	

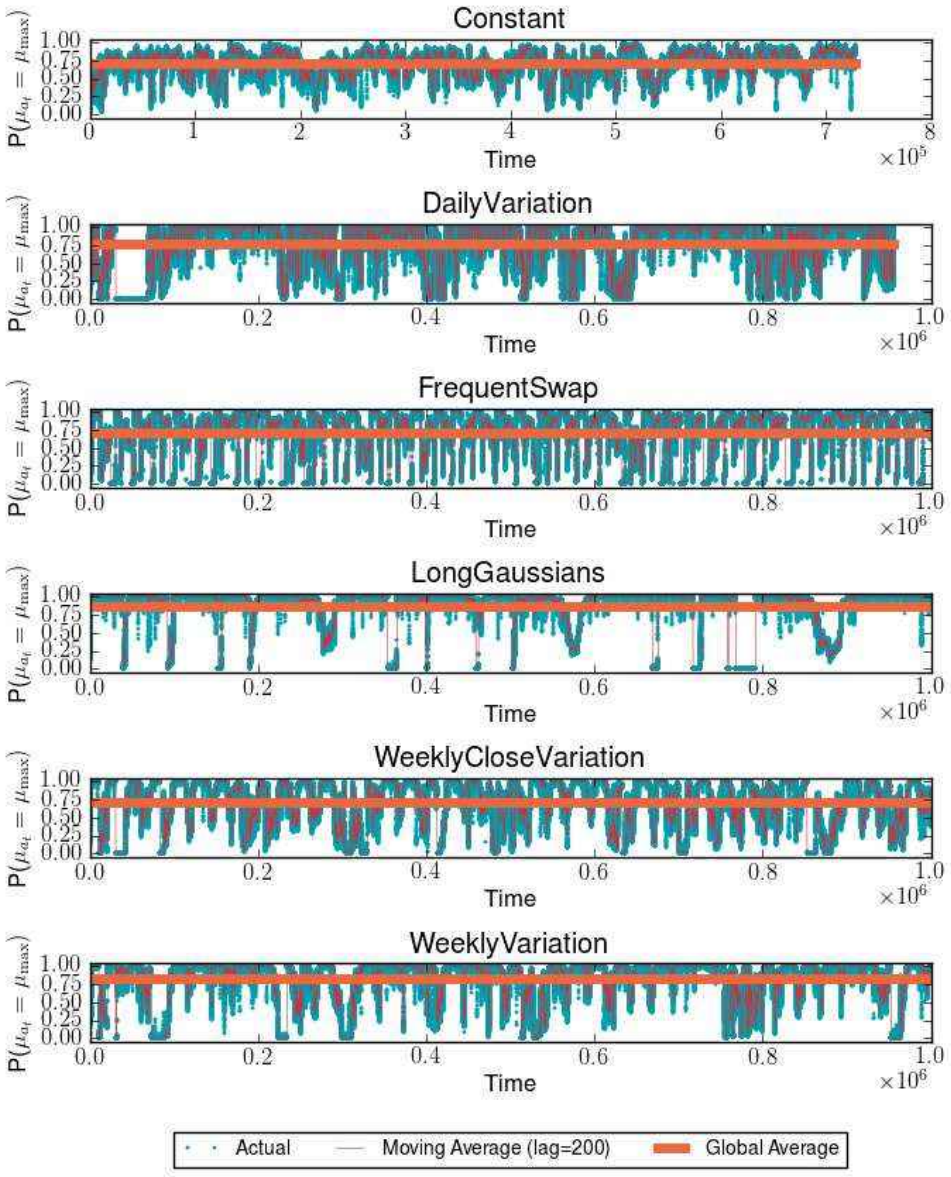


Figure 4.14: The probability of pulling the best arm for the Normal PA-CTS strategy over the course of a single run of the PASCAL challenge.

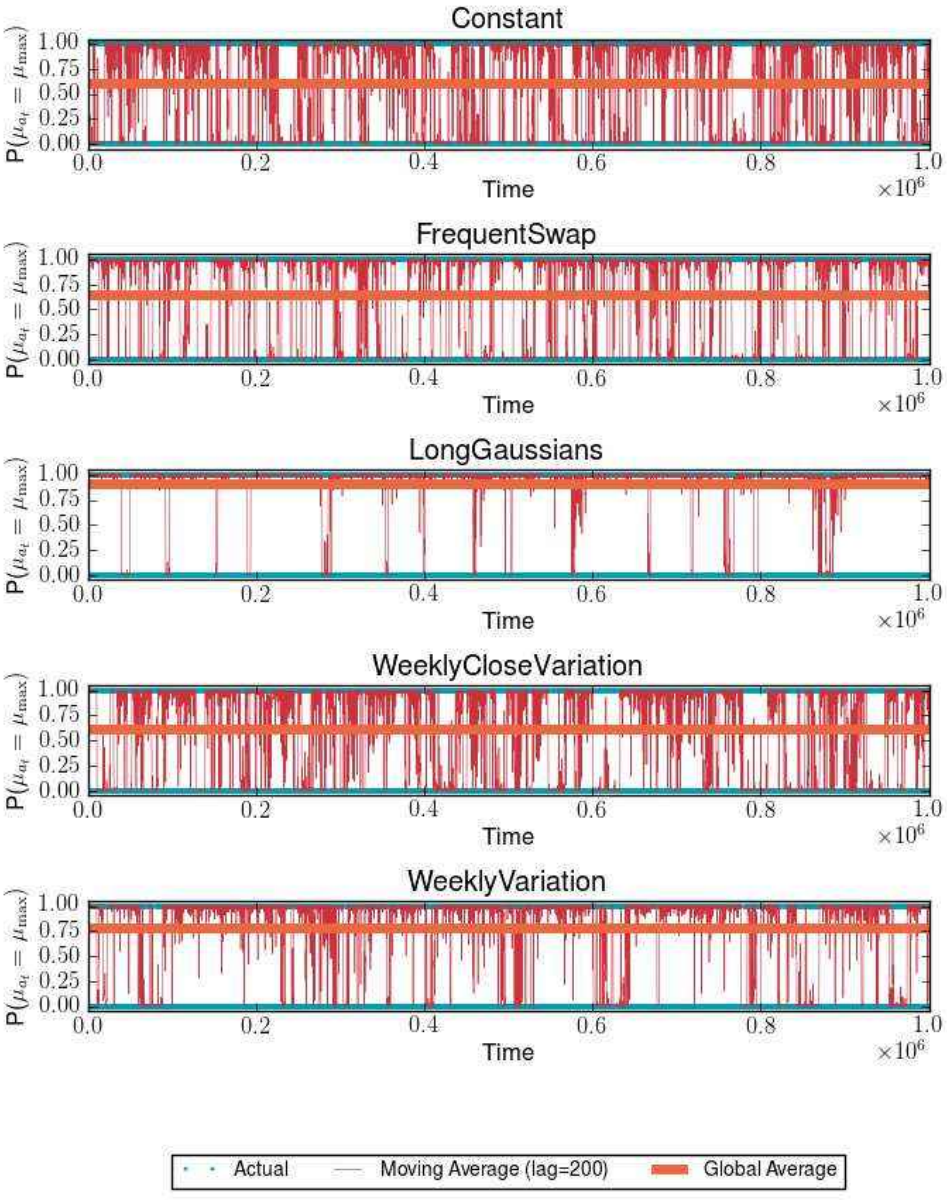


Figure 4.15: The probability of pulling the best arm for the DiscountedUCB strategy over the course of a single run of the PASCAL challenge.

4.14 Experiments: Yahoo! front page click log dataset

Yahoo! have produced a bandit algorithm dataset [77]. The dataset provides information about the top story presented to a user on the front page of Yahoo!. Each entry in the dataset gives information about a single article presented, the time it was presented, contextual information about the user and whether the user "clicked-through" to the article or not. The dataset was designed for the contextual bandit problem. Given context of a user the goal is to select an article to present to the user so as to maximise the expected rate at which users click on the article to read more (click-through). The articles also change during the dataset, and so bandit algorithms designed specifically for this environment also need the ability to modify the number of arms they can select from.

For the purposes of our experiments we do not concern ourselves with the contextual case, nor do we try to incorporate new articles as they arrive. Instead we ignore the context, and we only pick from a set number of articles. This reduces the problem to a conventional multi-armed bandit problem. To maximise the amount of data used, for each run we randomly selected the set of articles (in our case 5 articles) from a list of 100 permutations of possible articles which overlapped in time the most. The click-through rates were estimated from the data by taking the mean of an article's click-through rate every 1000 time ticks. The simulation then proceeded as described by Li et al. [46]. The results are presented in Table 4.7. The regret for each run was normalised by the number of arm pulls, since this was different in each run of the simulation. Parameters were set as for the PASCAL challenge dataset. This may explain the poor results of DiscountedUCB. It is likely the case it's performance could be greatly improved by tuning it's parameters. The CTS family of algorithms proved much more robust in this respect and coped in many different environments with the same parameter settings.

Table 4.7: Results against Yahoo! Front Page Click Log Dataset(±Std. Error)

Name	Regret	Name	Regret
Global-CTS	0.489 ± 0.035	Global-CTS2	$\mathbf{0.443 \pm 0.031}$
PA-CTS	0.522 ± 0.028	PA-CTS2	0.505 ± 0.028
PF Global-CTS	0.490 ± 0.029	PF PA-CTS	0.590 ± 0.018
PF Global-CTS2	0.530 ± 0.026	PF PA-CTS2	0.563 ± 0.018
UCB	0.526 ± 0.040	DiscountedUCB	0.568 ± 0.022

4.15 Experiments: Foreign exchange rate data

We constructed a final test environment from Foreign Exchange Rate data[18]. Ask prices for 4 currency exchange rates (GBP-USD, USD-JPY, NZD-CHF, EUR-CAD) at a resolution of 2 minutes spanning 7 years were used. This amounted to approximately 10^6 datapoints per exchange rate pair. The bandit problem using this data was setup as follows. Each exchange rate was thought of as a 2-armed bandit. It was imagined that the agent could make fictitious trades, and could either decide to buy a long call option (if they believe the rate will increase) and a short call option (if they believe the rate will go down). To turn this into a Bernoulli bandit problem, we ignore the scale of the change and provide a reward of 1 if the bandit predicted correctly the rate going up/down and 0 otherwise. When the rate remains the same, the agent receives a reward of 0 irrespective of their decision. For the purpose of the experiment we imagine the option length is 100 time ticks, so that the agent has to decide if the exchange rate will increase or decrease in 100 time ticks. Although this bandit scenario is not true to life, we believe that the underlying data should exhibit some of the characteristics of a switching system for which the algorithms were designed [59]. We can not estimate a "true" average payoff at each timestep, and so can not measure the regret of these algorithms, instead we report the error. The results are shown in Table 4.8.

Table 4.8: Results against Foreign Exchange Bandit Environment (number of mistakes $\times 10^{-3} \pm$Std. Error)

Name	Error	Name	Error
Global-CTS	351.9 ± 14.1	Global-CTS2	358.0 ± 13.95
PA-CTS	370.4 ± 13.7	PA-CTS2	380.9 ± 12.5
PF Global-CTS	$\mathbf{348.2 \pm 13.7}$	PF PA-CTS	353.5 ± 13.8
PF Global-CTS2	353.2 ± 13.4	PF PA-CTS2	352.0 ± 13.9
UCB	613.9 ± 17.7	DiscountedUCB	606.3 ± 16.0

4.16 Comparing Global-CTS to Exp3

The switching model of the non-stationary environment has some resemblance to the adversarial bandit setting that has been studied by Auer et al. [8]. They consider a regret that compares the performance of a strategy against some arbitrary sequence of events. They introduce a hardness measure to rank the difficulty of the task (see Equation 2.46 in Section 2.7.3). The harder the task it is to compete against a given arbitrary sequence, the higher the hardness measure is. The hardness measure for the adversarial multi-armed bandit problem roughly corresponds to the number of switches in the switching multi-armed bandit problem. Obviously they are not exactly the same since the switching system is still a well behaved stochastic process, and so the expected regret is an expectation over both the randomness present in the environment and the agent's strategy. Whereas, the expected regret for the adversarial is an expectation purely over the randomness present internally in the agent. There is no randomness present in the environment in the adversarial setting. A family of strategies were proposed by Auer et al. all prefixed by the name Exp3. In this section we briefly discuss the similarities between CTS and Exp3. Particularly we will consider Exp3 as defined in Algorithm 2.14 and Global-CTS for the case where its parameter $N = 2$.

4.16.1 Global-CTS with $N = 2$

We will consider the algorithm Global-CTS when the number of particles, N, is limited to 2 (therefore the only sensible value of M is 1). Here, by particle, we mean the set of hyperparameters defining the belief of the mean payoff of each arm for a single given runlength as well as the probability of that runlength. The algorithm is initialised with only a single particle (with a runlength of zero). The

algorithm then samples an estimate mean for each arm and pulls the arm with the highest estimate mean, as has been previously described. The environment supplies the algorithm with a reward. The algorithm then updates the belief in the arms mean payoff. This results in two particles, one whose hyperparameters have been updated by the new observation (and whose runlength has increased by one), and the other, a new particle, representing a runlength of zero with hyperparameters initialised to the prior values (for example $\alpha, \beta = 1$). However $N = 2$ which means that the algorithm will now resample using SOR to reduce the number of particles from two to one. When $N = 2$ the SOR algorithm discards the new particle, representing a runlength of zero, with probability $1-\gamma$, and discards the "old" particle with probability γ. This is because the unique solution α in Equation 4.7 is equal to one. Both particles are then candidates for resampling with their associated probabilities. The algorithm then samples a new estimate for each arm using the hyperparameters from the one remaining particle. This cycle then repeats as time proceeds. A flow diagram representing this operation is shown in Figure 4.16. The algorithm is further summarised in Algorithm 4.3.

Algorithm 4.3 Global Changepoint Thompson Sampling ($N = 2$)

Let $\alpha_{\text{old},k}, \alpha_{\text{new},k} = 1$,
    $\beta_{\text{old},k}, \beta_{\text{new},k} = 1$ for $k \in \{1, \ldots, K\}$.

for $t = 1, \ldots, T$ **do**
  With probability γ,
    Sample $\theta_i \sim \text{Beta}(\alpha_{\text{new},i}, \beta_{\text{new},i})$, for $i \in \{1, \ldots, K\}$.
    Let $\alpha_{\text{old},i} = \alpha_{\text{new},i}$
      $\beta_{\text{old},i} = \beta_{\text{new},i}$, for $i \in \{1, \ldots, K\}$.
  Otherwise,
    Sample $\theta_i \sim \text{Beta}(\alpha_{\text{old},i}, \beta_{\text{old},i})$, for $i \in \{1, \ldots, K\}$..
  Pull arm $a_t = \text{argmax}_i\, \theta_i$
  Let $\alpha_{\text{old},a_t} = \alpha_{\text{old},a_t} + \mathbb{1}(x_{a_t}(t) = 1)$
    $\beta_{\text{old},a_t} = \beta_{\text{old},a_t} + \mathbb{1}(x_{a_t}(t) = 0)$
  Let $\alpha_{\text{old},j} = \alpha_{\text{old},j}$
    $\beta_{\text{old},j} = \beta_{\text{old},j}$ for $j \in \{1, \ldots, K\} \setminus \{a_t\}$.
end for

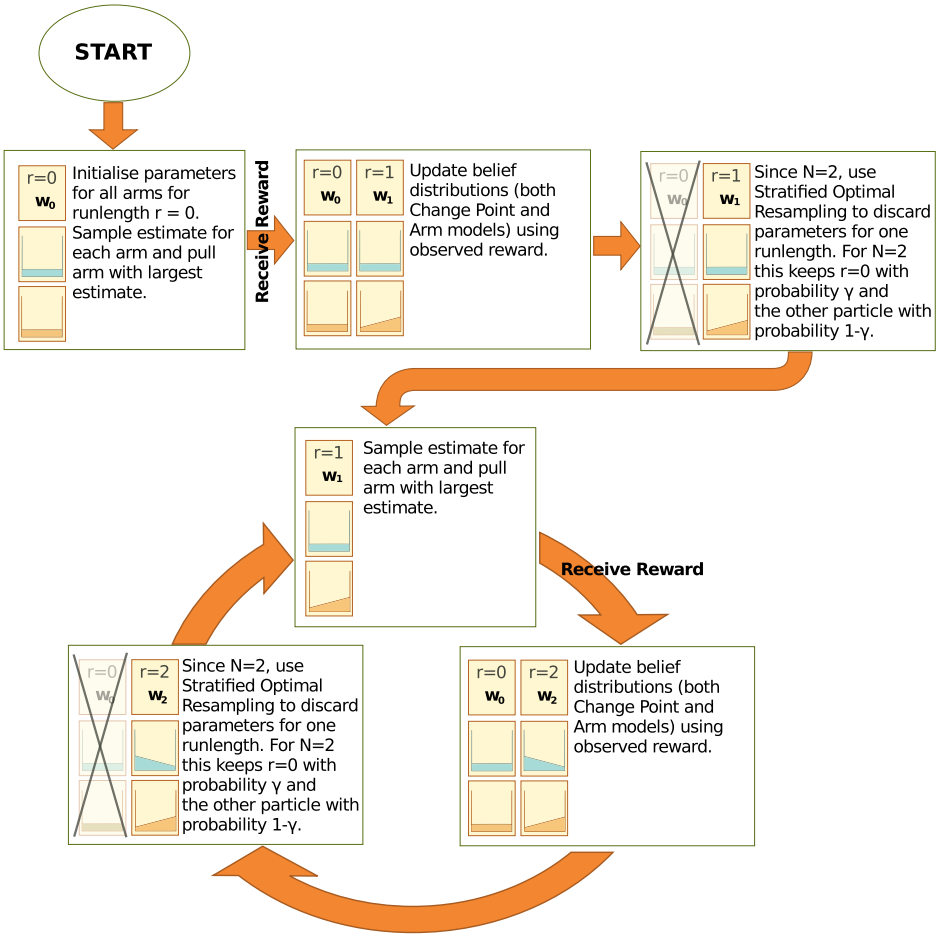


Figure 4.16: A flow diagram showing the stages of sampling, updating and re-sampling for Global-CTS when $N = 2$.

4.16.2 Comparison to Exp3

Remembering the definition of Exp3 defined in Algorithm 2.14, the agent chooses an arm i with probability $p_i(t)$ defined as,

$$p_i(t) = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\gamma}{K}, \tag{4.14}$$

where $w_i(t)$ is a weight defined by the algorithm. The $\{p_i(t) : 1 < i < K\}$ form a distribution from which we can sample in order to pick an arm to pull. An alternative to sampling directly from $\{p_i(t) : 1 < i < K\}$ is to form three auxiliary distributions, $\pi_{\text{mix}} = \text{Bernoulli}(\gamma)$, $\pi_{\text{old}}(i) = \frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)}$ and $\pi_{\text{new}}(i) = \frac{1}{K}$. To sample equivalently we first draw from π_{mix}, if 0 is drawn then sample from π_{old} and

otherwise sample from π_{new}. In the same way when $N = 2$ Global-CTS can be describe as a sampling procedure from three auxiliary distributions. The first, π_{mix} is equivalent to Exp3. The second, π_{new} is also equivalent (when $\alpha_{\text{new},i}, \beta_{\text{new},i} = 1$ as in algorithm 4.3), since the procedures pick each arm equiprobably. The third term π_{old} is not equivalent as it is defined by a Thompson Sampling procedure from Beta distributions, $\text{Beta}(\alpha_{\text{old},i}, \beta_{\text{old},i})$ rather than a distribution defined by the terms $\frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)}$. There are some striking similarities however, despite them not being equivalent. In Exp3 the weights, $w_i(t)$, are updated by an exponential factor, $\propto e^{x_t/p_i(t)}$. The larger the probability of pulling the arm, the smaller the change to the distribution defined by $\frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)}$. For Thompson Sampling the updates to the Beta distributions on an observations are also exponential, and share the property that the update factor is inversely dependent on the increase in probability that the arm is pulled on subsequent decisions.

To empirically demonstrate the perceived similarities in the two algorithms we compared both algorithms in an adversarial bandit setting. The bandit had 5 arms and was given a hardness of 5 (corresponding to the "best" a posteriori strategy being allowed to change its pure strategy 5 times). The results of the experiment are presented in Figure 4.17 which shows the adversarial regret of the two algorithms as a function of time. The graph shows the similar trends of the algorithms during the decision-making process.

