Exploring Solutions to the Stochastic Multi-Armed Bandit Problem

A Literature Review and Simulation Study

Harry Spearing

STOR601 Research Topic II

Lancaster University

March 27, 2018

Abstract

This report begins with an introduction to machine learning and its various classifications, and the crossover between machine learning and statistical learning in multi-armed bandit problems. A literature review of solutions to multi-armed bandit problems are presented: the ϵ-greedy algorithm, Thompson sampling algorithms, and Upper-Confidence-Bound (UCB) algorithms.

A simulation is then formulated and coded in `R` and tested on a 10-armed Bernoulli bandit to compare these methods along with an algorithm introduced in this report, the greedy-η algorithm. It is shown that the greedy-η algorithm outperforms the standard ϵ-greedy algorithm after approximately a 10th of the horizon time, and will always outperform the ϵ-greedy algorithm asymptotically, independent of parameter selection. In a macro test of the code, it is shown that the greedy-η algorithm is more robust and its performance is less dependent on prior information. The simulation is based in a Bayesian setting with a Beta posterior distribution. The `R` code for this simulation can be found on my website: `http://www.lancaster.ac.uk/~spearing/#MAB`

Finally a comparison of these methods is presented based on theoretical guarantees on regret bounds, the simulation in this report, and other simulations in the literature.

STOR-i excellence with impact | Lancaster University

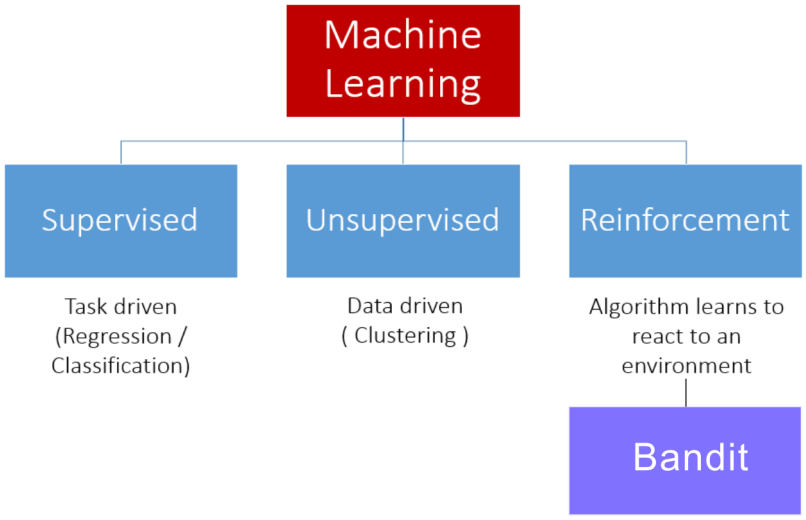

Figure 1: Machine learning subject area. Bandit algorithms can fall into machine learning or statistical learning.

1 Motivation

Consider a stage II clinical trial, where we would like to test multiple drugs on patients to determine which treatment is the best. The well-being of the patients is important, so we would like to treat patients with drugs that are believed to be optimum or near-optimum. It is also necessary that all proposed drugs are tested to some extent, in order to explore the potential for even better drugs so that future patients can benefit from superior treatment. This is an example of *exploration* vs. *exploitation*. To identify the best drug it is necessary to *explore*, whilst we simultaneously wish to *exploit* our current knowledge and treat patients effectively. It may seem that there is a positive correlation between exploration and exploitation, however this is generally not the case, as to determine the best drugs requires some patients to be treated via drugs which are expected to have a sub-optimal benefit. This is an example of a multi-armed bandit (MAB) problem, where each drug corresponds to an independent 'arm'. Villar (2015) reported "Our results indicate that bandit approaches offer significant advantages, in terms of assigning more patients to better treatments."

2 Overview

There are many examples in nature of learning by interacting with the environment, perhaps the most famous being Pavlov's dog (I. P. Pavlov 1927). Machine learning aims to capture this learning process within a mathematical or algorithmic framework. It is a broad field, exploring the construction of algorithms which can then be improved based on data, to make better predictions. There are 3 main fields in machine learning, described in figure 1: supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning uses a training data set provided by a knowledgeable supervisor. Each situation corresponds to a correct action that the system should take. The system aims to learn on the training set and then extrapolate to broader situations. The downside of supervised learning is that once the system moves outside the training set, it can learn no more.
Unsupervised learning aims to find hidden structure in an agents experience [3], for example cluster analysis [4].
Reinforcement learning is a third and separate classification of machine learning, and aims to maximise some reward. Reinforcement learning often includes some time dependent state-space, usually a Markov process, such that the reward at any time depends only on the action taken at that time,

$$r(t) = r(X_t) \tag{1}$$

and the actions available depend only on the previous action taken.

$$\begin{aligned} P(X_t \in \{X_1 ... X_k\}_t | X_{t-1} = x_t - 1, X_{t-2} = x_t - 2, \ldots, X_0 = x_0 \\ = P(X_t \in \{X_1 ... X_k\}_t | X_{t-1} = x_t - 1) \end{aligned} \tag{2}$$

In this report the state-space will be considered stationary such that all possible actions in the set are available to the system at all time steps. This special case is called a bandit problem and, for an action set greater than length one, a multi-armed bandit (MAB) problem. This specific problem has many applications, in clinical trials [1], Recommendation systems such as the Netflix Prize competition [5], and Resource allocation to name but a few.