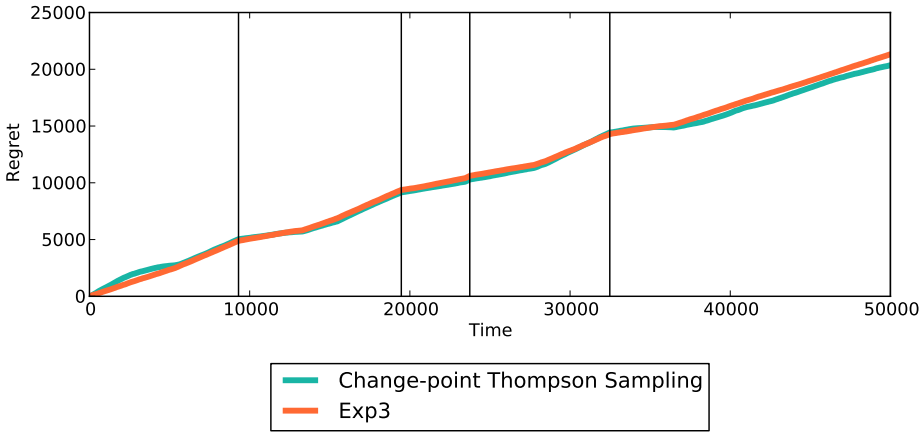


Figure 4.17: This figure shows a comparison between Global-CTS and Exp3. The hardness of the bandit problem was 5. The bandit problem had 5 arms. The parameter γ for both algorithms was set to same value.

4.17 Related work

4.17.1 Kalman Bayesian Learning Automaton

Granmo and Berg developed what the call the *Kalman Bayesian Learning Automaton* [25]. This is in effect Thompson Sampling where the posterior mean reward distribution is modelled to be a drifting environment. The rewards at a given time are assumed to be distributed Gaussian, and at each time step the mean of the Gaussians from which the rewards are drawn drift via a Gaussian random walk. To be more precise at time t the bandit of interest has K arms. The rewards of arm k are drawn from a Gaussian with mean $\mu_{k,t}$ and variance $\sigma^2_{k,\text{obs}}$. As time evolves from t to $t+1$ the mean reward of the arm changes such that $\mu_{k,t+1} = \mu_{k,t} + \varepsilon_t$ where $\varepsilon_t \sim \mathcal{N}(0, \sigma^2_{k,\text{drift}})$. The initial means $m_{k,0}$, the reward variances $\sigma^2_{k,\text{obs}}$ and the variances, $\sigma^2_{k,\text{drift}}$, of the Gaussian random walk thus define the dynamics of the environment. Both the reward variance and the variance of the random walk are assumed to be known. The Kalman filter is a Bayesian inference technique that leads to the optimal estimate of mean rewards in such environments [39]. The algorithm places a prior belief distribution on the mean of an arm at time 0 ($\mu_{k,0}$). At each step as time progresses from t to $t+1$ the Kalman filter infers a new belief distribution for $\mu_{k,t+1}$. There are two stages to the inference in a Kalman filter. These are,

1. Inferring the belief $P(\mu_{k,t+1}|\mu_{k,t})$ via the existing belief $P(\mu_{k,t})$ and the dynamics of the system.

2. Updating the belief distribution over $\mu_{k,t+1}$ using new observations x_{t+1} (i.e. $P(\mu_{k,t+1}|x_{t+1})$).

When the noise and dynamics are both Gaussian if the prior distribution is modelled as a Gaussian both steps can be combined into one update resulting in a new Gaussian belief distribution. This is due to the Gaussian distribution being its own conjugate prior. Let $P(\mu_{k,t}) = \mathcal{N}(m_{k,t}, s^2_{k,t})$ be the belief in the mean of arm k at time t. After receiving a reward x_t the belief in the mean at time $t+1$,

$P(\mu_{k,t+1})$, is $\mathcal{N}(m_{k,t+1}, s^2_{k,t+1})$ where,

$$m_{k,t+1} = \frac{(s^2_{k,t} + \sigma^2_{k,\text{drift}})x_t + \sigma^2_{k,\text{obs}} m_{k,t}}{s^2_{k,t} + \sigma^2_{k,\text{obs}} + \sigma^2_{k,\text{drift}}} \tag{4.15}$$

$$s^2_{k,t+1} = \frac{(s^2_{k,t} + \sigma^2_{k,\text{drift}})\sigma^2_{k,\text{obs}}}{s^2_{k,t} + \sigma^2_{k,\text{obs}} + \sigma^2_{k,\text{drift}}} \tag{4.16}$$

$$\tag{4.17}$$

4.17.2 Context dependent Thompson Sampling

The Kalman filter model proposed by Granmo et al. was for a fundamentally different type of environment than Changepoint Thompson Sampling. Simultaneously to our own work Lloyd et al. also considered applying Thompson Sampling to a model that was switching [47]. The motivation of their work stems from psychology. They wished to develop a simple mathematical model that mimicked some of the behaviours observed in animal decision making. Some traits that were deemed to be captured inadequately by previous models arise in the serial reversal-learning paradigm. The serial reversal-learning phenomena can be explained by way of the following example. Imagine a rat is placed in a simple T shaped maze. The rat is made to make a series of decisions in a number of rounds. In each round the experimenter places a reward in the form of food in either the left or the right arm of the T maze. If the experimenter initially only places the food in the left arm the rat will after several rounds learn to go directly to the left arm. However after some time the experimenter might switch where the food is being placed to the right arm. On the event of the change the rat will initially head to the left arm, until it eventually learns to switch to the right arm. Eventually after many cycles of the experimenter switching where the food is placed the rat will learn this switching behaviour and switch the arm they choose to head to more quickly based on the recent observations of where food was placed. The rat can get to the point where even a single round where the food is switched is enough to make the rat change its strategy.

They term contiguous periods, in which the observed rewards of actions follow the same statistical law, as *contexts*. In their model the contexts are not directly observable. Often in the bandit problem literature involving contexts the context is thought of as the extra observable information available to an agent other than the reward signal, which can be used to establish in which state the environment

is , and so allow the agent to associate different behaviours with different states. In this setting context instead refers to the hidden state for which the current distribution of rewards is dependent on. The context can thus only be inferred via the rewards observed by the agent. There are assumed to be an unknown number of contexts, the environment then switches between them, sometimes switching to a previously unseen context and other times returning to a past observed context. The generative model for this type of model is assumed to follow a *Chinese Restaurant Process* (CRP) [58]. The name is motivated by thinking of the process as trying to seat diners in a Chinese restaurant, an analogy credited to Pitman and Dubins by Aldous [4]. Let c_t be from some set of contexts $\mathcal{C}$. For each context there is a stationary distribution of rewards associated with each arm with mean. Let $\mu_k(c_t)$ denote the mean reward of arm k in context c_t (the context at time t). Then dynamics of the environment is then defined by the evolution of the contexts. The dynamics of the contexts are defined as follows,

$$c_t = \begin{cases} c_{t-1} & \text{with probability } 1-\pi \\ c \sim \text{CRP}(c_{1:t-1}; \alpha) & \pi, \end{cases} \tag{4.18}$$

where $c_{1:t-1}$ is the contexts observed up until time $t-1$ and α is the parameter of the Chinese Restaurant Process from which the new context is drawn. The dynamics are slightly different than a standard CRP since with probability $1-\pi$ the previous context is chosen. This allows the model to have contiguous periods where the context remains the same.

They propose a Thompson Sampling procedure for the above environment. The procedure maintains a distribution maintaining the probability of which context the process is currently in, and for each context the distributions associated with what the arms mean rewards are. To form a decision the context distribution is sampled in order to form an estimate of which context the environment is currently in. Then a sample from the arm distributions associated with the estimate context are drawn, one for each arm to estimate the mean reward. The arm with highest associated estimate is then pulled. Their model differs from our work in that they assume an switching environment where the switching occurs between contexts. A context may be revisited many times. In our model we assume that on a change that past observations are independent from future ones, and so only attempt to track when these changes occur. The added requirements to form a correspondence between a previous regime and the current one adds

extra complexity to the model. In some applications the changing will be of this repeating nature. In this case the model of Lloyd et al. could indeed be an improvement over our own. However although the model may in some cases gain added efficiency due to its ability to incorporate more past data than our own model (causing estimates of expected rewards to be sharper), in other cases the opposite may be true. There is a danger that the model may associate a new regime or context with a previous one when in fact they are different. In this case the decision making would be biased and over-confident potentially making the wrong decisions. It is unclear and an open question to find under what conditions remembering past contexts is advantageous and when our own model, of forgetting the past, is a more sensible strategy.

Let $n(c,t)$ be the number of times context c has occurred up to time t and let $N(c_{1:t})$ denote the number of unique contexts that have occurred until time t. The Chinese Restaurant Process that selects a context is defined as,

$$\text{CRP}(c_{1:t-1};\alpha) = \begin{cases} c \in c_{1:t-1} & \text{with probability } \frac{n(c,t-1)-\alpha}{t-1} \\ c \notin c_{1:t-1} & \text{with probability } \frac{\alpha N(c_{1:t-1})}{t-1}, \end{cases} \tag{4.19}$$

where $0 < \alpha < 1$. Here we can see that the probability of a new unseen context ($c \notin c_{1:t-1}$) shrinks as time continues. This is in contrast to our model where when a switch occurs the probability of an unseen context is certain. It may be possible to trade off between the merits of both our own environment model and this one by generalising both. We can do this by way of an added parameter to the environment model, $0 \leq \lambda \leq 1$. The environment model would then become,

$$c_t = \begin{cases} c_{t-1} & \text{with probability } 1-\pi \\ c \sim \text{CRP}(c_{1:t-1};\alpha) & \pi(1-\lambda) \\ c \notin c_{1:t-1} & \pi\lambda. \end{cases} \tag{4.20}$$

Such a model has not been explored further and has been left as potential future work.

4.17.3 Bayesian Bandits with Resets

Other Thompson Sampling models which has since been proposed for the same switching environments for which we developed our algorithms are by Viappiani [73]. The work considers the independently considers the global switching model,

which Viappiani calls "Bayesian Bandits with Resets". They consider two models, one a particle filter approach and the other they term Geometric-Beta (reset-aware) Thompson Sampling.

Particle Filter Thompson Sampling

Whereas our solution performs an exact inference step, followed by an approximating resampling step, the particle filter model of Viappiani calculates the posterior via simulation. In the particle filter model each arm has a number of particles associated with it. Each particle q for a given arm represents a possible mean for that arm. Initially these are drawn from the prior Beta distribution (normally the parameters are such that this is a uniform distribution). At each round several steps are performed. Firstly, for each arm, each particle is reset (meaning redrawn from the prior distribution) with the probability p_{reset}, the rate at which arms reset or switch. The algorithm then samples a particle for each arm, pulling the arm with largest sampled particle. A reward is then observed for the pulled arm. The likelihood of each particle can be found, being q if the reward is 1 and $1 - q$ if the reward is 0. These likelihoods for each particle of the pulled arm are used as weights in an importance sampling step. Particles that were more likely based on the last observation are more likely to be resampled. The estimation of the posterior is likely to be much higher variance than our own model, since the approximation in our approach only comes from the resampling the runlength distribution in a optimal way (we retain the Beta hyperparameters for kept runlengths). The particle filter method is implicitly approximating both the Beta distributions and the runlength distribution with no guarantee of optimality in either case. To perform well it is likely to require large numbers of particles, increasing time and space requirements.

Geometric-Beta Thompson Sampling

The second algorithm that Viappiani presents is called Geometric-Beta Thompson Sampling. Conceptually the full history of pulls and rewards are kept. A geometrically distributed random variable is drawn for each arm parametrised by the probability of error, this samples an estimate of when a reset or switch occurred for each arm. For each arm only the rewards since the estimated last reset are used to form the hyperparameters for the Beta distribution, which in turn is modelling the belief in a mean of the arm. A sample is taken from the

Beta distribution to be the estimate of the mean of the arm for the current round. As with Thompson Sampling the arm with highest mean estimate is then pulled. The first observation of this method is that it completely ignores the data in deciding when a switch has occurred. This certainly reduces the amount of computation required but it is also clearly a disadvantage in choosing the correct runlength of data. For instance in an extreme case, imagine an arm had produced only rewards of 1 from pulls 1 to 100 and then for the next 10 pulls produces rewards of 0. Geometric-Beta Thompson Sampling would ignore this evidence of a reset or switch whereas Changepoint Thompson Sampling would incorporate the evidence and more likely based its estimate only on recent data.

4.18 Conclusion

In this chapter we argue that in order to make multi-armed bandit strategies more applicable to many real world applications then we must design them to perform well in a changing non-stationary environment. We suggest that there are two main types of change that can occur, drifting and switching. We offer or cite some examples of where we expect to see switching behaviour such as in financial applications and game playing. We argue that a switching environment is a useful model from which to design bandit-like algorithms that are adaptable in more realistic settings.

The main contribution of this chapter is a class of algorithm we term collectively as Changepoint Thompson Sampling. These are a set of Thompson Sampling algorithms for the switching multi-armed bandit problem. We empirically demonstrate the effectiveness of the strategies in a wide range of bandit problems. Some of the problems exhibit the switching for which our strategies are designed, while others exhibit behaviour more akin to drifting. The algorithms perform well in comparison to other benchmarks, and have reasonable computational cost.

Chapter 5

Thompson Sampling for the Best Arm Identification Problem

In order to make good decisions an agent must have or acquire information about each possible choice. We have seen how the stochastic multi-armed bandit problem is one useful model for decision making in an uncertain environment. The goal in the problem is to minimise the cumulative regret of the agent. The cumulative regret being the expected difference in reward accumulated by the actual strategy of the agent and the strategy that plays only the arm with the largest expected payoff. In the multi-armed bandit problem the information must be acquired in real-time while the agent is making decisions. The agent must balance the need to get the largest instantaneous reward they can, by choosing the arm they think is best (exploiting), and gathering information about the true expected payoff of uncertain arms in order to improve the chances of making better future decisions (exploring). Since the regret is cumulative the outcome of every decision directly contributes to the regret the agent feels. In this way there is a cost associated with each possible action related to its expected reward. For many situations it is sensible to have this type of cost associated with the outcome of each decision, for instance in Thompson's example of clinical trials, we care about the outcome of each patient in the trial, not just about the improvement of future treatments after the trial has concluded. However, there are other situations where this is not true. There may be no cost directly related to the outcome of a single decision. In these situations, the agent can set aside a period of exploration in which it can try different choices and gain information about their benefits. Due to the limited time of the exploration and the stochastic nature of a decisions

consequences, the agent can not fully determine facts related to their problem. The agent must then budget their opportunity of exploration and focus their resources to learn the most salient information in relation to their eventual decision under these constraints. The *best arm identification problem* provides a simple model in which to study this dilemma. The problem can be used to model for example network optimisation problems [27], product testing within marketing, and in connection with AB testing [37].

For example, we can imagine a company wishing to release a new product to the market. The more demand for the product the more money they are likely to make. So before releasing the product they decided to do some market research to gauge the market demand. They have several prototype versions of the product and want to know which will invoke the largest demand from customers. Due to constraints on the large scale manufacture of products it only makes sense for one of the prototypes to be mass produced and made available on the consumer market. The company decide to make batches of prototypes on a small scale to provide to a sample of the consumer population in order to ascertain which product is preferred. The company may have a fixed budget for their market research, and thus have a fixed number of prototypes they can build before they have to decide on the final product. With the decision of how many of each prototype to make there is no direct cost, in the sense of loss in future profit, for making a sample of a sub-optimal prototype. As long as the company increases their chances of identifying the best of their prototypes before going to market there is no loss associated with the decision to build it. If we imagine that the company decides to build prototypes sequentially, building a single item and testing it before deciding which kind of prototype to make an item of next, then the problem can be modelled as the *Best Arm Identification Problem*.

In the best arm identification problem there are a set of possible arms K. Each arm $k \in K$ when pulled at time t provide a reward $x_k(t)$. The expected reward for arm k is μ_k. An agent A is given T trials in which to pull these arms before making a final decision $\Psi(T) \in K$ on which arm has the highest expected payoff. There is therefore an exploration phase for T trials followed by a final decision.

Many algorithms for this problem have been studied, including: uniform allocation [13], UCB style algorithms adapted for best arm identification [5], racing style algorithms [5] and, most recently, gap based algorithms [21, 27].

The contributions of this chapter are two types of algorithms using Thompson Sampling, and a proposal of a metric by which to describe the behaviour of a best-arm identification algorithm. Specifically our contributions are,

- *Order-Statistic Thompson Sampling* - A class of algorithm that naturally extends Thompson Sampling to be suitable for the Best Arm identification problem.
- *Maximum Boundary of Pairs* - An parameterless algorithm that makes use of Thompson Sampling.
- *Measure of Aggression* - A proposed measure to categorise the behaviour of the Order-Statistic Thompson Sampling strategy.

5.0.1 Problem formulation

Here we present the specific formulation of the problem discussed in this section, along with relevant notation.

A Bernoulli multi-armed bandit consists of a finite number of arms, K, indexed by $k \in \mathcal{K} = \{1, \ldots, K\}$. Each arm k, when pulled, receives reward $x_k(t)$ drawn from a Bernoulli distribution with mean μ_k. During an exploratory phase, at each time step t an agent must pull an arm j_t. The agent then receives the associated reward $x_{j_t}(t)$. The decision j_t is made based on past rewards. The sequence of decisions j_t is known as the allocation strategy. The exploratory phase ends at time T, whence the agent must make a final decision $\Psi(T)$, where the agent decides which arm has the highest mean. The allocation strategy is thus chosen to best improve the decision $\Psi(T)$. There are some different settings for the problem. In some cases the time horizon of the exploration phase is known. This information can aid a strategy in using their exploration budget more effectively. In other cases the strategy does not know the time horizon a head of time, but will continue in the exploratory phase until they are told to stop. For the algorithms we propose we do not assume that the length of the exploration phase, T, is known in advance.

Let $\sigma(i)$ denote the index of the arm with ith largest mean, such that $\mu_{\sigma(1)} \geq \mu_{\sigma(2)} \geq \cdots \geq \mu_{\sigma(K)}$. Let the gap $\Delta_k = \mu_{\sigma(1)} - \mu_k$ denote the distance between the largest expected reward and that of arm k. Let $\hat{X}_{i,T}$ be the empirical mean of arm i at time T.

A performance measure of interest to this problem is the simple regret. The simple regret at time T is the one step regret of choosing arm $\Psi(T)$. Formally

$$\mathbf{r}_T = \mu_{\sigma(1)} - \mu_{\Psi(T)} = \Delta_{\Psi(T)} \tag{5.1}$$

Another related measure which we will use in this section is the probability of error, P_e,

$$P_e = \mathbb{E}\left[\mu_{\sigma(1)} \neq \mu_{\Psi(T)}\right]. \tag{5.2}$$

Note that if a strategy attains low probability of error it also attains low simple regret.

Audibert et al. [5] introduced two measures of hardness, $\mathcal{H}_1$ and $\mathcal{H}_2$, that they showed characterised the difficulty of a given problem. The definitions are as follows,

$$\mathcal{H}_1 = \sum_{k=1}^{K} \frac{1}{\Delta_k^2}, \tag{5.3}$$

and,

$$\mathcal{H}_2 = \max_{k \in \{1,\ldots,K\}} \frac{k}{\Delta_i^2}. \tag{5.4}$$

.

5.1 Motivation

In order to better motivate the work presented in this chapter we will briefly discuss some potential applications for the best arm identification problem.

5.1.1 Automatic algorithm configuration in optimisation

Many tasks can be framed as an optimisation problem, from large logistical planning (i.e. travelling salesman problem) to drug discovery. In fact many machine learning problems can be reduced to an optimisation problem. For instance by using the principle of maximum margin, as a guide to achieving good generalisation, a support vector machine reduces the classification problem to a quadratic programming problem. For particularly hard optimisation problems the computational complexity of the task leads to in practice finding approximate solutions. Popular optimisation algorithms for hard problems include evolutionary algorithms and branch-and-bound strategies. These algorithms have parameters

which to perform well require tuning. Some algorithms internally have a stochastic component, such as evolutionary algorithms. However, even if the algorithm is deterministic the instance of the problem and data which it receives can change and can be thought to be drawn stochastically from some underlying distribution. In these settings we wish to tune the parameters for an algorithm (or perhaps choose between a group of different algorithms) in order to pick the best method for the particular task, but the performance under each setting is unknown and stochastic. We can think of the different algorithms (or the different parameter settings) as arms in a best arm identification problem and automatically configure an optimisation procedure to choice the best settings. Such problems have been considered by Birattari et al. [10].

5.1.2 Wireless channel allocation and frequency selection

Wireless networks are often made up of many non-centrally controlled nodes which must communicate amongst each other over a shared frequency spectrum. For instance the mobile phone network is made up of a series of cells that constitute a mobile operators network. Mobile phones then connect to these cells in order to make a call, or to send other data through the network. Communication is enabled using a range of frequency bands of the electromagnetic spectrum. If more than one device uses the same band then there will be interference and the quality of the signal will degrade possibly significantly. Each mobile phone wants to interfere as little as possible with other surrounding phones to ensure a better quality of service for the user. Since the network is ad-hoc in the sense that mobile phones change which cell they communicate with or connect and disconnect from the network regularly, a decentralised method of choosing which frequency range to communicate over is advantageous. This is called *dynamic frequency selection*. The choice of which cell a phone should connect also depends on stochastic factors such as signal strength and load. Adapting the cell with which a mobile phone communicates with is called *dynamic channel allocation*. In both problems the phone wishes to choose between a number of choices (be it frequency ranges, or cells) in which the observable state of each option is noisy. The phone can not wait a long time before making the decision as this would lead to poor service, but has a short fixed amount of time in which to find the best option. This too can be viewed as a best arm identification problem as suggested by Audibert et al. [5].

5.2 Ordered-Statistic Thompson Sampling

In this section we extend Thompson Sampling algorithms to the best-arm identification problem. The Thompson Sampling method requires only that one can sample from a posterior that is defined via the distribution of the rewards. A strength of the algorithm is that for many reward distributions this is cheap to compute. We hope to extend these benefits to the best-arm problem. However Bubeck, Munos, and Stoltz [13] observed that the best arm identification problem is fundamentally different to the multi-armed bandit problem. This means any algorithm that achieves the lower bound for the multi-armed bandit problem can not be optimal for the best arm identification problem.

We will explore models inspired by Thompson Sampling for the best arm identification problem. We propose one such model, Ordered-Statistic Thompson Sampling (OSTS). This algorithm class is described in Section 5.2.1 and can be viewed as a simple generalisation of Thompson Sampling. We provide a bound on the simple regret for this class of algorithm. The algorithm class presented requires a ranking distribution over the ordinals 1 to K, however we propose one such distribution whose long term behaviour should mimic Successive Rejects and so can be seen as parameter-free and without the requiring knowledge of the time horizon T. We investigate other ranking distributions that in expectation allocate more resources to higher ranked arms, and so can be seen as a more aggressive strategy.

5.2.1 Ordered-Statistic Thompson Sampling

Algorithm Description

In Thompson Sampling an arm is pulled with the probability that it is the best arm. This is done in a 1 sample Monte-Carlo fashion, where a sample is drawn from the posterior of each arm, and the maximum sample is used. In the best arm identification problem this leads to over exploiting, the arm with best mean is pulled too often and we do not pull the other arms sufficiently often in the exploratory phase in order to sufficiently reduce the probability of making the wrong guess in the final decision. This suggests that we do not want to pull just the arm we perceive to be best, but other arms as well. In order to achieve this Ordered-Statistic Thompson Sampling augments the basic Thompson Sampling algorithm with a distribution over the rank of the arms and via sampling uses

this distribution to determine what rank sample should be used.

Let the distribution over the rank of arms be denoted as $\nu(\mathcal{K})$. For a finite set of arms this would be a discrete distribution over the support $\{1, \ldots, K\}$. In the case of a Bernoulli armed bandits the uncertainty of the true means is modelled by Beta distributions. At each time step t in the exploratory phase Ordered-Statistic Thompson Sampling (OSTS) first samples from the distribution over the rank of arms, $o_t \sim \nu(\mathcal{K})$, this selects the ordered statistic, or rank to use. A sample, $\theta_{k,t}$, from the Beta distribution of each arm is drawn. The arms are ordered with respect to their samples. Let $\sigma_t(i)$ denote a function mapping a rank to an arm such that the ith *largest* sample corresponds to arm k when $\sigma_t(i) = k$. The ordering of the samples is then such that $\theta_{\sigma_t(1),t} \geq \theta_{\sigma_t(2),t} \geq \cdots \geq \theta_{\sigma_t(K)}$. We call i the rank of the sample. The algorithm selects arm $\sigma_t(o_t)$ to be pulled. The algorithm is summarised in Algorithm 5.1.

We can see that Ordered-Statistic Thompson Sampling is exactly Thompson Sampling when $\nu(k = 1) = 1$ and $\nu(k = i) = 0$ for $i = 2, \ldots, K$ and so this generalises Thompson Sampling. The algorithm is simple to implement and does not require knowledge of the length of the exploratory phase.

The performance of the algorithm will depend on the choice of $\nu(\mathcal{K})$. For instance $\nu(\mathcal{K})$ could be uniform across the integers from 1 to K. In expectation such a choice would behave much like a uniform allocation strategy that pulled each arm equally often. This is likely to waste too much of the allocation resource on arms that are extremely unlikely to be the best. Like Thompson Sampling, OSTS is a probability matching algorithm, given a rank it pulls an arm with the probability that it is that rank. The framing of the algorithm makes it natural then to think of an agent showing confidence by aggressively following in their belief when they try to pull the best arm, and acting more cautiously when they try to pull arms they perceive to be inferior. Choices of $\nu(\mathcal{K})$ that more aggressively allocate resources to high performing arms are likely to perform better. For instance a discrete distribution that decreases linearly between 1 and K would be more "aggressive" than uniform. Conversely $\nu(\mathcal{K})$, and therefore the agent, can not be too aggressive since we know that the standard Thompson Sampling algorithm for the multi-armed bandit is sub-optimal in that it does not reduce the simple regret exponentially.

In this thesis we consider three choices of $\nu(\mathcal{K})$. The first distribution is a Zipf distribution. For a certain parametrisation of the Zipf distribution we show

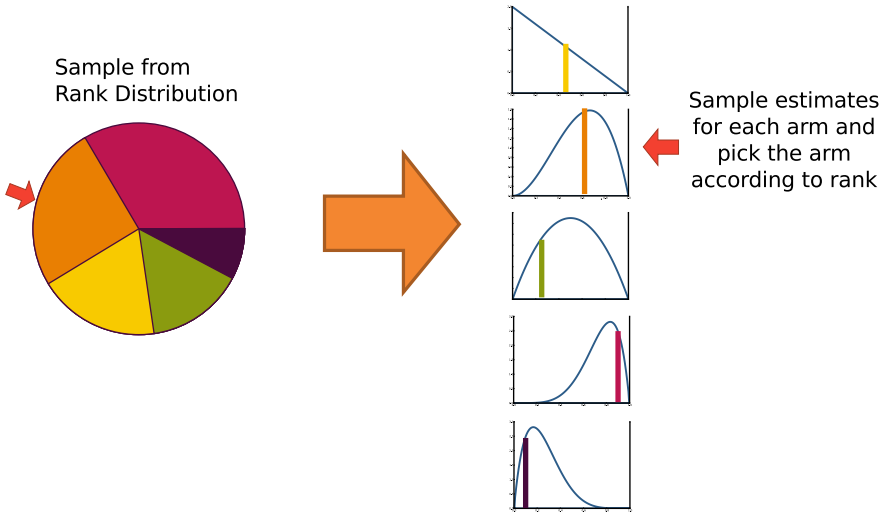


Figure 5.1: A picture summarising the Ordered-Statistic Thompson Sampling procedure. There is a rank distribution (bar chart on left), and a posterior distribution for each arm (shown on right). To choose an arm, the rank distribution is sampled. This specifies a rank, in this picture rank 2 has been sampled. The arm posterior distributions are sampled to produce mean estimates. The estimates are then ranked in order and the arm which is of the rank we sampled is pulled (marked with an red arrow)

for large T that the performance of OSTS should be asymptotically similar to that of Successive Rejects, a parameter-free algorithm proposed by Audibert, Bubeck, and Munos [5] that requires knowledge of the time horizon. The second distribution is a Poisson distribution, this was chosen to be more aggressive than the Zipf-like distribution. The third and final choice forms a distribution based on estimates of the gaps between arms in an attempt to adapt the level of aggression dependent on the problem.

Measure of Aggression

We feel that a useful way to compare different versions of OSTS is in terms of *aggressiveness*. We think that many other best arm algorithms can also be attributed a level of aggression. A more aggressive algorithm pursues the arms estimated as the better arms over the lesser ones; a less aggressive algorithm pulls the arms more uniformly. To formalise this notion we introduce a measure of aggression. This is joint work with my supervisor Jonathan Shapiro and is

Algorithm 5.1 Ordered-Statistic Thompson Sampling + Empirically Best Arm

Let $\alpha_{1,k} = 1$,
$\beta_{1,k} = 1$ for $k \in \{1, \ldots, K\}$.

Exploratory Phase:
for $t = 1, \ldots, T-1$ **do**
Sample $\theta_{t,i} \sim \text{Beta}(\alpha_{t,i}, \beta_{t,i})$, for $i \in \{1, \ldots, K\}$.
Sample $o_t \sim \nu(\mathcal{K})$
Pull arm $a_t = o_t\text{th largest}\, \theta_{t,i}$
Let $\alpha_{t+1,a_t} = \alpha_{t,a_t} + \mathbb{1}(x_{a_t}(t) = 1)$
$\beta_{t+1,a_t} = \beta_{t,a_t} + \mathbb{1}(x_{a_t}(t) = 0)$
Let $\alpha_{t+1,j} = \alpha_{t,j}$
$\beta_{t+1,j} = \beta_{t,j}$ for $j \in \{1, \ldots, K\} \setminus \{a_t\}$.
end for

Final Decision:
Let $\Psi(T)$ be $\text{argmax}_i\, \hat{X}_{i,T}$

being prepared for publication. The measure we propose is given by,

$$A(\nu(\mathcal{K})) = \frac{K-1}{\mathbb{E}[k] - 1} - 1, \tag{5.5}$$

where $\mathbb{E}[k]$ is the expected rank drawn from $\nu(\mathcal{K})$. We will treat $\frac{1}{0}$ as infinite. Here we can see that Thompson Sampling, designed for the multi-armed bandit problem, has unbounded aggression. The more focus that is put on pulling lower suboptimal arms, the lower the aggression will be. The minimum value this measure can take is 0 which occurs when we try to pull only the worst arm. We will restrict our measure to be only on distributions which are non-decreasing with rank. Much as the measures of hardness introduced by Audibert et al. [5] tell us something about the difficulty of a given *problem*, the aim of the measure of aggression introduced is to tell us about the behaviour of a given *strategy*. We later give some empirical evidence to suggest that there is a relation between this measure and performance, in that two variants of OSTS with the same aggression achieve similar levels of performance.

OSTS-Zipf

The first distribution $\nu(\mathcal{K})$ we consider is that of the truncated Zipf distribution, a distribution that follows a power law. The Zipf distribution is,

$$P_{\text{Zipf}}(j; N, s) = \frac{1}{j^s \sum_{i=1}^{N} \frac{1}{i^s}}, \tag{5.6}$$

where $1 \leq j \leq K$. We can then choose $\nu(k = i) = P_{\text{Zipf}}(i; K, s)$ for the parameter s. This will then favour higher ranked arms, as s increases the algorithm behaves more aggressively. For instance choosing the parameters $N = K$ and $s = 1$, this becomes $P_{\text{Zipf}}(j; K, 1) = \frac{1}{jH_K}$, where H_K is the Kth harmonic number. This makes the measure of aggression for OSTS-Zipf(1.0) as follows,

$$A(\text{Zipf}(j; K, 1)) = \frac{K\left(\left(\sum_{n=1}^{K} 1/n\right) - 1\right)}{K - \sum_{n=1}^{K} 1/n}. \tag{5.7}$$

This aggression grows logarithmically with K.

If we make a slight modification to this distribution then we can produce an interesting connection with an existing algorithm in the literature. The modification made is so that the probability of selecting the perceived best arm is the same as selecting the perceived second best arm. The particular way we choose to do this is by defining the modified Zipf distribution as follows

$$P_{\text{ModZipf}}(j; K) = \frac{1}{(\lfloor 1/j \rfloor + j)(H_K - 1/2)}. \tag{5.8}$$

We thus investigate when $\nu(k = i) = P_{\text{ModZipf}}(i; K)$. We will refer to the modified form just discussed as OSTS-Zipf, and when we refer to the unmodified form we will distinguish it by providing the value of parameter s used as OSTS-Zipf(s). The modified version of the Zipf distribution is interesting to look at because of how it compares to the algorithm Successive Rejects introduced by Audibert et al. [5].

Successive Rejects is split in to phases. In the first phase all arms are considered and pulled an equal number of times depending on the length of the phase. At the end of a phase the arm with the smallest empirical mean is dropped. The next phase continues to pull all remaining arms equally for the duration of that

phase, before the next arm is dropped. The strategy continues with the final phase having only two remaining arms. To choose the length of the phases the time horizon T needs to be known. The order at which the arms are dropped gives an ordering of the arms. We can think of this ordering as the perceived ranking of the arms for Successive Rejects. The arm dropped in the first phase is considered to have a rank of K, the arm dropped in the next phase is consider to have a rank of $K-1$ and so on. Successive Rejects pulls what it thinks is the best and second best arm n_2 times. n_k, for $k = 2, \ldots, K$, is defined to be,

$$n_k = \lceil \frac{T-K}{(H_K - 1/2)\, k} \rceil, \tag{5.9}$$

where here H_K is the Kth harmonic number ($H_K = \sum_{n=1}^{K} 1/n$), and $n_1 = n_2$. For $k > 1$ Successive Rejects pulls the arm it considers the kth best arm n_k times. This means the kth ranked ($k \in \{2, \ldots, K\}$) arm is played $\lceil \frac{T-K}{(H_K - \frac{1}{2})k} \rceil$ times and the 1st ranked arm played $\lceil \frac{T-K}{(H_K - \frac{1}{2})(2)} \rceil$. The total number of pulls of all the arms is $n_2 + \sum_{i=2}^{K} n_i \leq T$.

We wish to compare Successive Rejects to OSTS-Zipf. In expectation the arms associated with the kth ranked sample in OSTS-Zipf are pulled

$$\frac{T}{(\lfloor 1/k \rfloor + k)(H_K - 1/2)} \tag{5.10}$$

times. If we treat the phase in which an arm is dropped as the rank of an arm in Successive Rejects, then we can consider the ratio between the number of times we pull the kth ranked arm in Successive Rejects and the expected number of times pull an arm associated with rank k in OSTS-Zipf. This is given by the ratio of equations 5.9 and 5.10, $\frac{T-K}{T}$. When $T >> K$ the ratio tends to 1 and so the expected aggression of each algorithm is approximately the same. We can think of OSTS-Zipf as a randomised version of Successive Rejects. For small time horizons this is likely to bring a slight drop in performance, but does have the benefit of no longer requiring the time horizon to be known compared to Successive Rejects.

OSTS-Poisson

Audibert et al. observe that Successive Rejects may continue to pull an arm for some considerable time after it becomes clear that it will be rejected at the end

of the current phase. This suggests that for some problems it is not "aggressive" enough towards allocating most budget to the perceived better arms. Given this and the similarities to our proposed algorithm OSTS-Zipf, we wished to investigate a distribution that could be seen to be more aggressive by assigning larger probabilities to higher ranks of arms. Since the Poisson distribution decays faster than the Zipf distribution then OSTS-Poisson will likely spend less time pulling worse arms than OSTS-Zipf. The distribution is as follows,

$$\nu(k=i;K,\lambda)=\frac{\lambda^{i-1}}{(i-1)!\sum_{j=1}^{K}\frac{\lambda^{j-1}}{(j-1)!}}. \tag{5.11}$$

This leads to a measure of aggression for OSTS-Poisson(1.0) of,

$$A(\nu(k=i;K,1.0))=\frac{e\Gamma(K+1)\left((K+1)\Gamma(K,1)-\Gamma(K+1,1)\right)}{e\Gamma(K+1)\Gamma(K,1)} \tag{5.12}$$

$$+\frac{\Gamma(K)\left(e\Gamma(K+1,1)-1\right)}{e\Gamma(K+1)\Gamma(K,1)}. \tag{5.13}$$

The growth in aggression as a function of K is much larger for OSTS-Poisson(1.0) than for OSTS-Zipf(1.0) (and OSTS-Zipf). The aggression of OSTS-Poisson(1.0) is linear in K whereas the aggression of OSTS-Zipf is logarithmic in K. We can see how the aggression depends on K for some different versions of OSTS in Figure 5.2.

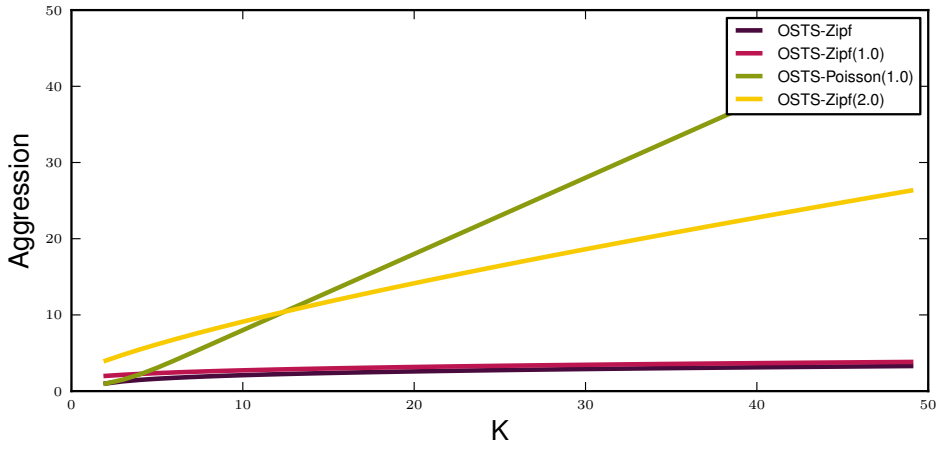


Figure 5.2: A plot of aggression for OSTS algorithms as a function of K, the number of arms

OSTS-Gap

A potential shortcoming of both OSTS-Poisson and OSTS-Zipf is that the distribution $\nu(i)$ (and therefore the number of times each arm is pulled) does not depend on the size of the gaps between arms Δ_k for $k = 2, \ldots, K$. Each arm is sampled based on its rank and so any arm of a given rank will be pulled in expectation the same number of times regardless of how far apart the arms are. However, the larger the gap between arms, the less time a given arm is associated with an incorrect ordered statistic.

In order to rectify this shortcoming, we devise an algorithm that does depend on the size of the gaps. Because we do not know the gaps, we use a Bayesian estimator to estimate them empirically.

Consider trying to minimise the probability of error, rather than minimising the simple regret. Let $E_{>\mu_{\sigma(1)}}$ be the event that we believe any sub-optimal arm has a higher mean than the true optimal, then we can upper bound this probability with a union bound as follows,

$$\begin{aligned} P(E_{>\mu_{\sigma(1)}}) &\leq P(x > \mu_{\sigma(1)}) + \sum_{i \in \{2,\ldots,K\}} P(x < \mu_{\sigma(i)}) \\ &\leq \sum_{i \in \{1,\ldots,K\}} e^{-2(\mu_i - x)^2 t_i}, \end{aligned} \tag{5.14}$$

for any $x \in [0, 1]$. The second inequality follows from Hoeffding's inequality, and t_i representing the number of times arm i is pulled.

Ideally, we would choose x and the t_i to make the bound as tight as possible. However, minimising the above bound is challenging. We want a closed form expression for $\nu(\mathcal{K})$ to make the procedure simple to compute, but we could not find an exact expression so we use an approximation. First, notice that unless $\mu_{\sigma(1)} = \mu_{\sigma(2)}$, the minimum value for x will lie between $\mu_{\sigma(1)}$ and $\mu_{\sigma(2)}$. We set it to be halfway between the two, $x = (\mu_{\sigma(1)} + \mu_{\sigma(2)})/2$, which is the correct value when $K = 2$. Assuming this value of x, it is optimal to pull the best and second best arm the same amount of times, therefore $t_{\sigma(1)} = t_{\sigma(2)}$. To set the values of the other t_is, we postulate that each arm should contribute to the above bound (Equation (5.14)) equally. That is,

$$e^{-2(\mu_{\sigma(1)} - x)^2 t_{\sigma(1)}} = e^{-2(\mu_{\sigma(2)} - x)^2 t_{\sigma(2)}} = \cdots = e^{-2(\mu_{\sigma(K)} - x)^2 t_{\sigma(K)}}. \tag{5.15}$$

Given this assumption and our assumed value of x, we can write t_i the number of times we wish to pull arm i in terms of $t_{\sigma(1)}$.