Although reinforcement learning is generally accepted as being a classification of machine learning, many approaches to solving the MAB problem are classified as statistical learning. Whereas machine learning is based on generating point estimate predictions, statistical learning makes decisions based on the whole distribution of possible outcomes. In modern applications of statistical learning, methods must be computationally tractable, especially for on-line applications. There are several extensions to the MAB problem such as adversarial bandits where there is some dependence from one reward to the next, but this report will focus mainly on the standard stochastic multi-armed bandit.

3 Formulation

- There are K independent actions, or 'arms'.
- The bandit is played up to a horizon time $T \in \mathbb{R}$.
- At time t an action is chosen $X_t \in \{1, 2, \ldots, K\}$.
- $N_{i,t} = \sum_{t=0}^{t} \mathbb{I}_{x_t = i}$ represents the total number of times arm i has been played at time t.
- Reward observed at time t, r_t is independent from the history of the rewards and depends only on the action chosen.
- $\mathcal{H}_t = x_1, r_1, \ldots, x_{t-1}, r_{t-1}, x_t$ represents the whole history of the system up to the current state t.
- The probability of reward is modelled as coming from some posterior distribution,

  $$\pi(\theta | \mathcal{H}_t) \propto L(\theta | \mathcal{H}_t) \pi(\theta)$$

  such that $r_t \sim \theta_{X_t}$, where $\boldsymbol{\theta} \in \{\theta_1, \theta_2, \ldots, \theta_K\}$ are the parameters of the distributions of the corresponding arms at the current time.
- Finally, arm i has unknown but fixed true probability of reward P_i.

3.0.1 Bernoulli Bandit

Bernoulli bandits are special case where the reward $r_t \in \{0, 1\}$. This makes simulation more straightforward since the system can be moved into a Bayesian setting with Beta prior and posterior distributions, and of course the likelihood follows a Bernoulli distribution.
As such, at each time point the prior belief is updated by the likelihood of the current action to produce K independent posterior distributions for each arm. These posterior distributions dictate the the next action. This link between reward and action is the fundamental principle in reinforcement learning.

3.0.2 Regret

Regret is a measure of how much worse the algorithm performed given the prior information than if it had known the best strategy before-hand. The instantaneous regret $\mathcal{R}(t)$ at any time t is written as

$$\mathcal{R}(t) = \mathbb{E}[\max_k \theta_k - \theta_{x_t} | \pi(\theta)]$$

We are interested in the regret, that is sum of the instantaneous regret at each iteration.

$$\mathcal{R}(T) = \sum_{t=1}^{T} \mathcal{R}(t) = \sum_{t=1}^{T} \mathbb{E}[\max_k \theta_k - \theta_{x_t} | \pi(\theta)] \tag{3}$$

In this report, two measures of success are considered, by either maximising the average reward, or minimising the regret.

4 Gittins Index

Gittins and Jones (1974) showed that instead of solving the K-armed bandit problem, an optimal solution can be obtained via solving K 1-armed bandit problems. In order to solve the 1-armed bandit problem, a procedure must be developed to *calibrate* such a bandit.
Consider a one-armed bandit, $\mathcal{P}$ with mean $\hat{\theta}_P$, which is in competition with another bandit $\mathcal{Q}$, who has **known** success probability q. Clearly, if $q = 1$, bandit $\mathcal{Q}$ is preferable. If at any time t, $q \leq \hat{\theta}_P$, bandit $\mathcal{P}$ is preferable, as there is the prospect to explore band $\mathcal{P}$ whilst still exploiting the same amount. Therefore there must be some value $\hat{\theta}_P < q(\theta_P) < 1$ such that both $\mathcal{P}$ and $\mathcal{Q}$ are optimal. This value $q(\theta_P)$ is the Gittins Index of bandit $\mathcal{Q}$. The K-armed bandit problem is solved my playing the arm with the largest Gittins index at any time t. The Gittins index is calculated as

$$Q(x_i) = \sup_{\tau > 0} \frac{\mathbb{E}\left[\sum_{t=0}^{T} r_{i,t}\right]}{\sum_{t=0}^{T} \gamma^t} \tag{4}$$