Letting $\Delta_{n,m} = \mu_n - \mu_m$, the expression for $t_{\sigma(i)}$ is found as follows,

$$\begin{aligned} e^{-2(\mu_{\sigma(i)}-x)^2 t_{\sigma(i)}} &= e^{-2(\mu_{\sigma(1)}-x)^2 t_{\sigma(1)}}, \\ (\mu_{\sigma(i)} - x)^2 t_{\sigma(i)} &= (\mu_{\sigma(1)} - x)^2 t_{\sigma(1)}, \\ (\Delta^2_{\sigma(1),\sigma(i)} + \Delta^2_{\sigma(2),\sigma(i)} + 2\Delta_{\sigma(1),\sigma(i)}\Delta_{\sigma(2),\sigma(i)}) t_{\sigma(i)} &= \Delta^2_{\sigma(1),\sigma(2)} t_{\sigma(1)}, \end{aligned}$$

finally rearranging to get,

$$t_{\sigma(i)} = \frac{\Delta^2_{\sigma(1),\sigma(2)} t_{\sigma(1)}}{\Delta^2_{\sigma(1),\sigma(i)} + \Delta^2_{\sigma(2),\sigma(i)} + 2\Delta_{\sigma(1),\sigma(i)}\Delta_{\sigma(2),\sigma(i)}},$$

for $i \in \{3, \ldots, K\}$. We do not know the values of $\mu_{\sigma(i)}$ and so we do not know the values of the Δ terms. We estimate these by using the empirical expectation values $\mathbb{E}\left[\Delta^2_{\sigma(1),\sigma(2)}\right]$, $\mathbb{E}\left[\Delta^2_{\sigma(1),\sigma(i)}\right]$, $\mathbb{E}\left[\Delta^2_{\sigma(2),\sigma(i)}\right]$ and $\mathbb{E}\left[\Delta_{\sigma(1),\sigma(i)}\Delta_{\sigma(2),\sigma(i)}\right]$ using the posteriors of arms $\sigma(1)$, $\sigma(2)$, and $\sigma(i)$. The two arms with the highest empirical means being considered arm $\sigma(1)$ and $\sigma(2)$. After calculating t_i in terms of $t_{\sigma(1)}$ for all posteriors (excluding the two with the highest empirical mean) the distribution $\nu(\mathcal{K})$ can be formed as follows,

$$\nu(k = i) = \frac{t_{\sigma(i)}}{\sum_{j \in \{1,\ldots,k\}} t_{\sigma(j)}}. \tag{5.16}$$

This proposed distribution is likely to be very aggressive compared to Successive Rejects when the gap between the top two arms is much higher than the gap between the best arm and all other arms. However, this distribution will also be less aggressive than Successive Rejects when the gaps are very close to one another, tending to a much more uniform allocation of arms, this is due to the assumption that the best and second best arm should be pulled the same amount of times. If all sub-optimal arms have the same mean then $\nu(\mathcal{K})$ will tend to the uniform distribution. This appears to be reasonable behaviour to tend to. To see this we can consider a problem where all sub-optimal arms do indeed have the same mean. Then $\mathcal{H}_2 = K\Delta^{-2}$, where Δ is the gap of all the arms. If we assume a strategy that did just pick arms uniformly such that each arm was pulled the

same number of times the probability could be bounded as follows,

$$P_e \leq \sum_{k=2}^{K} P\left(\hat{X}_{\sigma(1),T/K} \leq \hat{X}_{\sigma(k),T/K}\right) \tag{5.17}$$

$$\leq (K-1)e^{\frac{-T\Delta^2}{K}} \tag{5.18}$$

$$\leq (K-1)e^{\frac{-T}{\mathcal{H}_2}}. \tag{5.19}$$

The above follows by applying a union bound, and then Hoeffding's inequality. This is in line with the lower bound produced by Audibert et al. [5] and so in some sense is optimal for this case. It does not however necessarily follow that OSTS-Gap itself will be optimal for this case as this would depend on factors such as how quickly the rank distribution tended to a uniform distribution. This is just to show that the approximation we make to derive the rank distribution is justified in this case.

Since the algorithm adapts the ranking distribution $\nu(\mathcal{K})$ the aggression of the algorithm is not just a function of the number of arms, but the gaps between them. The aggression will change during the running of the algorithm as the gaps become more certain.

Analysis of OSTS

We show that if $\nu(\mathcal{K})$ is a distribution with non-zero support everywhere, then the expected simple regret diminishes exponentially.

The simple regret can be bounded as follows,

$$\begin{aligned}
\mathbb{E}\left[\mu_{\sigma(1)} - x_T\right] &= \mathbb{E}\left[\sum_{i\in\{2,\dots,K\}} \Delta_{\sigma(i)} \mathbb{P}(i = \text{argmax}_{j\in\{1,\dots,K\}}\, \hat{\mu}_{\sigma(j),T})\right] \\
&\leq \mathbb{E}\left[\sum_{i\in\{2,\dots,K\}} \Delta_{\sigma(i)} \mathbb{P}(\hat{\mu}_{\sigma(i),T} > \hat{\mu}_{\sigma(1),T})\right] \\
&\leq \mathbb{E}\left[\sum_{i\in\{2,\dots,K\}} \Delta_{\sigma(i)} \left(\mathbb{P}(\hat{\mu}_{\sigma(i),T} > x_{\sigma(i)}) + \mathbb{P}(\hat{\mu}_{\sigma(1),T} < x_{\sigma(i)})\right)\right] \\
&\leq \mathbb{E}\left[\sum_{i\in\{2,\dots,K\}} \Delta_{\sigma(i)} \left(e^{-2(\mu_{\sigma(i)} - x_{\sigma(i)})^2 T_{\sigma(i)}} + e^{-2(\mu_{\sigma(1)} - x_{\sigma(i)})^2 T_{\sigma(1)}}\right)\right] \\
&\leq \sum_{i\in\{2,\dots,K\}} \Delta_{\sigma(i)} \left(\mathbb{E}\left[e^{-2(\mu_{\sigma(i)} - x_{\sigma(i)})^2 T_{\sigma(i)}}\right] + \mathbb{E}\left[e^{-2(\mu_{\sigma(1)} - x_{\sigma(i)})^2 T_{\sigma(1)}}\right]\right),
\end{aligned}$$

where T_i is the number of times arm i has been pulled and x_i is some chosen constant where $\mu_i < x_i < \mu_{\sigma(1)}$. The second line is a union bound, line three overcounts and the fourth applies Hoeffding's inequality.

In order to show that this yields a regret which is exponentially decreasing with T, it suffices to show that each arm is pulled a number of times proportional to T, i.e. $T_i = \Omega(T)$ for all arms i. This follows from the fact that the sampling from the rank distribution $\nu(\mathcal{K})$ at time t is independent of the sampling at other times. Therefore, the probability of pulling any arm does not go to zero no matter the length of T.

First we give the general argument. We assume that the rank distribution is monotonic, so that $0 < \nu(K) \leq \nu(k-1) \leq \cdots \leq \nu(1) < 1$. We also assume that all the νs are independent of T. Then, if the sampled rank of an arm at time t is r, the probability of pulling that arm is $\nu(r)$. The probability of not pulling that arm is $1 - \nu(r)$. Since $\nu(r) \geq \nu(K)$ for any rank r, the probability of not pulling an arm at any round is less than or equal to $1 - \nu(K)$. This is true independent of the arm and the round, so the probability of not pulling an arm N times is less than $(1 - \nu(K))^N$. In order to not pull an arm $\Theta(T)$ times in T rounds, N would have to be $\Theta(T)$, and this upper bound to the probability would go exponentially to zero.

To make this more precise, we will bound the probability that $T_i = o(T)$.

Theorem 5.2.1. *$\exists T_0, \varepsilon_0 > 0$ s.t. $\forall T > T_0$ and for $0 < \varepsilon < \varepsilon_0$*

$$P(T_i/T < \varepsilon) < e^{-\alpha T/2}$$

with $\alpha = |\log[1 - \nu(K)]|$.

Proof. Let Y_i^t be 1 if arm i is pulled at time t and 0 otherwise. The number of times the ith arm is pulled, $T_i = \sum_{t=1}^{T} Y_i^t$. In order to make no assumptions about the dependencies between Ys at different times, let D_i^t denote everything on which Y_i^t depends. Then,

$$P(T_i = \tau) = \sum_{\sum Y_i^t = \tau} \prod_{t=1}^{T} P(Y_i^t | D_i^t), \tag{5.20}$$

where the sum is over all configurations which have $Y_i^t = 1$ exactly τ times. Since, for all t, $P(Y_i^t = 1|D_i^t) \leq \nu(1)$ and $P(Y_i^t = 0|D_i^t) \leq 1 - \nu(K)$, it follows that

$$P(T_i = \tau) \leq \sum_{\sum Y_i^t = \tau} \nu(1)^\tau (1 - \nu(K))^{T-\tau}, \tag{5.21}$$

$$= \binom{T}{\tau} \nu(1)^\tau (1 - \nu(K))^{T-\tau}. \tag{5.22}$$

The simplified bound,

$$(\frac{n}{e})^n \leq n! \leq ne(\frac{n}{e})^n \tag{5.23}$$

along with the assumption that $T_i/T \to 0$ in the limit $T \to \infty$, can be used to produce

$$P(T_i = \tau) \leq (eT)^{\tau+1} \tau^{-\tau} \exp\left(-\frac{\tau^2}{T}\right) \nu(1)^\tau (1 - \nu(K))^{T-\tau}. \tag{5.24}$$

If $\tau = o(\sqrt{T})$, then this decreases to zero faster than any exponential decay with exponent less than $|\ln(1 - \nu(K))|$, e.g. $o(\exp[\frac{1}{2}\ln(1 - \nu(K))])$. If τ grows more quickly than $\sqrt{T}$ (but still more slowly than T), then it could decay faster still. □

Thus, with a probability exponentially close to 1, each arm will be pulled a number of times which grows as fast as T, that is $T_i = \Omega(T)$ for all $i \in \{1, \ldots, K\}$.

This shows that if $\nu(\mathcal{K})$ is fixed with non-zero support everywhere the simple regret will diminish exponentially. We do not attempt to establish tight bounds,

just show the exponential reduction in simple regret.

A tight bound would be similar to the bound given for Successive Rejects. Audibert et al. give their bound for Successive Rejects in terms of the probability of error as,

$$P_e \leq \frac{K(K-1)}{2} \exp\left(-\frac{T-K}{(H_K - 1/2)\mathcal{H}_2}\right). \tag{5.25}$$

Since $H_K \approx \log K$ the time horizon only needs to be $o\left(\log K\mathcal{H}_2\right)$ before we can say with some confidence that the best arm will be identified. This is within a logarithmic factor of the hardness of the problem.

Optimistic Thompson Sampling

It was found empirically that the time horizon of the exploratory phase can be quite large before OSTS-Zipf behaves in a similar manner to Successive Rejects (see Section 5.2.2). The amount of times an arm other than the kth ranked is pulled when our sample from $\nu(\mathcal{K})$ dictates to pull the kth ranked is of order $O(\log(T))$, and so for small horizons the arm allocation will be closer to uniform than Successive Rejects. May et al. proposed Optimistic Thompson Sampling, with the insight that no advantage was gained by sampling from the left tail of the posterior distributions [53]. The proof of bounds for Thompson Sampling by Agrawal et al. [3] can be modified for the Optimistic Thompson Sampling algorithm. A reduction in additive constant terms (with respect to T) in the bound is shown in Section 3.2 and would be expected if the algorithm had an advantage. The number of times the incorrect arm is pulled when we attempt to pull the kth ranked arm is still $O(\log(T))$, however the leading constant may be significantly reduced. The hypothesis being that for shorter time horizons if we use Optimistic Thompson Sampling, we should marginally improve the performance.

The algorithm for Optimistic Thompson Sampling will not suffice for efficiently sampling the kth ranked arm, as it assumes we are looking for the maximum sample. In OSTS we are looking for the sample with the k rank. A proposed modification is to order the sampling distributions based on their current means. For all means ranked lower than the kth rank, draw a sample from the right hand side of the distribution. For all means ranked higher than the kth, draw a sample from the left hand side, and for the kth ranked mean draw the mean itself. We can think of this as the sampling procedure of OUTS (discussed in Section 3.2.3)

applied to the subset of arms with low means and applied to a pessimistic version of OUTS applied to the subset of arms with high means. The strategy is described by Algorithm 5.2.

Algorithm 5.2 Optimistic Ordered-Statistic Thompson Sampling + Empirically Best Arm

Let $\alpha_{1,k} = 1$,
$\beta_{1,k} = 1$ for $k \in \{1, \ldots, K\}$.

Exploratory Phase:
for $t = 1, \ldots, T-1$ **do**
Sample $o_t \sim \nu(\mathcal{K})$
Let $m_{t,i} = \frac{\alpha_{t,i}}{\alpha_{t,i}+\beta_{t,i}}$.
Let $M_t = o_t\text{th largest}\, m_{t,i}$.
Sample $\theta^{\text{samp}}_{t,i} \sim \text{Beta}(\alpha_{t,i}, \beta_{t,i})$.

$$\text{Let } \theta_{t,i} = \begin{cases} \max\left(\theta^{\text{samp}}_{t,i}, \frac{\alpha_{t,i}}{\alpha_{t,i}+\beta_{t,i}}\right) & m_{t,i} < M_t \\ m_{t,i} & m_{t,i} = M_t \\ \min\left(\theta^{\text{samp}}_{t,i}, \frac{\alpha_{t,i}}{\alpha_{t,i}+\beta_{t,i}}\right) & m_{t,i} > M_t. \end{cases} \text{, for } i \in \{1, \ldots, K\}.$$

Pull arm $a_t = o_t\text{th largest}\, \theta_{t,i}$
Let $\alpha_{t+1,a_t} = \alpha_{t,a_t} + \mathbb{1}(x_{a_t}(t) = 1)$
$\beta_{t+1,a_t} = \beta_{t,a_t} + \mathbb{1}(x_{a_t}(t) = 0)$
Let $\alpha_{t+1,j} = \alpha_{t,j}$
$\beta_{t+1,j} = \beta_{t,j}$ for $j \in \{1, \ldots, K\} \setminus \{a_t\}$.
end for

Final Decision:
Let $\Psi(T)$ be $\text{argmax}_i\, \hat{X}_{i,T}$

5.2.2 Experiments

The experiments are identical to those by Audibert, Bubeck, and Munos [5] and similarly we report performance in terms of probability of error. Table 5.1 lists the experiments performed. In the experiments we compare our algorithms to Thompson Sampling [34], Successive Rejects [5] and BayesGap [27], for completeness descriptions are provided in Section 2.8. Thompson Sampling provably can not asymptotically achieve exponentially diminishing probability of error in the best arm identification problem since it achieves the lower bound of regret in the multi-armed bandit problem. Successive Rejects has been proved to achieve exponentially diminishing probability of error as has BayesGap. In all experiments

Table 5.1: Parameters of best arm identification experiments. These are the same as those proposed by Audibert et al. [5]

Experiment	K	Arm Means			
1	20	$\mu_{\sigma(1)}$	$= 0.5,$	$\mu_{\sigma(2):\sigma(20)}$	$= 0.4$
2	20	$\mu_{\sigma(1)}$	$= 0.5,$	$\mu_{\sigma(2):\sigma(6)}$	$= 0.42,$
		$\mu_{\sigma(7):\sigma(20)}$	$= 0.38$		
3	4	$\mu_{\sigma(1)}$	$= 0.5,$		
		$\mu_{\sigma(i)}$	$= 0.5 - (0.37)^i,\ i \in \{2,3,4\}$		
4	6	$\mu_{\sigma(1)}$	$= 0.5,$	$\mu_{\sigma(2)}$	$= 0.42,$
		$\mu_{\sigma(3):\sigma(4)}$	$= 0.4,$	$\mu_{\sigma(5):\sigma(6)}$	$= 0.35$
5	15	$\mu_{\sigma(1)}$	$= 0.5,$		
		$\mu_{\sigma(i)}$	$= 0.5 - 0.025i,\ \ i \in \{2,\ldots,15\}$		
6	20	$\mu_{\sigma(1)}$	$= 0.5,$	$\mu_{\sigma(2)}$	$= 0.48,$
		$\mu_{\sigma(7):\sigma(20)}$	$= 0.37$		
7	30	$\mu_{\sigma(1)}$	$= 0.5,$	$\mu_{\sigma(2):\sigma(6)}$	$= 0.45,$
		$\mu_{\sigma(7):\sigma(20)}$	$= 0.43,$	$\mu_{\sigma(21):\sigma(30)}$	$= 0.38$

the parameter γ_{bgap} for Bayes Gap was set to $\gamma_{\text{bgap}} = 0.0001$. This was found to give good performance for the algorithm during testing.

Performance

The algorithms were run on all experiments described. For each experiment 50 permutations of the arms were created randomly. For each permutation the algorithm was run 1000 times to estimate the error rate, the error rates were then averaged over the 50 permutations. Figure 5.5 shows the results for OSTS (Poisson, Zipf and Gap) as compared with the other algorithms. Figure 5.3 shows the performance of OSTS-Zipf(s) and OSTS-Poisson as the aggression of the algorithm is increased. We can see for a given experiment that aggression predicts performance better than the particular type of distribution used, with distributions with the same aggression having roughly the same performance. Unfortunately however the problem determines the appropriate level of aggression. Experiments 6 and 7 showed larger aggression leading to lower error. Experiment

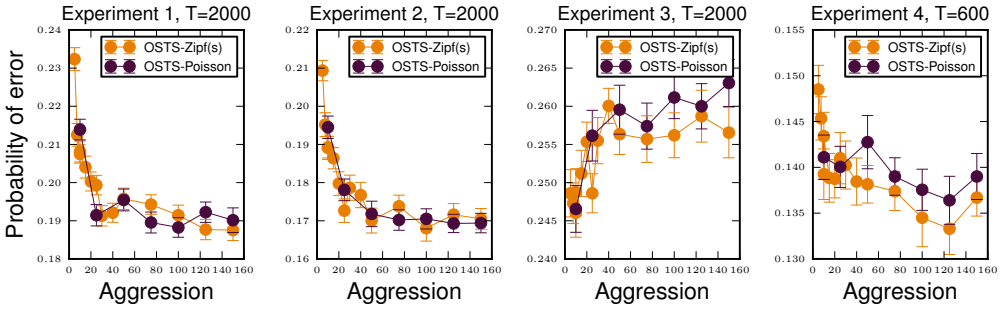
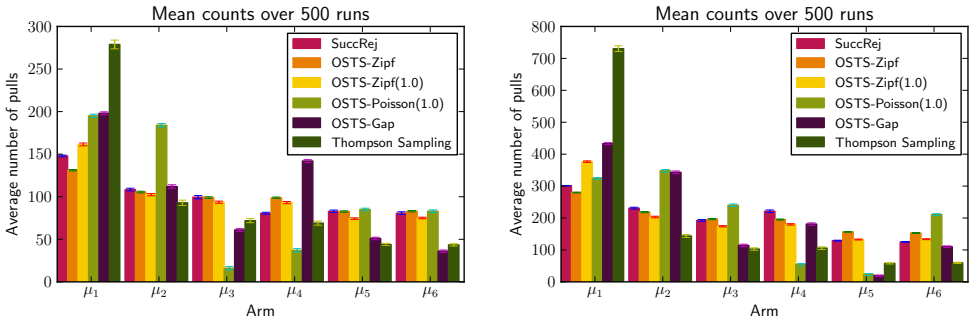


Figure 5.3: Comparing two distribution types (Zipf,Poisson) with varying levels of aggression. We can see that they follow similar levels of performance for the same aggression.

5, like experiment 3, showed lower error with lower aggression.

Comparing distribution of allocation

To understand how the algorithms were distributing their allocation of pulls the pull counts for each arm were recorded for a single instance of an experiment. Figure 5.4 shows the distribution of arm pulls for several of the algorithms on experiment 4. It can be seen how Thompson Sampling "exploits" the best arm far more than any other algorithms. The figure also shows a close resemblance between the number of times Successive Rejects pulls a particular arm, and how many times OSTS-Zipf pulls the same arm. As expected OSTS-Poisson is more exploitative than OSTS-Zipf, but still far less than Thompson Sampling. OSTS-Gap also appears to exploit more than OSTS-Zipf does. Empirically we found OSTS-Zipf behaved like Successive Rejects for large time horizons, however for a small time horizon OSTS-Zipf appear to perform less well, the allocation being less aggressive than Successive Rejects.



(a) Time Horizon: 600

(b) Time Horizon: 1200

(c) Time Horizon: 2400

(d) Time Horizon: 4800

Figure 5.4: Comparison of allocation distributions on experiment 4

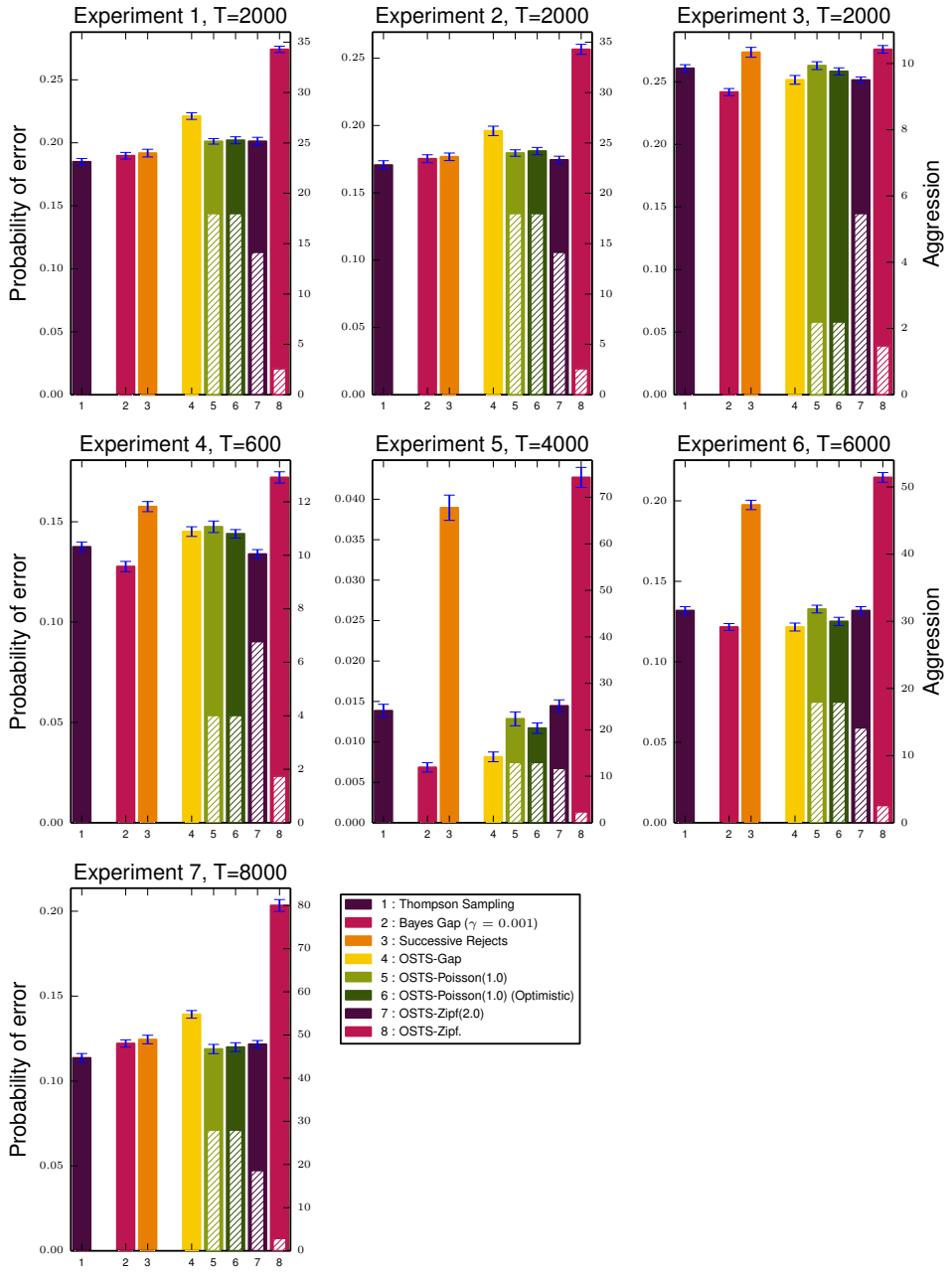


Figure 5.5: Coloured bars show percentage of errors, and white bars show aggression, and is only plotted for OSTS-Zipf and OSTS-Poisson algorithms. We can see that in general more aggressive strategies empirically perform better. However, Experiment 5 shows standard Thompson Sampling beginning to perform worse than less aggressive strategies. Empirically the optimistic version of OSTS-Poisson performs the best all round out of the Thompson Sampling based strategies tried, performing better than Successive Rejects, and close in performance to the parametrised algorithm Bayes Gap. Optimism appears only to give extremely marginal improvements to the general algorithm in this case.

Optimism in OSTS

A small experiment was performed to investigate if there was much advantage to using optimism in Ordered-Statistic Thompson Sampling. Experiment 4 was chosen and OSTS-Zipf(s) was compared with an optimistic version for varying levels of aggression. Experiment was repeated $50,000$ times for each aggression level. Figure 5.6 shows the mean probability of error and the standard error for this experiment. The difference appears to be marginal, only showing a slight improvement for some of the aggression levels tried.

Further empirical comparison of OSTS-Zipf and Successive Reject

We have discussed in Section 5.2.1 the similarities between OSTS-Zipf and Successive Reject. For large T the allocation of each algorithm should be approximately the same, which has been shown empirically in Section 5.2.2. Another experiment was performed to investigate how quickly it takes for their performance to match. We ran the experiments 100 times for time horizons of $1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 5500, 6000, 6500, 7000, 7500$. Figure 5.7 shows two plots of the probability of error for the algorithms as a function of the time horizon for experiment 1. Two versions of the Zipf strategies were plotted, the version where the top two ranked arms are pulled equally in expectation (referred to as OSTS-Zipf) and an unmodified version with parameter one (OSTS-Zipf(1.0)). Thompson Sampling is also plotted to a provide a comparison. It can be seen that as the time horizon increases the different between OSTS-Zipf and Successive Rejects diminishes. OSTS-Zipf(1.0), a more aggressive algorithm to OSTS-Zipf, performs slightly better than OSTS-Zipf and eventually outperforms Successive Rejects for a large enough time horizon. We see that for experiment 1 Thompson Sampling has a similar performance to Successive Rejects initially and empirically then starts to outperform it. However we know that as the time horizon gets very big it must start to degrade in performance in comparison to Successive Rejects. This plot also shows that for Optimistic OSTS-Zipf is almost indistinguishable from standard OSTS-Zipf. It appears that the effects of optimism are more noticeable for more aggressive strategies and become ever more marginal as the aggressiveness is reduced.

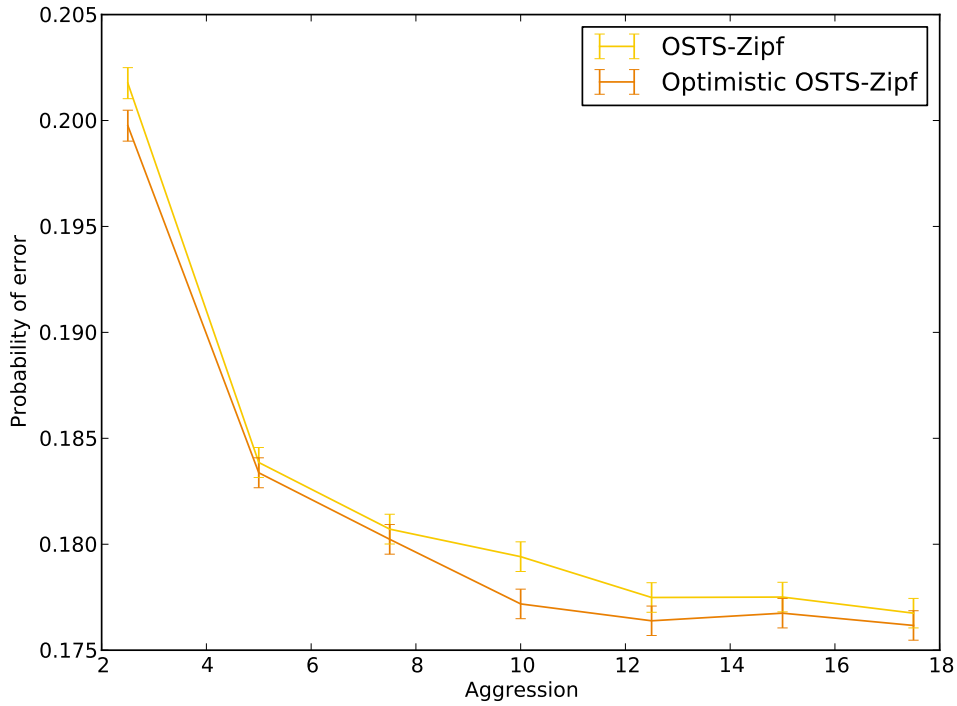


Figure 5.6: A comparison of OSTS-Zipf(s) with an optimistic version. We can see that the optimistic version appears to marginally outperform the non-optimistic version. However the performance difference is small and we only attain a statistically significant difference for some aggression levels.

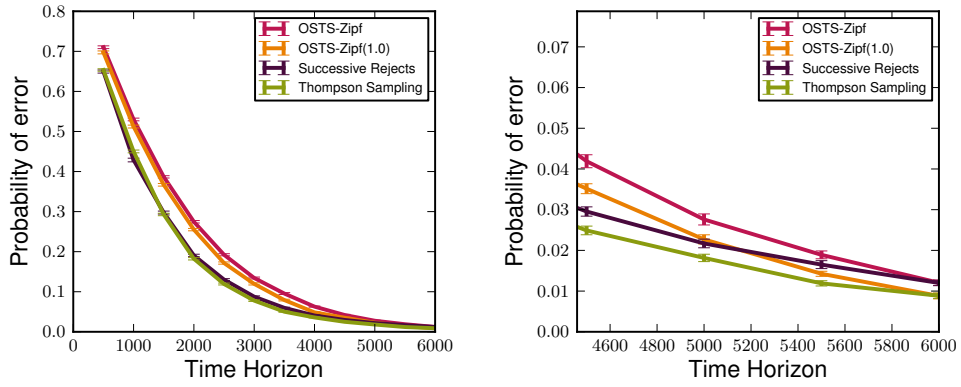


Figure 5.7: Two plots of performance on experiment 1 for varying time horizons. The OSTS-Zipf/OSTS-Zipf(1.0) algorithms initially perform worse than Successive Reject, but interestingly also Thompson Sampling. This implies that a bias to being very aggressive initially might be beneficial. The performance of OSTS-Zipf gets closer to Successive Reject as the time horizon gets larger.

5.2.3 Increasing aggression in early rounds

We observe that for small time horizons a very aggressive algorithm, such as Thompson Sampling, performs better than its less aggressive counterparts. However we know that Thompson Sampling in the long term is sub-optimal. An aggressive strategy like Thompson Sampling performing well initially indicates that the standard OSTS procedure could be improved by introducing a bias towards aggressiveness, especially in the early stages. As the agent becomes more sure of the arms means the aggression should then revert to what is specified by $\nu(\mathcal{K})$, becoming less aggressive. To test this theory we also investigated a modification to the OSTS algorithm we term Aggressive OSTS. In Aggressive OSTS the expected value of the arms mean is also stored for each arm (m_i for $i \in \{1, \ldots, K\}$). The expected values also have a rank. Let $\sigma_m(i)$ be the arm with rank i when ranked by the expected value. As before with OSTS, at each round, a rank o_t is drawn from $\nu(\mathcal{K})$ and a sample θ_i drawn for each arm. If $o_t > 1$ then we ascend through the arms, $\sigma_m(j)$, for $j = 2$ to o_t. Let $E_m(j)$ be the event that $(m_{\sigma_m(j-1)} + m_{\sigma_m(j)})/2 < \theta_{\sigma_m(j)}$ If $E_m(j)$ occurs then we take this as indication that we have not sufficiently distinguished between the arms $\sigma_m(j)$ and $\sigma_m(j-1)$. In this event we then just perform Thompson Sampling using just the arms $\sigma_m(k)$ for $k \geq j-1$. If event $E_m(j)$ does not occur for all $j \leq o_t$ then this is interpreted as higher arms being well estimated, the algorithm then reverts to performing OSTS. When the arms are poorly estimated event $E_m(j)$ will occur more frequently for $j < o_t$ and so the algorithm will be given a bias to behave more aggressively. As the arms being better estimated then $E_m(j)$ occurs less and less frequently and so the algorithm reverts to behaving like OSTS. The hypothesis is that this added bias will be beneficial to the performance of the algorithm. We investigated this by using Aggressive OSTS-Zipf on experiment 1 and 6 for varying time horizons and comparing it to OSTS-Zipf, Successive Rejects and Thompson Sampling. Figure 5.8 shows these experiments. The improvements by adding this bias are clear. Aggressive OSTS-Zipf closely maps the performance of Successive Rejects for short time horizons without the need to know the horizon ahead of time.

The same set of experiments as shown in Figure 5.5 were performed using the biased aggressive versions of OSTS. The results of these experiments can be shown in Figure 5.9.

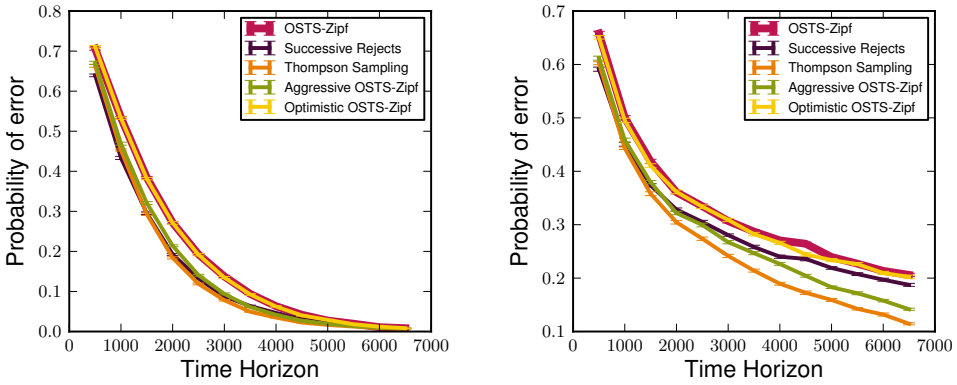


Figure 5.8: Plots of performance for Experiment 1 and 6. The results clearly show the improvement of Aggressive OSTS-Zipf as compared to OSTS-Zipf especially for small time horizons.

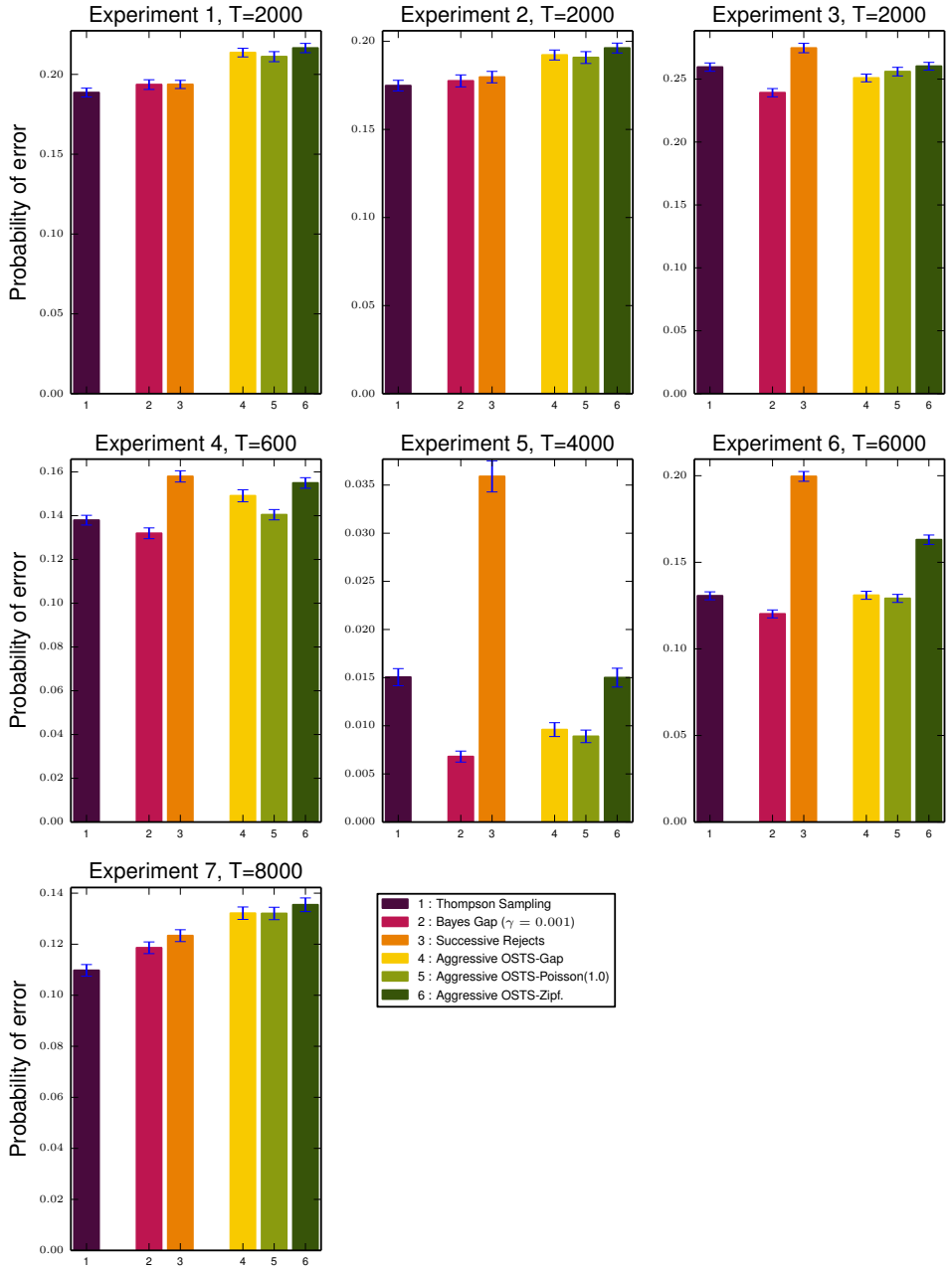


Figure 5.9: Comparison of benchmark algorithms to some Aggressive OSTS algorithms. We can see an improvement to OSTS when it is biased to be more aggressive over the basic OSTS algorithm (some results shown in Figure 5.5).

5.2.4 OSTS summary

With OSTS we have proposed a class of best arm identification algorithm. Some of the advantages of OSTS are,

- It is not necessary to know the time horizon prior to making decisions.
- It is computationally efficient to compute a decision. The time complexity scales linearly with the number of arms K.
- Like Thompson Sampling, OSTS can be used for problems with different reward distributions. Any reward distribution with a conjugate prior can be handled without loss of computational efficiency. More complicated posterior distributions can be approximated using MCMC for example. Like Thompson Sampling it also means that arms need not be independent.

We introduced a concept of aggression as a property summarising the behaviour of an algorithm. We find that empirically more aggressive algorithms appear to perform better. Comparisons of OSTS-Poisson and OSTS-Zipf suggest that the specifics of a distribution are far less important than its aggressiveness in determining performance. In environments for which we expect OSTS-Gap to be aggressive it consistently outperforms other parameter-free algorithms like Successive Reject. More aggressive variants such as OSTS-Poisson are comparable to the parametrized gap based method.

It is still left to be shown which fixed distribution $\nu(\mathcal{K})$ is the most suitable choice in balancing exploration. For this fixed $\nu(\mathcal{K})$ we could expect to get a much stronger theoretical bound on its performance, which we would expect to be, like Successive Reject, within a logarithmic factor of the optimal sample complexity. It is expected that one avenue to pursue this bound would require much stronger control on the tail distribution for the regret of Thompson Sampling, in order to have control on the minimum number of times each arm had been pulled. However we empirically show that a more aggressive adaptation of OSTS-Zipf has behaviour comparable or better to Successive Rejects. OSTS-Gap adapts the sampling distribution and can be seen as a parameterless version of the algorithm. The algorithm is not very aggressive when all suboptimal arms have the same mean, and so further work might try to improve the adaptive distribution so that it is more aggressive in this case.

We also show that optimism can be adapted to this setting, however in the experiments we have tried the gain, if any, was marginal. This is counter to the behaviour exhibited in Thompson Sampling where optimism (and also the form of optimism shown in OUTS) shows a statistically significant gain in performance.

An interesting avenue of investigation would be to see how the algorithm behaves in the M-best identification problem, where an agent must select not only the best arm but the top M arms. This problem has recently been investigated by Wang et al. [74]. Although not investigated in this thesis it is conjectured that little to no adaptation of the general algorithm would be necessary, only perhaps that the distribution $\nu(\mathcal{K})$ be carefully chosen.

5.3 Towards an anytime Successive Rejects

In the previous section we have seen OSTS-Zipf performs similarly to Successive Rejects for time horizons that are much larger than the hardness, $\mathcal{H}_2$, of the problem. We have also shown that other choices of $\nu(\mathcal{K})$ outperform Successive Rejects for a variety of test problems while being suitable for an "anytime" best arm identification problem. By an anytime best identification problem we mean that the time horizon is not known in advance. Instead a recommendation can be asked at any time. Audibert et al. [5] conclude the paper in which they introduce Successive Rejects by asking what an anytime version of the algorithm may look like. OSTS from the previous section shows that we can achieve an anytime algorithm that empirically outperforms Successive Rejects for a suitable $\nu(\mathcal{K})$. However, it is difficult to provide an analysis as to what this $\nu(\mathcal{K})$ might look like in general. The reason is that it is non-trivial to have tight control on the tail probabilities for the number of times an arm is pulled. Proposition 1 of Kaufmann et al. [34] makes steps towards such control, but unfortunately does not appear to be strong enough. As a step towards an anytime version of Successive Rejects we propose an alternative sampling algorithm, Anytime SR. Like OSTS we still use the idea of ranking of the arms. The ranking is now instead based on the number of times an arm has been pulled. This corresponds to how we viewed Successive Rejects as having ranked the arms. We also de-randomise the algorithm by removing the sampling from the rank distribution.