where $r_{i,t} = a_{i,t}\gamma^t$. $a_{i,t}$ is the reward given by arm i at time t, and $r_{i,t}$ is the corresponding value of this we receive due to discounting.
Here γ^t is the discounting factor where $0 < \gamma < 1$.
The Gittins index is a belief-lookahead method. In these approaches, all future rewards are considered and are incorporated into a fully Bayesian setting where both exploitation and exploration are considered simultaneously. The arms are assigned an index, the Gittins index, based on their expected reward, and the arms are played in order of decreasing Gittins indices.
The Gittins Index approach relies on accurate informative prior distributions since all future decisions are made before the first arm is played. This means that there is always positive probability that a sub-optimal arm will be played infinitely often [9].
The method also requires a discounted infinite horizon objective, which in practise has to be artificially introduced and do not map well to realistic reward distributions[10].
The limited computational tractability of Gittins indices are one of the main reasons for its obsoleteness in modern applications of the MAB problem. Furthermore the tractability of this approach becomes increasingly impossible to follow when being applied to more general reinforcement learning problems. However by using Gaussian approximations, Brezzi and Lai [12] made an approximation to the Gittins index method. Despite this, Wang et al. (2005) states, "in all but trivial circumstances, there is no hope of exactly following an optimal action selection strategy"[4] and indeed, Powel (2007) states that 'Unfortunately, at the time of this writing, there do not exist easy to use software utilities for computing standard Gittins indices[11]. The Gittins Index is nontheless a beautiful solution to the standard multi-armed bandit problem.

The ϵ-Greedy algorithm.
At time t=0
$\hat{\boldsymbol{\theta}}_t = \mathbb{E}[\pi(\boldsymbol{\theta})]$
$x_t = \begin{cases} \text{unif}(\{1,\ldots,K\}) & \text{w/ probability } \epsilon \\ \arg\max_k \hat{\boldsymbol{\theta}}_t & \text{otherwise} \end{cases}$
At time t=1:T
$\hat{\boldsymbol{\theta}}_t = \mathbb{E}[\pi(\boldsymbol{\theta}\|\mathcal{H}_t)]$
$x_t = \begin{cases} \text{unif}(\{1,\ldots,K\}) & \text{w/ probability } \epsilon \\ \arg\max_k \hat{\boldsymbol{\theta}}_t & \text{otherwise} \end{cases}$

Table 1: The ϵ-greedy algorithm chooses a random arm with probability ϵ - the exploration. Otherwise, it simply chooses the arm whose posterior density has the highest expected reward - the exploitation. At $t = 0$, the judgement is based on the prior distribution.

5 Greedy

Since a fully belief-lookahead approach is often intractable, many heuristics have been developed. The Greedy algorithm is one of the simplest, and is an example of machine learning because every action it takes is based off a point estimate and it does not consider the whole reward distribution. It selects the arm with the highest expected reward based on some prior belief and forever plays this arm. This is an extreme example of exploitation over exploration. Of course, we would like to test if there are any better arms.

5.1 ϵ-greedy

The ϵ-Greedy algorithm detailed in table 5.1, tests for any better arms by playing a random arm with probability $0 \leq \epsilon \leq 1$. The value of epsilon determines how often the algorithm explores. with probability $1 - \epsilon$, the ϵ-greedy algorithm behaves as the standard greedy algorithm. The introduction of the ϵ means that this algorithm relies less heavily on informative prior distributions, unlike the standard greedy.
The value of ϵ determines the levels of exploration and exploitation in the algorithm, but this choice is fixed and it is difficult to determine which value of ϵ to choose, so some form of prior information is still necessary in practise in order to set the parameter ϵ. The ϵ-greedy algorithm is a *undirected* method, meaning that it explores randomly. This makes it extremely easy to implement since no form of ranking is required to determine which arm to explore. It can mean that it performs poorly if the difference between the expected rewards of the arms is large because it will explore arms which have already been excessively explored and are known to have low expected rewards.

5.2 greedy-η

The algorithm introduced in this report aims to sidestep this issue by using a time dependent parameter to determine the amount of exploration $\eta(t)$ and exploitation $1 - \eta(t)$. We wish that $\eta(t)$ should be a monotonically decreasing function of t such that $0 \leq \eta(t \in \mathbb{R}^+) \leq 1$. The aim is to determine a rate of decay such that $\mathcal{R}(T)$ is minimised. I present a monotonically decreasing function such that

$$\eta(t) = \exp\left(\frac{-\beta t}{T}\right)$$

Choosing an exponential function insures that the probability of exploration is bounded in [0,1]. It also allows the parameter β to be selected to optimise the regret. This method is also an undirected method, and should be improved by introducing a ranking system based on variance estimates which directs the exploration, but this will be left for further work.

The greedy-η algorithm (exponential).

Require: T(horizon)

$\eta(t) = \exp\left(\frac{-\beta t}{T}\right)$

At time t=0

$\hat{\boldsymbol{\theta}}_t = \mathbb{E}[\pi(\boldsymbol{\theta})]$

$x_t = \{$ unif$(\{1, \ldots, K\})$ w/ probability 1

At time t=1:T

$\hat{\boldsymbol{\theta}}_t = \mathbb{E}[\pi(\boldsymbol{\theta}|\mathcal{H}_t)]$

$$x_t = \begin{cases} \text{unif}(\{1, \ldots, K\}) & \text{w/ probability } \eta(t) \\ \arg\max_k \hat{\boldsymbol{\theta}}_t & \text{otherwise} \end{cases}$$

Table 2: The greedy-η algorithm selects a random arm with probability $\eta(t)$, which is a monotonically decreasing function (in this case exponential). Otherwise, it simply chooses the arm whose posterior density has the highest expected reward.

The Thompson Sampler.

Require: Ability to sample from distribution $\pi(\cdot)$

At time t=0

$\boldsymbol{s} \sim \pi(\boldsymbol{\theta})$

$x_t = \arg\max_k \boldsymbol{s}$

At time t=1:T

$\boldsymbol{s} \sim \pi(\boldsymbol{\theta}|\mathcal{H}_t)$

$x_t = \arg\max_k \boldsymbol{s}$

Table 3: At each iteration t, we take a sample s from each of the posterior distributions and play the arm with the largest sample. Because it samples, it naturally incorporates the variance of the posterior distribution which acts as the exploration bonus. At $t = 0$, it samples from the priors.

6 Thompson sampling

The Thompson sampler samples from the posterior distribution of each arm and plays the arm which returns the largest sample, thereby selecting an action according to the probability of it being optimal. It is a stochastic algorithm which considers the whole posterior distribution of each arm and not just a point estimate, thus Thompson sampling is a statistical learning algorithm.

It is a *myopic* method, meaning that it seeks to reduce the uncertainty of the arms 'in the right places'[16]. Literally meaning "short-sighted", myopic methods don't consider all future rewards directly like belief look-ahead methods, this means they are often easier to implement and computationally cheaper.

Thompson sampling, also referred to *posterior sampling* in the literature, was introduced around 85 years ago [15], but has not been largely studied with application to multi-armed bandit problems until recently.

Perhaps this is because Thompson sampling often has to rely on computational approaches. In the examples in this report, a Bernoulli bandit is considered and it is trivial to sample from the posterior distribution, but often Markov chain Monte Carlo (MCMC) methods are required to sample from the posterior. In most cases this is a very efficient approach, however it is possible that MCMC in itself becomes expensive, making Thompson sampling slower to converge in real time.

Two papers [14, 16] have proved asymptotic convergence, which in applications such as mining operations [13] can be sufficient because of the long-term nature of the operation. More recently however, Russo and Van Roy (2014) have provided some of the first theoretical properties of Thompson sampling in finite time by showing theoretical bounds on the Bayesian regret. Bayesian regret, or Bayesian

risk is defined as

$$\mathcal{R}(T) = \sum_{t=1}^{T} \mathbb{E}[\max_k \theta_k - \theta_{x_t}] \tag{5}$$

such that it is an expectation taken over all possible prior distributions. Note that in the asymptotic regime the Bayesian regret converges towards the regret because $\pi(\theta|\mathcal{H}_t) \rightarrow L(\theta|\mathcal{H}_t)$.

6.1 Optimistic Thompson Sampling

The random sampling element of Thompson sampling helps to minimise regret because it makes us more likely to select an action that we are uncertain about, because an action with large variance has a high chance of producing the largest sample. However this action also has large probability of producing a sample much lower. Since there appears to be no gain in underestimating, or being "pessimistic" about an arm, optimistic Thompson sampling, known as *Optimistic Bayesian sampling*[16] was introduced. This makes a very simple change to the Thompson sampling algorithm. Instead of selecting $x_t = \arg\max_k \boldsymbol{s}$, we select $x_t = \arg\max_k[\max(\boldsymbol{s}, \mathbb{E}[\boldsymbol{\theta}|\mathcal{H}_t])]$. In short, if the sample we get from the posterior is less than the expected reward, the we just take value to be the expected reward. Optimistic Bayesian sampling shows some marginal gains [8] however these simulations were done on a large number of arms, and it is therfore likely that the arm chosen is one that is already above its expected value and therefore the addition to the algorithm is redundant. It is expected that OBS would have a large difference when applied to sparse MAB.

6.2 Posterior Reshaping

The introduction of randomness through sampling in the Thompson sampler is one of its major benefits in terms of its robustness, but the posterior these samples are drawn from could be reshaped such that it suits the application. For example, sharpening the posterior would force the algorithm to exploit more, and flattening the posterior would cause the algorithm to explore more. In the Bernoulli bandit setting, the posterior distribution is a Beta(a,b) distribution. By introducing a parameter α such that the posterior becomes Beta(a/α,b/α), the expectation remains the same but the variance increases by a factor of α^2[27].

7 Upper-Confidence-Bound

Upper-confidence-bound (UCB) action selection is a statistical learning algorithm which forms an "optimistic" estimate of the reward, such that the system makes a selection as if the expected rewards are equal to the highest possible values that are compatible with the past observations[17]. Like Thompson sampling, UCB methods are myopic and are classified as statistical learning methods since they consider the whole distribution of rewards by introducing confidence bounds. They are not stochastic since they use an indexing approach rather than sampling.