5.3.1 Algorithm description

Instead the procedure is as follows. We sample optimistically from the posterior distributions describing the means of the arms , finding the arm, b_t, that corresponds to the largest sampled estimate (like Optimistic Thompson Sampling). Next we rank the arms based on number of times pulled such that rank 1 corresponds to the arm pulled most, and K corresponds to the arm pulled least. For rank $k = K, \ldots, 2$ (starting from the least pulled arm) we see if kth ranked arm has been pulled less than $2/K$ times that of the arm b_t has been pulled. If we find an arm for which this is true, we pull it. Otherwise we pull the arm b_t. This procedure effectively performs Optimistic Thompson Sampling until the sub-optimal arms are pulled too infrequently compared to the candidate best arm. The rationale behind using the counts as a ranking mechanism is that the closer the mean of an arm is to the true best mean, the more likely we are to pull it in Optimistic Thompson Sampling. The details of this procedure are shown in Algorithm 5.3.

5.3.2 Experiments

Performance

We perform the experiments described in section 5.1, comparing Anytime SR to Successive Rejects. The results of these experiments are shown in Figure 5.10. We can see the extremely close resemblance of the two algorithms in terms of performance.

Distribution of pulls

Empirically we can demonstrate why we might expect the distribution of number of pulls to be much more predictable than OSTS. This will allow for easier control over the number of times each arm is pulled. To see this we look at a 4 arm best arm identification problem. The arm means are 0.5, 0.475, 0.45, and 0.425, with a time horizon of 1000. The experiment was run 3000 times and the distribution of counts over the runs was recorded. Figure 5.11 shows the results comparing OSTS-Zipf, Anytime SR and Successive Rejects. It shows the distribution for each arm separately. It can be seen that Anytime SR shows a very close resemblance to Successive Reject. The distribution for OSTS-Zipf is much less sharp for this time horizon showing why the analyse may be much harder.

Algorithm 5.3 Anytime Successive Rejects

Let $n_t(j)$ be the number of times arm j is pulled at time t.
Let $\alpha_{1,k} = 1$,
$\beta_{1,k} = 1$ for $k \in \{1, \ldots, K\}$.

Exploratory Phase:

for $t = 1, \ldots, T$ **do**
Sample $\theta_i \sim \text{Beta}(\alpha_{t,i}, \beta_{t,i})$, for $i \in \{1, \ldots, K\}$..
Find arm $b_t = \text{argmax}_i \max\left(\theta_i, \frac{\alpha_{t,i}}{\alpha_{t,i}+\beta_{t,i}}\right)$
Order arms by pulls s.t. c_1 is the arm pulled most, and c_K pulled least.
Find largest j s.t. $n_t(b_t) > \frac{jn_t(c_j)}{2}$
$a_t = \begin{cases} c_j & \text{if } j \text{ exists,} \\ b_t & \text{otherwise,} \end{cases}$
Pull arm a_t.
Let $\alpha_{t+1,a_t} = \alpha_{t,a_t} + \mathbb{1}(x_{a_t}(t) = 1)$
$\beta_{t+1,a_t} = \beta_{t,a_t} + \mathbb{1}(x_{a_t}(t) = 0)$
Let $\alpha_{t+1,j} = \alpha_{t,j}$
$\beta_{t+1,j} = \beta_{t,j}$ for $j \in \{1, \ldots, K\} \setminus \{a_t\}$.
end for

Final Decision:

Let $\Psi(T)$ be $\text{argmax}_i \hat{X}_{i,T}$

5.3.3 Analysis

We give a weak upper bound on the probability of error of Anytime SR.

Lemma 5.3.1. *The probability of error, P_e, of Anytime SR is upper-bounded as follows,*

$$P_e \leq (K-1)e^{-\frac{2t}{K^2\mathcal{H}_2}} \tag{5.26}$$

for $t > K(K-1)/2 + 1$.

Proof. By a union bound and application of a Hoeffding bound we have,

$$P_e \leq \sum_{k=2}^{K} e^{-\Delta_k^2 n_k}, \tag{5.27}$$

where n_k is the number of times arm k is pulled. This is derived in much the

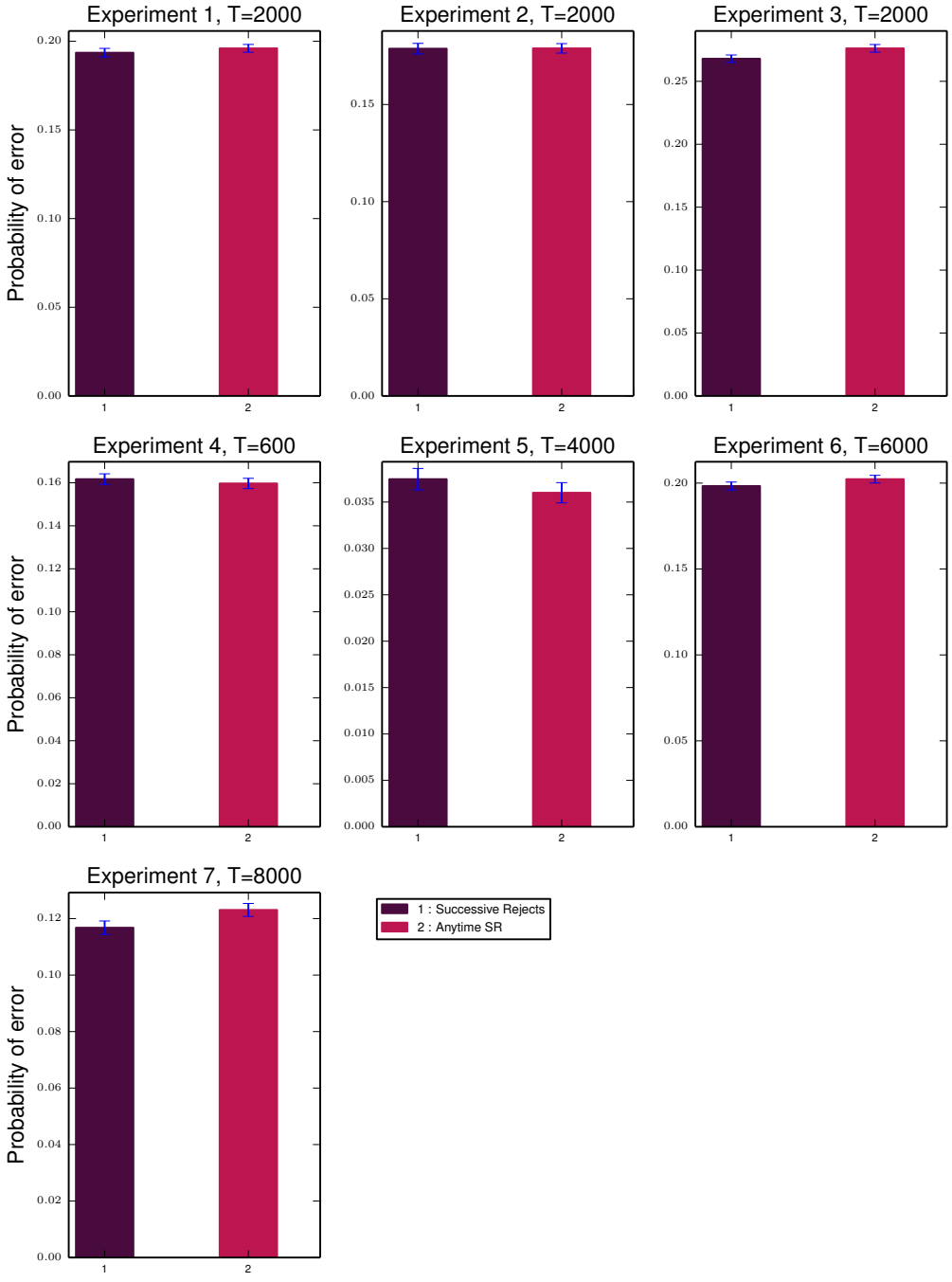


Figure 5.10: Comparing Anytime SR to Successive Rejects.

same way as with the bound for Successive Reject by Audibert et al. [5].

By definition of the algorithm the most pulled arm can only be $K/2$ times that of the lowest pulled arm. Let t be the time horizon and let $n_{\min}$ be the smallest number of times any arm can be pulled. For $t > K(K-1)/2$ the least pulled arm

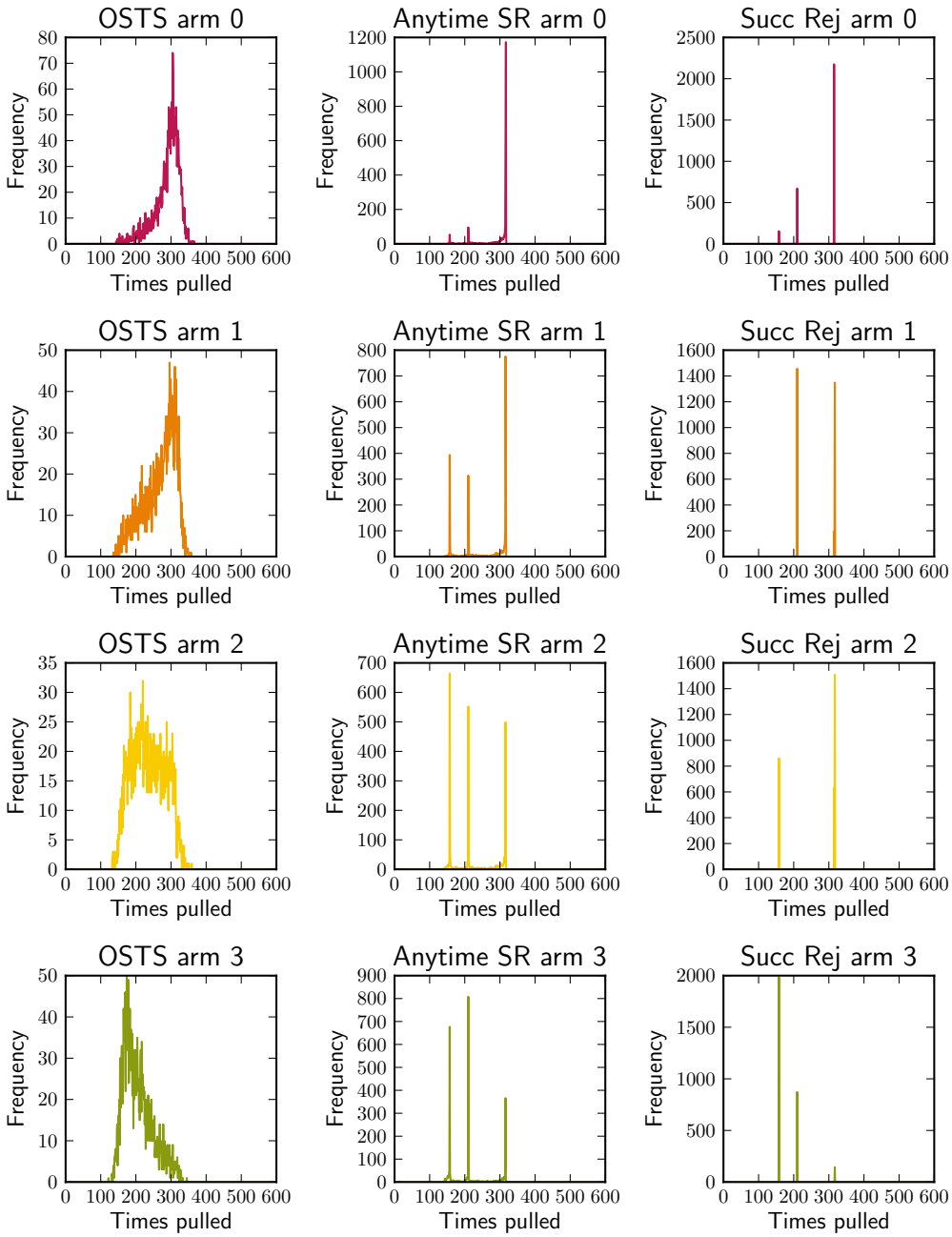


Figure 5.11: Comparing distribution of pulls for OSTS-Zipf, Anytime SR, and Successive Rejects.

is smallest when all other arms have been pulled an equal number of times. This means that $n_{\min} + \frac{K(K-1)n_{\min}}{2} = t$. Rearranging we get that $n_{\min} = \frac{2t}{K(K-1)+2}$. We

note that $n_{\min} \geq \frac{2t}{K^2}$. Putting $n_{\min}$ into Equation 5.27 we get,

$$P_e \leq (K-1)e^{-\frac{2\Delta_k^2 t}{K^2}}. \tag{5.28}$$

Then by the definition of $\mathcal{H}_2$ we have,

$$P_e \leq (K-1)e^{-\frac{2t}{\mathcal{H}_2 K^2}}. \tag{5.29}$$

□

This proof is a weak result, but serves to formalise that the algorithm has a probability error that reduces exponentially with time.

The next lemma tells us the probability of a sample of the kth arm being larger than a sample of the best arm given the number of times each is pulled.

Lemma 5.3.2. *Let n_k be the minimum of the number of times the kth best arm and the best arm are pulled. Let θ_k be a sample from the posterior of this arm. Then,*

$$P\left(\theta_k > \theta_1\right) \leq 3e^{-\frac{2\Delta_k^2 n_k}{9}}. \tag{5.30}$$

Proof. For $\mu_{\sigma(k)} < x_{\sigma(k)} < y_{\sigma(k)}$ it follows that,

$$P\left(\theta_{\sigma(k)} > \theta_{\sigma(1)}\right) \leq P\left(\hat{\mu}_{\sigma(k)} > x_{\sigma(k)}\right) + P\left(\theta_{\sigma(k)} > y_{\sigma(k)}, \hat{\mu}_{\sigma(k)} < x_{\sigma(k)}\right) + P\left(\hat{\mu}_{\sigma(1)} < y_{\sigma(k)}\right). \tag{5.31}$$

The first term denotes the probability that the estimated mean of a suboptimal arm is poorly estimated and can be bounded using a Chernoff-Hoeffding bound as follows,

$$\begin{aligned} P\left(\hat{\mu}_{\sigma(k)} > x_{\sigma(k)}\right) &\leq P\left(\hat{\mu}_{\sigma(k)} > \mu_{\sigma(k)} + (x_{\sigma(k)} - \mu_{\sigma(k)})\right), && (5.32)\\ &= P\left(n_{\sigma(k)}\hat{\mu}_{\sigma(k)} > n_{\sigma(k)}\mu_{\sigma(k)} + (x_{\sigma(k)} - \mu_{\sigma(k)})n_{\sigma(k)}\right), && (5.33)\\ &\leq e^{-2(x_{\sigma(k)} - \mu_{\sigma(k)})^2 n_{\sigma(k)}}. && (5.34) \end{aligned}$$

The second term is the probability that the sample estimate of the suboptimal arm is large when the estimated mean is well estimated. The proof is given by lemma 3 in the Thompson Sampling proof of Agrawal and Goyal [3] which gives

the following bound,

$$P\left(\theta_{\sigma(k)} > y_{\sigma(k)}, \hat{\mu}_{\sigma(k)} < x_{\sigma(k)}\right) \leq e^{-n_{\sigma(k)} d(x_{\sigma(k)}, y_{\sigma(k)})}, \tag{5.35}$$

where $d(x_k, y_k) = x_k \log \frac{x_k}{y_k} + (1 - x_k)\frac{1-x_k}{1-y_k}$. Using Pinsker's inequality we get that,

$$P\left(\theta_{\sigma(k)} > y_{\sigma(k)}, \hat{\mu}_{\sigma(k)} < x_{\sigma(k)}\right) \leq e^{-2(x_{\sigma(k)} - y_{\sigma(k)})^2 n_{\sigma(k)}}. \tag{5.36}$$

The third term is the probability that the optimal arm is poorly estimated, which again follows from a Chernoff-Hoeffding bound as follows,

$$P\left(\hat{\mu}_{\sigma(1)} < y_{\sigma(k)}\right) \leq e^{-2(y_{\sigma(k)} - \mu_{\sigma(1)})^2 n_{\sigma(k)}}. \tag{5.37}$$

By setting $x_{\sigma(k)} = \mu_{\sigma(k)} + (\mu_{\sigma(1)} - \mu_{\sigma(k)})/3$ and $y_{\sigma(k)} = \mu_{\sigma(k)} + 2(\mu_{\sigma(1)} - \mu_{\sigma(k)})/3$ we arrive at the bound given in the lemma. □

We can see from lemma 5.30 that once an arm whose gap is $\Delta_{\sigma(K)}$, or greater, has been pulled $O(\frac{K}{\Delta^2_{\sigma(K)}})$ times then we have a small probability of that arm having the largest sample. It will therefore not be pulled when Anytime SR behaves analogously to Optimistic Thompson Sampling. It instead will only be pulled a number of times based on its "rank" (which is based on the number of times it has been pulled).

5.3.4 Anytime SR summary

This section proposes an anytime algorithm called Anytime SR which attempts to have similar behaviour to Successive Rejects but without the requirement to know the time horizon. This is motivated by two things, firstly OSTS-Zipf is an anytime algorithm that eventually behaves similarly (for large time horizon that may be much larger than the hardness of the problem). Secondly Audibert et al. raise the question of an anytime version of Successive Rejects in the paper in which it is introduced [5]. Empirically it has been demonstrated to have extremely similar behaviour and so is a strong candidate algorithm to answering Audibert's question. It is hoped that it can be shown that this algorithm has an upper-bound on its probability of error which is the same within a constant factor to that of Successive Rejects.

5.4 Maximum Boundary of Pairs

In this section we propose a parameterless Bayesian algorithm for the best arm identification problem. We call this method Maximum Boundary of Pairs (MBoP). At each round the method uses Thompson Sampling to select a candidate best arm. However, rather than pull this arm, we form a particular upper bound on the probability of error that assumes the candidate as the best arm. The method then applies the principle of optimism in the face of uncertainty to pick an arm that could reduce the given upper bound the most. We compare our proposal to a number of algorithms in the literature to empirically evaluate the methods performance.

Algorithm Description

Maximum Boundary of Pairs is a Bayesian method that employs ideas such as Thompson Sampling and the principle of optimism in the face of uncertainty. For each arm the algorithm stores a posterior distribution modelling the belief of what the mean payoff for the arm is. In this section, the bandit is assume to have Bernoulli distributed rewards. We therefore use Beta distributions to model these beliefs.

At each round the algorithm first draws a sample from each posterior. Each sample is used as an estimate for the mean payoff of an arm. The arm associated with the highest sample is selected as the candidate best arm. This is the Thompson Sampling procedure. However, in MBoP this arm is not necessarily pulled. Let c_t be the candidate chosen by Thompson Sampling for a round t. If c_t is truly the best arm we will improve our final decision if we reduce the probability of mistaking any of the other arms as being better than c_t. This can be done by either improving the confidence we have in our estimate of the "best arm" so that we are more sure the arms mean is high relative to the others. Alternatively we could remove the uncertainty of the arm most likely to be mistaken with it (this is only mistaken in the sense of assuming c_t is the best arm).

For each of the non-candidate arms we find a point $b_{i,t}$ that minimizes the expression $F^{\text{Beta}}_{\alpha_{t,c_t},\beta_{t,c_t}}(b_{i,t}) + 1 - F^{\text{Beta}}_{\alpha_{t,i},\beta_{t,i}}(b_{i,t})$. The expression upper bounds the probability of each arm i being mistaken as better than the candidate c_t. Letting $b_{\max,t}$ be the largest $b_{i,t}$ we then compute $F^{\text{Beta}}_{\alpha_{t,i},\beta_{t,i}+1}(b_{\max,t}) - F^{\text{Beta}}_{\alpha_{t,i},\beta_{t,i}+1}(b_{\max,t})$ for each non-candidate arm. The procedure then finds the arm n_t with the largest

such quantity. Each non-candidate arm is considered sub-optimal and so we wish to reduce our belief that they could have a true mean higher than the "optimal" candidate arm. A reward of 0 will reduce this belief and reward of 1 will increase it. We then optimistically assume that we receive a reward of 0 which will reduce our belief as desired. $F^{\text{Beta}}_{\alpha_{t,i},\beta_{t,i}+1}(b_{\max,t}) - F^{\text{Beta}}_{\alpha_{t,i},\beta_{t,i}+1}(b_{\max,t})$ can be viewed as an estimate of how much this belief will be reduced. The number of pulls of n_t is compared with the number of pulls of c_t, and the arm pulled least so far is then pulled.