It selects the arm at time t such that

$$x_t = \text{argmax}_k \left[\mathbb{E}[\pi(\boldsymbol{\theta}|\mathcal{H}_t)] + \alpha \sqrt{\frac{\ln(t)}{N_{k,t}}} \right] \tag{6}$$

Equation 6 is made up of two terms. The expectation term is the same as the greedy algorithm. The second term, the *bias* term, considers the distribution as a whole. Essentially, it adds some *exploration bonus* such that we are more likely to pick an arm with larger the variance since there is more to learn.

UCB is based on the idea of optimism in the face of uncertainty, but it is important that good actions are not ruled out just because they may do poorly early on. So we would like to know that the true mean reward P_i is less than the estimated upper bound with a high confidence. It is possible to bound the regret of the UCB algorithm in the form of equation (7) [19] when the reward, $r_t \in [0, 1]$.

$$\mathcal{R}(T) \leq \sum_{i:P_i < \text{ argmax}_k(P_k)} \frac{8 \ln(T)}{\text{argmax}_k(P_k) - P_i} + C \tag{7}$$

UCB algorithm
At time t=1:T
$x_t = \text{argmax}_k \left[\mathbb{E}[\boldsymbol{\theta}\|\mathcal{H}_{t-1}] + \alpha\sqrt{\frac{\ln(t)}{N_{k,t}}} \right]$

Table 4: The first term is an expectation which incorporates the exploitation, the second term inflates indexes with more uncertainty which incorporates the exploration.

The UCB algorithm has the bonus that we don't need to sample from the posterior distribution, unlike Thompson sampling, which can decrease computational cost in the case of a complex distribution, however it does introduce a parameter α which must be tuned accordingly, and determines how much weighting to give to the exploration.
The denominator of the bias term simply counts how many times that arm has been played. For the arm being played, the nominator increases at a slower rate and thus in the limit the variance of the optimum arm will decrease as approximately $\sqrt{\frac{\ln t}{t}}$. However for sub-optimal arms the nominator increases without bound in periods of inactivity. Essentially, the $\ln(t)$ term artificially inflates the variance such that even in the limit there is a non-zero probability of any arm being played, and infinite exploration is guaranteed. This ensures a zero-probability of playing a sub-optimal arm asymptotically.

7.1 UCB2

There are many variants of the UCB algorithm. In fact, the UCB algorithm detailed in the previous section is also referred to in the literature as UCB1, because UCB2 was introduced in the same paper[20]. UCB2 works by dividing the plays into "epochs". In each epoch, arm i is pulled $\sigma(d_i+1) - \sigma(d_i)$ times. Here, d_i is the number of epochs where arm i has been played, and σ is an exponential function, such that the machines are played for longer and longer periods and are therefore exploited more. The arm selected to play at each epoch satisfies $x_t = \text{argmax}_k \hat{\theta}_k + v_{t,d_i}$ where

$$v_{t,d_i} = \sqrt{\frac{(1+\alpha)\ln(T/\sigma(d))}{2\sigma(d)}}$$

where parameter $0 < \alpha < 1$.

7.2 UCB-Tuned

When used in practise, the bound on UCB can be more more closely approximated. The upper con fidence bound for machine i can be written as

$$V_i(N_{i,t}) = \mathbb{E}[\pi(\theta_i(t)^2|\mathcal{H}_t)] - (\mathbb{E}[\pi(\theta_i(t)|\mathcal{H}_t)])^2 + \sqrt{\frac{2\ln t}{N_{i,t}}}$$

which is the the sample variance plus the logarithmic term. Then the upper confidence bound $\sqrt{\frac{\ln t}{N_{i,t}}}$ from equation (6) becomes

$$\sqrt{\frac{\ln t}{N_{i,t}} \min\{\frac{1}{4}, V_i(N_i(t)\}}$$

This version of UCB, called UCB-Tuned, outperforms standard UCB in most simulations and was used to great effect in the 2006 Pascal Challenge [23] though theoretical regret bounds in finite time had not been produced at the time.

7.3 UCB-V

Equation (6) is actually a specific case for the reward $r_t \in [0, 1]$. For the more general case, $r_t \in [0, b]$ the variance term becomes [20]

$$\sqrt{\frac{b^2 \ln t}{N_{i,k}}}$$

and the expected regret from equation (7) in this case becomes

$$\mathbb{E}[\mathcal{R}(t)] \leq \sum_{i:P_i<\text{argmax}_k(P_k)} \frac{8b^2 \ln(T)}{\text{argmax}_k(P_k) - P_i} + C$$

That is, the regret actually scales with b^2 which becomes large for systems with larger rewards, and would be better to lessen the dependence on it. Audibert et al (2007) introduce a variant called UCB-V which reduces the scaling of the regret such that it is linear in b. UCB-V uses variance estimates such that the bias term has the form

$$\sqrt{\frac{2V_i(N_{i,t}\epsilon(N_{i,t})}{N_{i,t}}} + c\frac{2b\epsilon(N_{i,t})}{N_{i,t}}$$

Here $V_i(N_{i,t})$ denotes the variance in arm i with $N_{i,t}$ samples, and ϵ is the exploration function such that $\epsilon = \zeta \ln(t)$. With these variance estimates, and the tuning parameters selected sensibly, the bound on the expected regret [21] becomes

$$\mathcal{R}(T) \leq \sum_{i:P_i< \text{ argmax}_k(P_k)} \left(\frac{\tau_k^2}{\text{argmax}_k(P) - P_i} + 2b\right) \ln(T) \tag{8}$$

Where τ is the variance of the rewards. Now the regret scales with b, an improvement over b^2 scaling resulting from standard UCB. This improved scaling comes at the cost of now having two parameters: $\zeta, c > 0$. lower limits on these tuning parameters are given such that if the tuning parameters are set below these limits the regret increases sharply.

7.4 Minimax Optimal Strategy in the Stochastic case

Minimax Optimal Strategy in the Stochastic case (MOSS)[22] is a small variation on UCB. It still has an index for each arm and selects the highest index, however instead of $\ln(t)$ in equation 6, it uses $\ln(\frac{T}{KN_{i,t}}$. This ensures that actions which have been played a lot are less likely to be played, thus increasing the exploration.

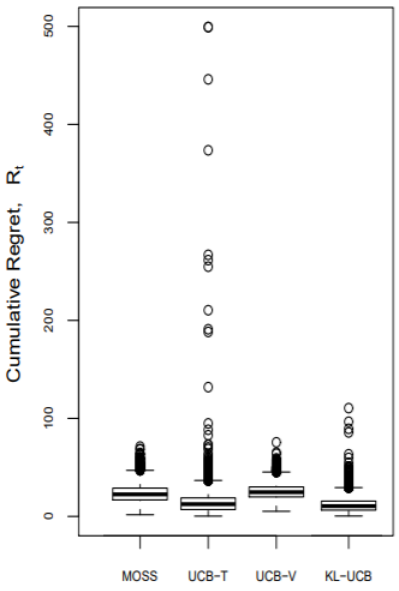

Figure 2: Results from a macro simulation comparing various Upper-Confidence-Bound methods [8]

7.5 Kullback-Leibler - Upper Confidence Bound Algorithm

The Kullback-Leibler divergence is a measure of how much two distributions diverge from one another. By incorporating this into the UCB type algorithm, it can consider the distance between the estimated distributions of each arm when calculating the exploration factor. The Bernoulli Kullback-Leibler divergence [29] for $i, j \in \{1, \ldots, K\}$ is given by,

$$d(\theta_i, \theta_j) = \theta_i \log \frac{\theta_i}{\theta_j} + (1 - \theta_i) \log \frac{1 - \theta_i}{1 - \theta_j}$$

The KL-UCB algorithm is asymptotically optimal for the multi-armed Bernoulli bandit, in that $\mathcal{R}(T)$ is minimised with upper bound,

$$\lim_{T\to\infty} \frac{\mathbb{E}[\mathcal{R}(T)]}{\log(T)} \leq \sum_{i:P_i < \operatorname{argmax}_k(P_k)} \frac{\operatorname{argmax}_k(P_k)P_i}{d(P_i,\ \operatorname{argmax}_k(P_k))} \tag{9}$$

The regret in this method matches the lower-bound provided by Burnetas and Katehakis (1996). These bounds apply to any variable in [0,1], not just Bernoulli. KL-UCB has the best regret bounds of all UCB type algorithms.
The results of the simulation shown in figure 7.4 show the regret over many replications of MOSS, UCB-V, UCB-Tuned and KL-UCB. It can be seen that although UCB-Tuned and KL-UCB have lower expected regret than the other Moss and UCB-V, the spread is much larger and therefore they are more "risky" approaches as there is a larger probability of returning a very large regret. Moss and UCB-V perform comparatively. KL-UCB has the lowest expected regret.

8 Simulation

The simulation detailed in this report follows a K-armed Bernoulli bandit structure. $K = 10$ arms were randomly generated with each arm i having a probability of success drawn from $P_i \sim U(0,1)$. The best arm had a reward probability of $\operatorname{argmax}_k(P) = 0.78$. The initial prior beliefs followed a Beta(1,1) distribution

8.1 ϵ - Greedy: simulation performance

Figure 3 shows the average reward received for various values of ϵ and illustrates the sensitivity of the parameter. The black line, $\epsilon = 0.5$, converges very quickly to the best arm, but it can then only exploit this arm 50% of the time and thus its average reward is very low. Comparatively, the average reward for $\epsilon = 0.05, 0.1$ initially converges to a sub-optimal arm since they do not explore very much. It is only after 2700 plays that both variations switch to playing the optimal arm.
This can be seen more clearly by analysing the regret for all the different values of ϵ in figure 3. For $\epsilon = 0.05, 0.01$ the regret begins to converge to ≈ 0.1 up until 2700 iterations, where their behaviour changes and the regret decreases suddenly, indicating that the previous behaviour was exploiting a sub-optimal arm. The $\epsilon = 0.1$ variation temporarily outperforms the $\epsilon = 0.05, 0.01$ because its extra exploration means that the optimal arm is found at 1200 plays.