The algorithm is shown in Algorithm 5.4. A visualisation of the selection procedure for a single round is shown in Figure 5.12.

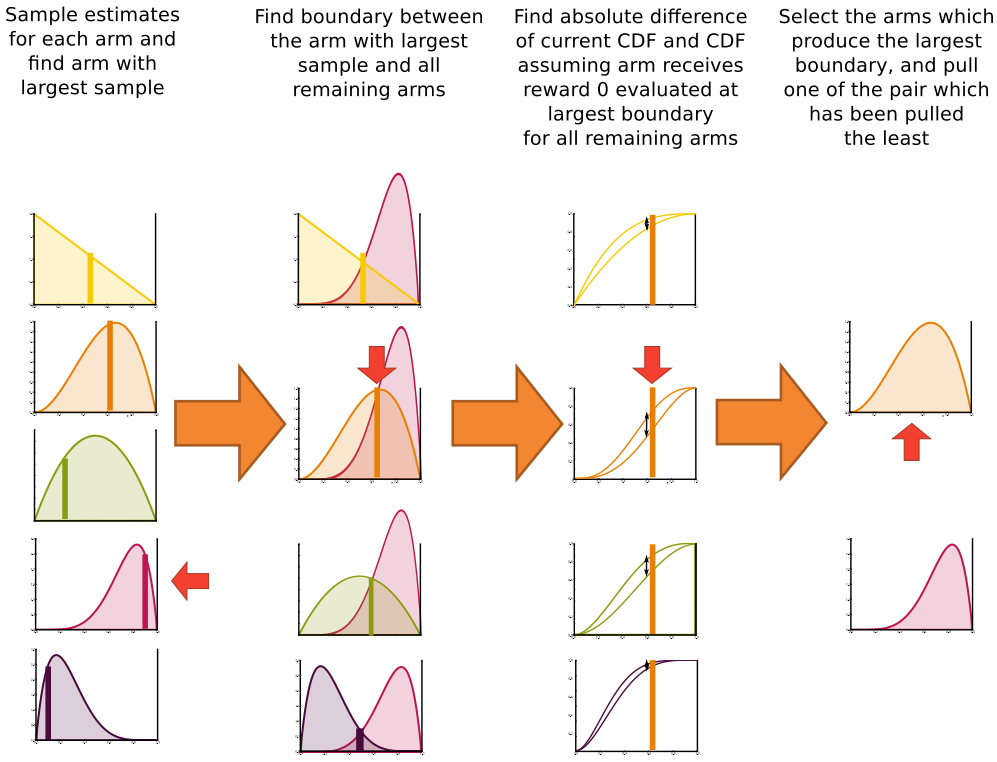


Figure 5.12: A visualisation of the MBoP strategy. The uncertainty in each arm is modelled by a Beta posterior. Thompson Sampling is used to first find the "best" arm. The decision boundary between the "best" arm and all other arms is found. The "best" arm and one other arm have the largest boundary value. The arm pulled least between these two is chosen as the next arm to pull.

We can view this procedure as optimistically reducing an upper bound on the

error, assuming the candidate arm c_t is in fact the best arm.

$$P_e = P\left(\bigvee_{i\in\{2,\dots,K\}} \mu_{\sigma(i)} > \mu_{\sigma(1)}\right) \tag{5.38}$$

$$\leq \sum_{i\in\{2,\dots,K\}} P\left(\mu_{\sigma(i)} > x\right) + P\left(x > \mu_{\sigma(1)}\right) \tag{5.39}$$

$$= F^{\text{Beta}}_{\alpha_{c_t},\beta_{c_t}}(x) + \sum_{i\in\{1,\dots,K\}\setminus c_t} \left(1 - F^{\text{Beta}}_{\alpha_i,\beta_i}(x)\right), \tag{5.40}$$

where $x = \max_i b_{i,t}$. This bound is optimistically reduced, since the two arms than can reduce this bound the most are c_t or n_t, whichever has been pulled the least.

Algorithm 5.4 Maximum Boundary of Pairs + Empirically Best Arm

Let $\alpha_{1,k} = 1$,
$\beta_{1,k} = 1$ for $k \in \{1,\dots,K\}$.

Exploratory Phase:
for $t = 1,\dots,T-1$ **do**
Sample $\theta_{t,i} \sim \text{Beta}(\alpha_{t,i},\beta_{t,i})$, for $i \in \{1,\dots,K\}$.
Let $c_t = \text{argmax}_i\, \theta_{t,i}$
Find $b_{i,t} = \min_b F^{\text{Beta}}_{\alpha_{t,c_t},\beta_{t,c_t}}(b) + 1 - F^{\text{Beta}}_{\alpha_{t,i},\beta_{t,i}}(b)$, for $i \in \{1,\dots,K\}\setminus\{c_t\}$.
Let $b_{\max,t} = \max_i b_{i,t}$
Let $n_t = \text{argmax}_i\, F^{\text{Beta}}_{\alpha_{t,i},\beta_{t,i}+1}(b_{\max,t}) - F^{\text{Beta}}_{\alpha_{t,i},\beta_{t,i}+1}(b_{\max,t})$.
Pull $a_t = \text{argmin}_{i\in\{c_t,n_t\}}\, \alpha_{t,i} + \beta_{t,i}$.
Let $\alpha_{t+1,a_t} = \alpha_{t,a_t} + \mathbb{1}(x_{a_t}(t) = 1)$
$\beta_{t+1,a_t} = \beta_{t,a_t} + \mathbb{1}(x_{a_t}(t) = 0)$
Let $\alpha_{t+1,j} = \alpha_{t,j}$
$\beta_{t+1,j} = \beta_{t,j}$ for $j \in \{1,\dots,K\}\setminus\{a_t\}$.
end for

Final Decision:
Let $\Psi(T)$ be $\text{argmax}_i\, \hat{X}_{i,T}$

Computing $b_{i,t}$

For Bernoulli arms, when the posterior beliefs are modelled as Beta distributions, the value of the mean is bounded in the interval $(0,1)$. $b_{i,t}$ is the point that minimizes $F^{\text{Beta}}_{\alpha_{t,c_t},\beta_{t,c_t}}(b) + 1 - F^{\text{Beta}}_{\alpha_{t,i},\beta_{t,i}}(b)$. This point can be found by a binary

search within an appropriate interval in $(0, 1)$. It can therefore be computed reasonably efficiently.

For arms with Normally distributed rewards $b_{i,t}$ can be found in closed-form, and so using this form of the algorithm is extremely computationally efficient. Let μ_1,σ_1^2 and μ_2,σ_2^2 be the hyperparameters for the belief distributions of the two arms associated with $b_{i,t}$. $b_{i,t}$ is one of the, at most two, solutions to the equation,

$$\frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right) = \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{(x-\mu_2)^2}{2\sigma_2^2}\right). \tag{5.41}$$

The psuedocode for the normal approximated variant of the algorithm is given in 5.5.

Analysis

Lemma 5.4.1. *Let $\{N_{i,t}\}$ be the frequencies of pulls for all the arms until time t. If $N_{1,t} \geq N_{2,t} \geq \cdots \geq N_{K,t}$, so that $N_{1,t}$ is the most any arm has been pulled then $N_{1,t} - N_{2,t} \leq 1$*

Proof. The proof follows by induction over t. For $t = 1$ and $t = 2$ the proposition is clearly true since at $t = 1$ all $N_{i,1} = 0$, and at $t = 2$ only one arm has been pulled. This shows the base case.

Given at t the proposition is true, we only need to consider two cases.
Case 1: $N_{1,t} \neq N_{2,t}$,
By definition of the algorithm the arm pulled can next can not be most pulled arm. Assuming the proposition in this case $N_{1,t} - N_{2,t} = 1$, so either $N_{1,t+1} - N_{2,t+1} = 1$ or $N_{1,t+1} - N_{2,t+1} = 0$.
Case 2: $N_{1,t} = N_{2,t}$,

In this case, the most pulled arm can only increase by one, and so the proposition much hold. □

Lemma 5.4.2. *For fixed K the true best arm is pulled $\Omega(T)$ times in the Thompson Sampling strategy.*

Proof. Let N^* be the number of times the best arm is pulled. The expected number of times a sub-optimal arm, k is pulled, $\mathbb{E}[N_k] = O(\ln T)$, that is $\mathbb{E}[N_k] \leq C_k \ln T$ for some constant C dependent on the arm gaps, Δ_k. This means that $\mathbb{E}[N_*] \geq T - D \ln T$ for some constant D in terms of all Δ_k and K. For $T >> D$

Algorithm 5.5 Normal Maximum Boundary of Pairs + Empirically Best Arm for Bernoulli bandits

Let $s^2 = 0.25$ be a known variance of the rewards
Let $d_{1,k} = 0$,
$\mu_{1,k} = 0.0$,
$\sigma^2_{1,k} = 1$ for $k \in \{1, \dots, K\}$.

Exploratory Phase:
for $t = 1, \dots, T-1$ **do**
Sample $\theta_{t,i} \sim \mathcal{N}(\mu_{t,i}, \sigma_{t,i})$, for $i \in \{1, \dots, K\}$.
Let $c_t = \text{argmax}_i\, \theta_{t,i}$
Find $b_{i,t} = \min_b F^{\text{Normal}}_{\mu_{t,c_t},\sigma_{t,c_t}}(b) + 1 - F^{\text{Normal}}_{\mu_{t,i},\sigma_{t,i}}(b)$, for $i \in \{1, \dots, K\} \setminus \{c_t\}$.
Let $b_{\max,t} = \max_i b_{i,t}$
Let $m_{t,i} = \frac{\mu_{t,i} s^2}{s^2 + \sigma^2_{t,i}}$
Let $v^2_{t,i} = \frac{\sigma^2_{t,i} s^2}{\sigma^2_{t,i} + s^2}$
Let $n_t = \text{arg!}\max_i F^{\text{Normal}}_{m_{t,i},v^2{}_{t,i}}(b_{\max,t}) - F^{\text{Normal}}_{\mu_{t,i},\sigma_{t,i}}(b_{\max,t})$.
Pull $a_t = \text{arg!}\min_{i \in \{c_t, n_t\}} d_{t,i}$.
Let $\mu_{t+1,a_t} = \frac{\mu_{t,a_t} s^2 + x_{a_t} \sigma^2_{t,a_t}}{s^2 + \sigma^2_{t,a_t}}$
$\sigma^2_{t+1,a_t} = \frac{\sigma^2_{t,a_t} s^2}{\sigma^2_{t,a_t} + s^2}$
$d_{t+1,a_t} = d_{t,a_t} + 1$
Let $\mu_{t+1,j} = \mu_{t,j}$
$\sigma^2_{t+1,j} = \sigma^2_{t,j}$
$d_{t+1,j} = d_{t,j}$ for $j \in \{1, \dots, K\} \setminus \{a_t\}$.
end for

Final Decision:
Let $\Psi(T)$ be $\text{argmax}_i\, \hat{X}_{i,T}$

this is effectively T. Since T is the largest possible value of pulls then it follows that $N^* = \Omega(T)$ □

Lemma 5.4.3. *For fixed K the true best arm is pulled $\Omega(T)$ times in the MBoP strategy.*

Proof. We know that the Thompson Sampling strategy pulls the optimal arm $\Omega(T)$ times. The MBoP strategy is the same as the Thompson Sampling strategy whenever the candidate arm c_t has been pulled less than the other arm, n_t, in the chosen pair. Every time n_t is chosen it increases the probability that a sample from that arm lies close to its true mean, this then does not effect the order of the number of times that c_t will be the optimal arm. Since all other arms can

not be pulled for too long without the optimal arm being pulled less times then the optimal arm must also be pulled $\Omega(T)$ times. □

5.4.1 Experiments

The experiments are identical to those by Audibert, Bubeck, and Munos [5] and similarly we report performance in terms of probability of error. This is the same as the experiments done for Ordered-Statistic Thompson Sampling. The experiments performed can be found in Table 5.1 in Section 5.2.2. In the experiments we compare our algorithms to Thompson Sampling [34], Successive Rejects [5] and BayesGap [27], for completeness descriptions are provided in algorithms 3.1, 2.17 and 2.15 respectively. Thompson Sampling provably can not asymptotically achieve exponentially diminishing probability of error in the best arm identification problem since it achieves the lower bound of regret in the multi-armed bandit problem. Successive Rejects has been proved to achieve exponentially diminishing probability of error as has BayesGap. In all experiments the parameter γ_{bgap} for Bayes Gap was set to $\gamma_{\text{bgap}} = 0.0001$.

Performance

The algorithms were run on all experiments described. For each experiment 50 permutations of the arms were created randomly. For each permutation the algorithm was run 1000 times to estimate the error rate, the error rates were then averaged over the 50 permutations. The results of these experiments are shown in Figure 5.13. We can see that in most of the experiments MBoP outperforms both Successive Rejects and BayesGap. The Normal MBoP variant does not perform as well, but still in most cases improves upon Successive Rejects.

5.4.2 MBoP summary

We have proposed a parameterless best arm identification algorithm. The algorithm is Bayesian and uses Thompson Sampling to help guide exploration. It combines this with the principle of optimism in the face of uncertainty, in the sense that for a particular upper bound of the probability of error, it chooses the action that potentially reduces the error the most. Empirical results show that this algorithm performs well in comparison to other algorithms. The choice of

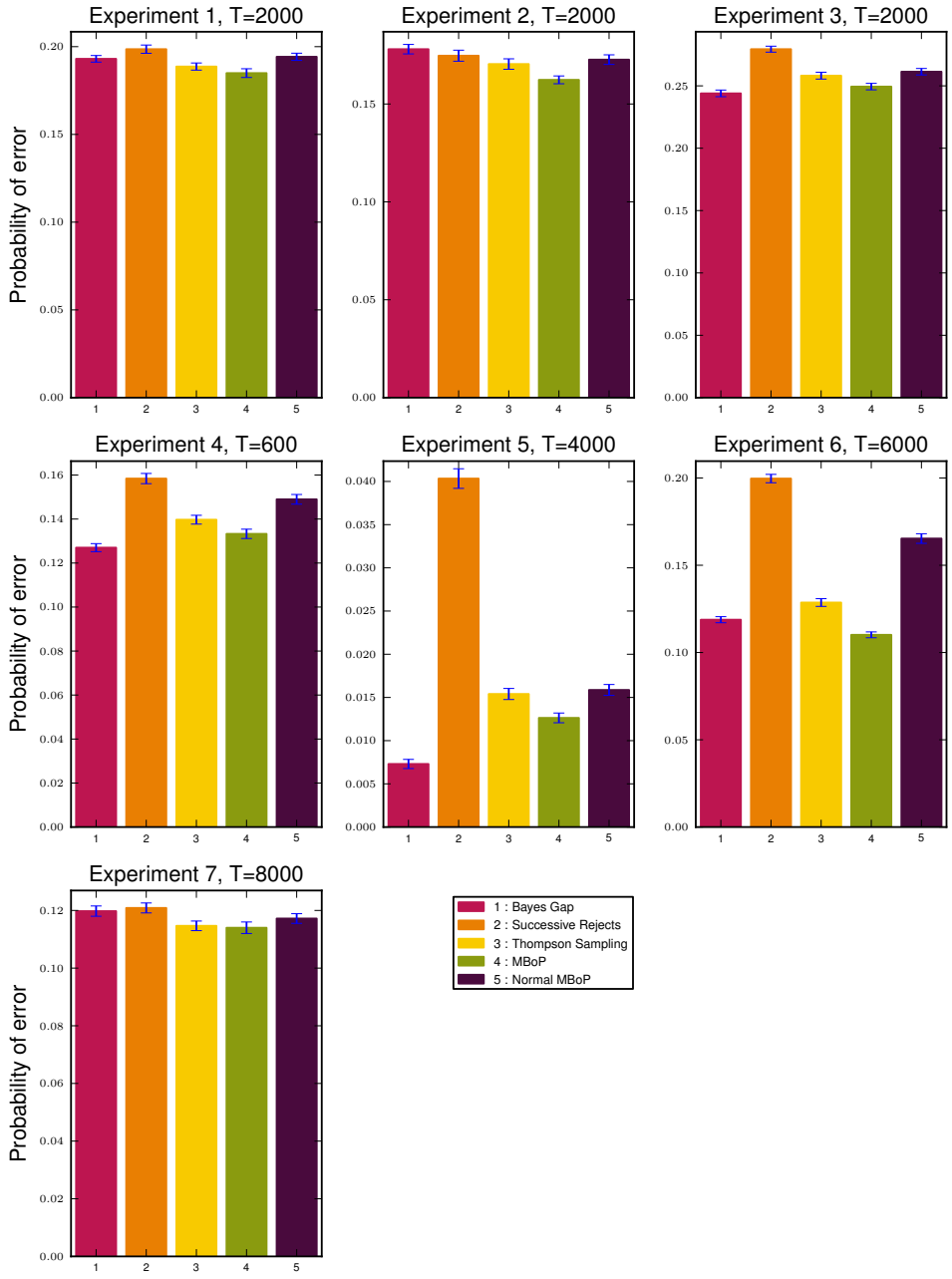


Figure 5.13: The bars show percentage of errors. For many of the experiments MBoP results in the lowest error. The normal version of MBoP also performs well on the test set, although with a slight increase in probability of error against the version using a Beta model. However, this cost to performance may be traded off against the decreased computational complexity. The performance for MBoP appears to be better or the same as the other algorithms. The exception being experiment 5.

error bound considered at each round could be improved. It would be of interest how a similar strategy using improved bounds would behave.

5.5 Conclusion

This chapter has explored using Thompson Sampling in the context of a best arm identification problem. The algorithm itself is known to be sub-optimal for this problem. This is because the strategy begins to exploit the arm it believes to be best too much rather than exploring. This leads to a polynomial reduction in probability of error, rather than exponential. In previous chapters we have discussed the benefits of Thompson Sampling and would like to leverage these benefits in the best arm identification problem.

Firstly we proposed Ordered-Statistic Thompson Sampling (OSTS). This is the most natural generalisation of Thompson Sampling suitable to the best arm identification problem. The algorithm is can be used when the time horizon is not known in advance. It is also efficient to compute. We provide a weak upper bound on the simple regret and show that it diminishes exponentially with the time horizon. Empirically we show that it is competitive with other algorithms in the literature. We go one to propose another algorithm, Anytime SR, which uses similar ideas. However, much of the randomisation present in OSTS has been removed. The goal of Anytime SR is to behave like Successive Rejects but without the need to know the time horizon. We again provide a weak bound on its probability of error and empirically show its strikingly close performance to Successive Rejects.

We also propose an alternative Thompson Sampling inspired algorithm called MBoP. This algorithm can also be seen to use the principle of optimism in the face of uncertainty. It is also an anytime algorithm. We find that it has strong empirically performance.

Chapter 6

Conclusion

Decision making is an important trait of any autonomous system. Learning to make decisions through experience provides a means to find ways to do this that are robust to noise, and are adaptable to any environment. We have highlighted the multi-armed bandit problem and the best arm identification problem (and related variants) as useful simple models in studying how to learn to make such decisions.

We have observed that Thompson Sampling is an effective means to tackle the multi-armed bandit algorithm. It is optimal for the Bernoulli armed bandit problem in the sense that it achieves the lower bound on expected cumulative regret. It is in many instances cheap to compute. Using conjugate priors, MCMC or variational techniques the ideas behind Thompson Sampling can model large numbers of decision-making problems, such as contextual bandit problems. It is robust to delays in rewards and can be made to account for dependency between arms. We extend the proof of Agrawal and Goyal to the optimistic variant of the algorithm and provide a modified form of optimism that also achieves the lower bound for cumulative regret. Surprisingly we also find that a Normal Thompson Sampling algorithm (one based on the Normal distribution as the conjugate prior) empirically outperforms the standard Thompson Sampling algorithm based on a Beta prior.

For the reasons mentioned above it is fruitful to propose Thompson Sampling inspired models for problems that go beyond the stochastic multi-armed bandit problem. Firstly we argue for the use of a switching multi-armed bandit problem as a more realistic and practical model in which to consider decision making.

We contribute a class of algorithms to tackle the switching multi-armed bandit algorithm that we call Changepoint Thompson Sampling. We empirically show the strategies are appropriate not only for the environment in which they are designed but also for a large number of environments that change in other ways. We find that in general the Per-Arm CTS strategy performs well in most situations with the Normal variant of this strategy performing the best in the PASCAL challenge. We also note some similarities between a parameterisation of our model and the Exp3 algorithm. The CTS model assumes a constant switch rate, that can be inferred from the data, but at a cost to the strategy due to the added uncertainty.