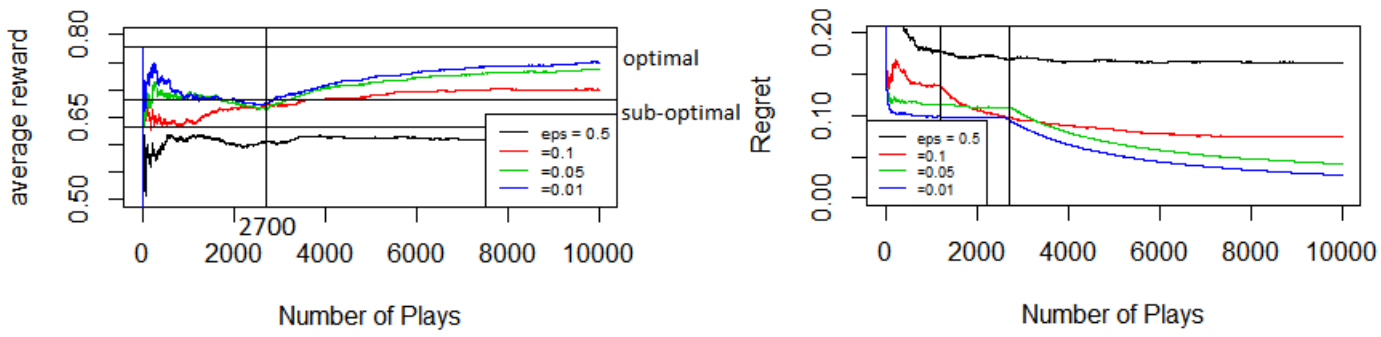

Figure 3: ϵ-Greedy algorithm with varying values of ϵ.

The asymptotic regret depends only on the value of ϵ since this determines how often the algorithm is allowed to exploit, therefore a smaller ϵ results in a smaller regret asymptotically. However the speed of convergence is also solely dependent on ϵ, such that a smaller ϵ results in longer time to convergence. This can be seen in figure **??** by the speed at which the 0.5-greedy algorithm converges. The 0.5-greedy algorithm has converged in 100 iterations, compared to 2700 for greedy-0.05 and greedy-0.01, and approximately 1200 for 0.1-greedy. Note that for any $\epsilon > 0$ the regret can never converge to 0.

8.2 greedy-η: simulation performance

We have seen that choosing a fixed ϵ forces a compromise between time-to-convergence and minimizing long term regret. It would seem much more intuitive to explore more initially and then exploit more as t increases. This report introduces the greedy-$\eta(t)$ algorithm.

Figure 4 demonstrates how the greedy-$\eta(t)$ algorithm out performs the ϵ-greedy algorithms for any time after $\approx$ 1500 plays. For this test, the parameter was set to $\beta = 10$, but it will be shown in section 9 that this algorithm is robust, and that it will still outperform ϵ-greedy algorithm for a very broad range of β. The greedy-$\eta(t)$ approach also has a much faster time to convergence, noting that the optimal arm was found after just 100 iterations. Unlike ϵ-greedy algorithms, the regret of the greedy $\eta(t)$ algorithm converges to 0, because $\eta(t)$ converges to 0.