Secondly we consider adapting the Thompson Sampling algorithm to the best arm identification problem. We contribute two classes of strategy. The first, Ordered-Statistic Thompson Sampling, generalises Thompson Sampling, augmenting it with a rank distribution. We show that it is competitive with other best arm identification algorithms. In the investigation of this algorithm class we propose a measure of aggression. The measure is used to quantify the extent to which the algorithm attempts to pull higher ranking arms. In this terminology we find that the aggression is the most important factor determining the performance of OSTS and not the specific distribution chosen as the rank distribution. Unfortunately the level of aggression that is suitable appears to be depend on the problem, which is not to say that there is not an aggression level that gives the best worst-case expected probability of error of all problems. We establish a weak bound on the probability of error for this algorithm.

We improve upon our algorithm, OSTS, by biasing its behaviour towards aggression. The class of algorithm we call Aggressive OSTS. Again we empirically demonstrate the improvements of this approach.

We see that the long term behaviour of OSTS can be made to asymptotically approach that of Successive Rejects, another algorithm in the literature. However, a strong bound on the probability of error has not been found, and so it is still unclear whether the sample complexity can be made similar to Successive Rejects.

We investigate the use of optimism in the setting of the best arm identification problem. Optimism empirically results in lower cumulative regret in the multi-armed bandit problem. However, we see find that the improvement to the probability of error is marginal, if there is in fact any.

The second strategy we contribute is called MBoP. Whereas OSTS was a

natural generalisation of Thompson Sampling, MBoP instead uses Thompson Sampling as a subcomponent of the strategy. We find that MBoP has extremely promising empirical results, and requires no parameter tuning. We have also not established a strong lower bound on the probability of error for this algorithm, instead justifying the algorithm empirically.

We have shown in this thesis that the ideas of Thompson Sampling can also be beneficial to manage exploration in decision problems other than the stochastic multi-armed bandit problem. This is evidenced by the three algorithm classes mentioned above. We believe that the technique can be further adapted to other variants of similar problems.

6.1 Future work

We now consider further avenues for research that become apparent from the work presented in this thesis.

6.1.1 Thompson Sampling

Further empirical results for the observation that Normal Thompson Sampling outperforms Beta Thompson Sampling in the Bernoulli bandit problem would be natural to investigate. The theoretical question to ask is what is it about the problem and the assumptions made in the strategies that makes this so, since intuitively the Normal variant is assuming the "wrong" model.

6.1.2 Changepoint Thompson Sampling

We would like to provide theoretical analysis for our class of strategy. One approach would be bound a form of regret proposed by Garivier and Moulines [22]. Such an approach is likely to need different techniques than those used by Agrawal and Goyal in bounding Thompson Sampling in the stationary case. Such a bound may provide further insight into how the algorithm operates and might hint as ways in which the algorithm could be improved. However asymptotic bounds of this form of regret are only non-trivial when the number of switches that occur scale sub-linearly with time. Such an environment, although more amenable to analysis, is not perceived to the likely scenario in many practical situations. Investigation into alternative performance measures would be another useful direction to look.

Further comparison to Exp3 is also deemed to be beneficial. Two questions immediately come to mind. Is there a parametrisation of Changepoint Thompson Sampling for which we can bound some form of regret in the adversarial case, such as the weak regret? Also, if not, how could we adapt CTS so that it more appropriate for the adversarial case?

6.1.3 Ordered-Statistic Thompson Sampling

The most pressing question for Ordered-Statistic Thompson Sampling is what is a tight upper bound on its probability of error? This would have to be in terms of the rank distribution $\nu(\mathcal{K})$. The next question would be, is there a choice of $\nu(\mathcal{K})$ that can provable make the policy optimal in the sense that is defined by Audibert et al. [5]. Further analysis of the Anytime SR algorithm proposed may provide a stepping stone towards these results.

Another direction is to see how the strategy behaves in the M-best arm identification problem as investigated by Wang et al. [74].

It would also be interesting to see how this strategy might be generalised for a continuous armed bandit, where the concept of rank is more ill-defined.

6.1.4 MBoP

The MBoP algorithm would also benefit from further theoretical analysis including a tight bound on its probability of error.

Bibliography

[1] Ryan Prescott Adams and David J.C. MacKay. *Bayesian Online Changepoint Detection*. Cambridge, UK, 2007.

[2] Shipra Agrawal and Navin Goyal. "Analysis of Thompson Sampling for the Multi-armed Bandit Problem". In: *COLT*. Ed. by Shie Mannor, Nathan Srebro, and Robert C. Williamson. Vol. 23. JMLR Proceedings. JMLR.org, 2012, pp. 39.1–39.26.

[3] Shipra Agrawal and Navin Goyal. "Further Optimal Regret Bounds for Thompson Sampling". In: *CoRR* abs/1209.3353 (2012).

[4] D.J. Aldous. "Exchangeability and Related Topics". In: *École d'Été St Flour 1983*. Lecture Notes in Math. 1117. Springer-Verlag, 1985, pp. 1–198.

[5] Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. "Best Arm Identification in Multi-Armed Bandits". In: *COLT*. Ed. by Adam Tauman Kalai and Mehryar Mohri. Omnipress, 2010, pp. 41–53. ISBN: 978-0-9822529-2-5.

[6] Peter Auer. "Using Confidence Bounds for Exploitation-Exploration Trade-offs". In: *Journal of Machine Learning Research* 3 (2002), pp. 397–422.

[7] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. "Finite-time Analysis of the Multiarmed Bandit Problem". In: *Mach. Learn.* 47.2-3 (May 2002), pp. 235–256. ISSN: 0885-6125.

[8] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. "The Nonstochastic Multiarmed Bandit Problem". In: *SIAM J. Comput.* 32.1 (Jan. 2003), pp. 48–77. ISSN: 0097-5397.

[9] Dirk Bergemann and Juuso Valimaki. *Bandit Problems*. Cowles Foundation Discussion Papers 1551. Cowles Foundation for Research in Economics, Yale University, Jan. 2006.

[10] Mauro Birattari, Zhi Yuan, Prasanna Balaprakash, and Thomas Sttzle. "F-Race and Iterated F-Race: An Overview". In: *Experimental Methods for the Analysis of Optimization Algorithms*. Ed. by T. Bartz-Beielstein, M. Chiarandini, Lus Paquete, and M. Preuss. Berlin, Germany: Springer, 2010, pp. 311–336.

[11] G. E. P. Box and M. E. Muller. "A Note on the Generation of Random Normal Deviates". In: *The Annals of Mathematical Statistics* 29.2 (1958), 610–611.

[12] Monica Brezzi and Tze Leung Lai. "Incomplete Learning from Endogenous Data in Dynamic Allocation". English. In: *Econometrica* 68.6 (2000), pp. 1511–1516. ISSN: 00129682.

[13] Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. "Pure Exploration in Multi-armed Bandits Problems". In: *ALT*. Ed. by Ricard Gavaldà, Gábor Lugosi, Thomas Zeugmann, and Sandra Zilles. Vol. 5809. Lecture Notes in Computer Science. Springer, 2009, pp. 23–37. ISBN: 978-3-642-04413-7.

[14] J. Carpenter, P. Clifford, and P. Fearnhead. "Improved particle filter for nonlinear problems". In: *Radar, Sonar and Navigation, IEE Proceedings* - 146.1 (1999), pp. 2–7. ISSN: 1350-2395. DOI: 10.1049/ip-rsn:19990255.

[15] Olivier Chapelle and Lihong Li. "An Empirical Evaluation of Thompson Sampling". In: *Advances in Neural Information Processing Systems*. Ed. by John Shawe-Taylor et al. 2011, pp. 2249–2257.

[16] GSL Project Contributors. *GSL - GNU Scientific Library - GNU Project - Free Software Foundation (FSF)*. http://www.gnu.org/software/gsl/. 2010. URL: http://www.gnu.org/software/gsl/.

[17] Luc Devroye. *Non-Uniform Random Variate Generation*. New York: Springer-Verlag, 1986.

[18] Dukascopy. *Dukascopy Historical Data Feed*. http://www.dukascopy.com/swiss/english/marketwatch/historical/. [Online; accessed 05-Oct-2012]. 2012.

[19] Paul Fearnhead and Zhen Liu. "On-line inference for multiple change points problems". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 69 (2007), pp. 589–605.

[20] D. Fink. *A Compendium of Conjugate Priors*. Tech. rep. 1995.

[21] Victor Gabillon, Mohammad Ghavamzadeh, Alessandro Lazaric, and Sébastien Bubeck. "Multi-Bandit Best Arm Identification". In: *Advances in Neural Information Processing Systems*. Ed. by John Shawe-Taylor et al. 2011, pp. 2222–2230.

[22] Aurélien Garivier and Eric Moulines. "On Upper-Confidence Bound Policies for Switching Bandit Problems". In: *ALT*. Ed. by Jyrki Kivinen, Csaba Szepesvári, Esko Ukkonen, and Thomas Zeugmann. Vol. 6925. Lecture Notes in Computer Science. Springer, 2011, pp. 174–188. ISBN: 978-3-642-24411-7.

[23] J. C. Gittins. "Bandit Processes and Dynamic Allocation Indices". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 41.2 (1979), pp. 148–177.

[24] Thore Graepel, Joaquin Quiñonero Candela, Thomas Borchert, and Ralf Herbrich. "Web-Scale Bayesian Click-Through rate Prediction for Sponsored Search Advertising in Microsoft's Bing Search Engine". In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. Ed. by Johannes Fürnkranz and Thorsten Joachims. Omnipress, 2010, pp. 13–20. ISBN: 978-1-60558-907-7.

[25] Ole-Christoffer Granmo and Stian Berg. "Solving non-stationary bandit problems by random sampling from sibling Kalman filters". In: *Proceedings of the 23rd international conference on Industrial engineering and other applications of applied intelligent systems - Volume Part III*. IEA/AIE'10. Cordoba, Spain: Springer-Verlag, 2010, pp. 199–208. ISBN: 3-642-13032-1, 978-3-642-13032-8.

[26] Cédric Hartland et al. "Change Point Detection and Meta-Bandits for Online Learning in Dynamic Environments". English. In: *CAp* (2007), pp. 237–250.

[27] Matthew W. Hoffman, Bobak Shahriari, and Nando de Freitas. "Best arm identification via Bayesian gap-based exploration". In: *CoRR* abs/1303.6746 (2013).

[28] Z. Hussain et al. *Exploration vs. Exploitation PASCAL Challenge*. `http://pascallin.ecs.soton.ac.uk/Challenges/EEC/`. 2006.

[29] Edwin T. Jaynes. "Prior probabilities". In: *IEEE Transactions on Systems Science and Cybernetics* 4 (1968), pp. 227–241.

[30] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001–. URL: `http://www.scipy.org/`.

[31] P. Joulani, A. György, and Cs. Szepesvári. "Online Learning under Delayed Feedback". In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. June 2013, pp. 1453–1461.

[32] Leslie Pack Kaelbling. "Associative Reinforcement Learning: A Generate and Test Algorithm". In: *Machine Learning*. 1994.

[33] Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. "On Bayesian Upper Confidence Bounds for Bandit Problems". In: *Journal of Machine Learning Research - Proceedings Track* 22 (2012), pp. 592–600.

[34] Emilie Kaufmann, Nathaniel Korda, and Rmi Munos. "Thompson Sampling: An Optimal Finite Time Analysis". In: *CoRR* abs/1205.4217 (2012).

[35] Godfrey Keller, Sven Rady, and Martin Cripps. "Strategic Experimentation with Exponential Bandits". In: *Econometrica* 73.1 (Jan. 2005), pp. 39–68.

[36] Donald E. Knuth. *The art of computer programming, volume 2 (3rd ed.): seminumerical algorithms*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997. ISBN: 0201896842.

[37] Ron Kohavi, Roger Longbotham, Dan Sommerfield, and RandalM. Henne. "Controlled experiments on the web: survey and practical guide". English. In: *Data Mining and Knowledge Discovery* 18.1 (2009), pp. 140–181. ISSN: 1384-5810.

[38] Nathaniel Korda, Emilie Kaufmann, and Rémi Munos. "Thompson Sampling for 1-Dimensional Exponential Family Bandits". In: *Advances in Neural Information Processing Systems*. Ed. by Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger. 2013, pp. 1448–1456.

[39] Narayan Kovvali, Mahesh K. Banavar, and Andreas Spanias. *An Introduction to Kalman Filtering with MATLAB Examples*. Synthesis Lectures on Signal Processing. Morgan & Claypool Publishers, 2013, pp. 1–81. ISBN: 9781627051408.

[40] Volodymyr Kuleshov and Precup Doina. "Algorithms for the multi-armed bandit problem". In: *Journal of Machine Learning Research* (2010).

[41] T.L Lai and Herbert Robbins. "Asymptotically efficient adaptive allocation rules". In: *Advances in Applied Mathematics* 6.1 (1985), pp. 4 –22. ISSN: 0196-8858.

[42] The Python Programming Language. *The Python Standard Library: `random` module*. `http://docs.python.org/2/library/random.html`. [Online; accessed 29-Jan-2014].

[43] Lihong Li. "Generalized Thompson Sampling for Contextual Bandits". In: *CoRR* abs/1310.7163 (2013).

[44] Lihong Li and Olivier Chapelle. "Open Problem: Regret Bounds for Thompson Sampling". In: *COLT*. Ed. by Shie Mannor, Nathan Srebro, and Robert C. Williamson. Vol. 23. JMLR Proceedings. JMLR.org, 2012, pp. 43.1–43.3.

[45] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. "A Contextual-bandit Approach to Personalized News Article Recommendation". In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. Raleigh, North Carolina, USA: ACM, 2010, pp. 661–670. ISBN: 978-1-60558-799-8.

[46] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. "Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms." In: *WSDM*. Ed. by Irwin King, Wolfgang Nejdl, and Hang Li. ACM, 2011, pp. 297–306. ISBN: 978-1-4503-0493-1.

[47] Kevin Lloyd and David S. Leslie. "Context-dependent decision-making: a simple Bayesian model". In: *Journal of The Royal Society Interface* 10.82 (2013).

[48] Po-Ling Loh and Sebastian Nowozin. "Faster Hoeffding Racing: Bernstein Races via Jackknife Estimates." In: *ALT*. Vol. 8139. Lecture Notes in Computer Science. Springer, 2013, pp. 203–217. ISBN: 978-3-642-40934-9.

[49] Odalric-Ambrym Maillard, Rémi Munos, and Gilles Stoltz. "A Finite-Time Analysis of Multi-armed Bandit Problems with Kullback-Leibler Divergences". In: *Journal of Machine Learning Research - Proceedings Track* 19 (2011), pp. 497–514.

[50] Shie Mannor, Nathan Srebro, and Robert C. Williamson, eds. *COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland*. Vol. 23. JMLR Proceedings. JMLR.org, 2012.

[51] George Marsaglia and Wai Wan Tsang. "A Simple Method for Generating Gamma Variables". In: *ACM Trans. Math. Softw.* 26.3 (Sept. 2000), pp. 363–372. ISSN: 0098-3500.

[52] Makoto Matsumoto and Takuji Nishimura. "Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudo-random Number Generator". In: *ACM Trans. Model. Comput. Simul.* 8.1 (Jan. 1998), pp. 3–30. ISSN: 1049-3301.

[53] Benedict C. May, Nathan Korda, Anthony Lee, and David S. Leslie. "Optimistic Bayesian Sampling in Contextual-Bandit Problems 13(1)". In: *Journal of Machine Learning Research* 98888 (June 2012), pp. 2069–2106. ISSN: 1532-4435.

[54] Joseph Mellor and Jonathan Shapiro. "Thompson Sampling in Switching Environments with Bayesian Online Change Detection". In: *AISTATS*. Vol. 31. JMLR Proceedings. JMLR.org, 2013, pp. 442–450.

[55] Oded Maron & Andrew Moore. "Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation". In: *Advances in Neural Information Processing Systems*. Ed. by Jack D. Cowan, G. Tesauro, J. Alspector. Vol. 6. 340 Pine Street, San Francisco, CA 94104: Morgan Kaufmann, 1994, pp. 59–66.

[56] The Numerical Algorithms Group (NAG). *The NAG C Library (2013)*. Oxford, United Kingdom. URL: `http://www.nag.com`.

[57] José Niño-Mora. "A $(2/3)n^3$ Fast-Pivoting Algorithm for the Gittins Index and Optimal Stopping of a Markov Chain". In: *INFORMS J. on Computing* 19.4 (Oct. 2007), pp. 596–606. ISSN: 1526-5528.

[58] J. Pitman. *Combinatorial stochastic processes*. Vol. 1875. Lecture Notes in Mathematics. Lectures from the 32nd Summer School on Probability Theory held in Saint-Flour, July 7–24, 2002, With a foreword by Jean Picard. Berlin: Springer-Verlag, 2006, pp. x+256. ISBN: 978-3-540-30990-1; 3-540-30990-X.

[59] T. Preis, J. J. Schneider, and H. E. Stanley. "Switching processes in financial markets". In: *Proceedings of the National Academy of Sciences* 108.19 (Apr. 2011), pp. 7674–7678. ISSN: 1091-6490.

[60] Herbert Robbins. "Some aspects of the sequential design of experiments". In: *Bulletin of the American Mathematical Society* 58.5 (1952), pp. 527–535.

[61] Michael Rothschild. "A two-armed bandit theory of market pricing". In: *Journal of Economic Theory* 9.2 (1974), pp. 185–202.

[62] Daniel Russo and Benjamin Van Roy. "Learning to Optimize Via Posterior Sampling". In: *CoRR* abs/1301.2609 (2013).

[63] Steven L. Scott. "A modern Bayesian look at the multi-armed bandit". In: *Applied Stochastic Models in Business and Industry* 26.6 (2010), pp. 639–658. ISSN: 1526-4025.

[64] Satinder Singh, Tommi Jaakkola, Michael L. Littman, and Csaba Szepesvári. "Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms". In: *MACHINE LEARNING*. 1998, pp. 287–308.

[65] Morten Sorensen. *Learning by Investing: Evidence from Venture Capital*. SIFR Research Report Series 53. Institute for Financial Research, May 2007.

[66] Richard S. Sutton. "Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding". In: *Advances in Neural Information Processing Systems*. MIT Press, 1996, pp. 1038–1044.

[67] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[68] David B. Thomas, Wayne Luk, Philip H. W. Leong, and John D. Villasenor. " Gaussian random number generators". In: *ACM Comput. Surv.* 39.4 (2007), pp. 11+. ISSN: 0360-0300.

[69] William R. Thompson. "On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples". English. In: *Biometrika* 25 (1933), pp. 285–294. ISSN: 00063444.

[70] Long Tran-Thanh, Alex Rogers, and Nicholas R. Jennings. "Long-term Information Collection with Energy Harvesting Wireless Sensors: A Multi-armed Bandit Based Approach". In: *Autonomous Agents and Multi-Agent Systems* 25.2 (Sept. 2012), pp. 352–394. ISSN: 1387-2532.

[71] Ryan Turner, Yunus Saatci, and Carl Edward Rasmussen. "Adaptive Sequential Bayesian Change Point Detection". In: *Advances in Neural Information Processing Systems: Temporal Segmentation Workshop*. 2009.

[72] Joanns Vermorel and Mehryar Mohri. "Multi-armed Bandit Algorithms and Empirical Evaluation." In: *ECML*. Ed. by Joo Gama et al. Vol. 3720. Lecture Notes in Computer Science. Springer, Nov. 17, 2005, pp. 437–448. ISBN: 3-540-29243-8.

[73] Paolo Viappiani. "Thompson Sampling for Bayesian Bandits with Resets". In: *ADT*. Ed. by Patrice Perny, Marc Pirlot, and Alexis Tsoukiàs. Vol. 8176. Lecture Notes in Computer Science. Springer, 2013, pp. 399–410. ISBN: 978-3-642-41574-6.

[74] Tengyao Wang, Nitin Viswanathan, and Sébastien Bubeck. "Multiple Identifications in Multi-Armed Bandits". In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. Ed. by Sanjoy Dasgupta and David Mcallester. Vol. 28. JMLR Workshop and Conference Proceedings, 2013, pp. 258–265.

[75] Richard Weber. "On the Gittins Index for Multiarmed Bandits". In: *The Annals of Applied Probability* 2.4 (Nov. 1992), pp. 1024–1033.

[76] Robert C. Wilson, Matthew R. Nassar, and Joshua I. Gold. "Bayesian online learning of the hazard rate in change-point problems". In: *Neural Comput.* 22.9 (2010), pp. 2452–2476. ISSN: 0899-7667.

[77] Yahoo! *Yahoo! Webscope dataset ydata-frontpage-todaymodule-clicks-v1_0.* `http://labs.yahoo.com/Academic_Relations`. [Online; accessed 05-Oct-2012]. 2011.