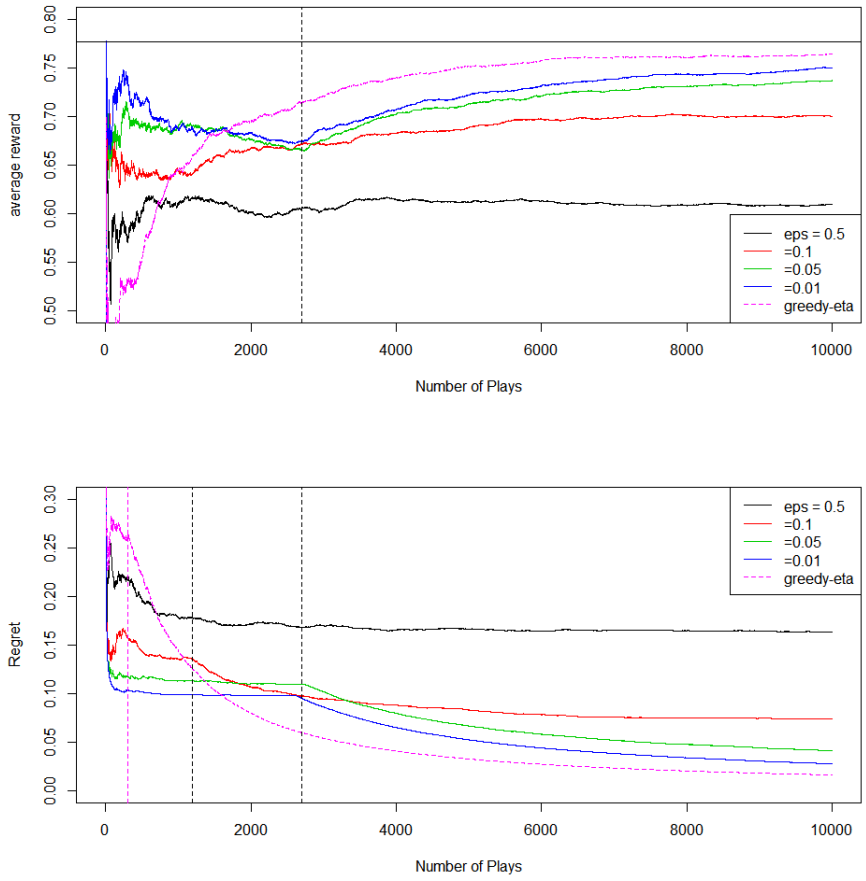

Figure 4: The greedy-$\eta(t)$ algorithm that this report demonstrates the benefit of having time dependent element in the probability of exploration since it improves on both the long-term mean reward and the long-term regret of the ϵ-greedy algorithm.

8.3 Thompson Sampling: simulation performance

In this report, the Beta distribution made sampling from the posterior computationally fast and so no MCMC or Sequential Monte Carlo methods were required. Figure 5a shows the average reward for each of the ϵ-greedy, greedy-η and Thompson sampler. For this comparison both the ϵ-greedy and greedy-$\eta(t)$ algorithms were tuned such that the parameters ϵ, β maximised and minimised the average reward and long-term regret respectively. This helps to illustrate the performance of the Thompson sampler, which has a higher average reward than a tuned ϵ-greedy algorithm. Figure 5 shows the regret of each algorithm. We can see a kink in the greedy-η line at around 170 iterations, indicating a switch to the optimum arm, however the Thompson sampler converges to the optimum arm at only 80 iterations, resulting in a much lower regret for the Thompson sampler. Both the Thompson sampler

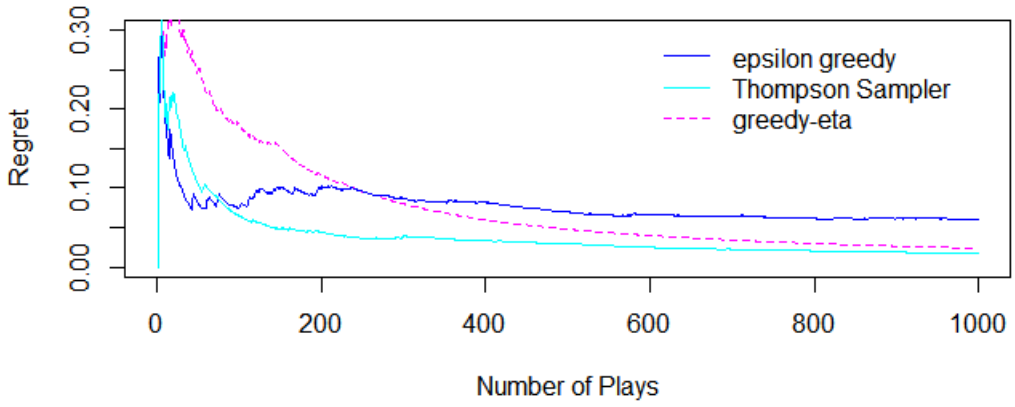

Figure 5: Comparison of ϵ-greedy, greedy-η and Thompson sampler over 1000 iterations.

and the greedy-η algorithm converge to 0 regret because of the nature of the increasing exploitation, where as the ϵ-greedy algorithm can never have 0 regret.

8.4 Upper-Confidence-Bound: simulation performance

Figure 6 compares all the algorithms with their tuned parameters. The UCB algorithm is superior in this context, with both a higher average reward for the whole simulation, and a lower regret after 80 iterations.

8.5 Macro Test

To test the robustness of these methods, the simulation was run in a macro test with each simulation taking common random numbers. Standard UCB, ϵ-greedy, Thompson sampling and greedy-η were compared. Figure 7 shows the result of the macro experiment with flat priors. The shaded areas represent a 95% confidence interval. Thompson sampling is always constant over the range of parameters since in its simplest form does not contain any parameters. It is important to note that the confidence intervals overlap for much of the parameter space, meaning that it is difficult to say which methods are best, though it appears that Thompson sampling out-performs all the other methods, apart from UCB which performs comparably when its parameter is chosen accordingly. Note the logarithmic scale an which the parameter space is represented.
Another experiment was tested with informative priors. Beta prior distributions were generated for each arm such that the parameters had the same mean as the true reward probability plus some normal noise.
Figure 8 shows the result of this macro test. It is clear that here the optimum parameters are set to exploit more since the optimum values of ϵ, c are smaller and β larger. With informative priors, all the algorithms can out-perform Thompson sampling at their optimum parameters. This may be

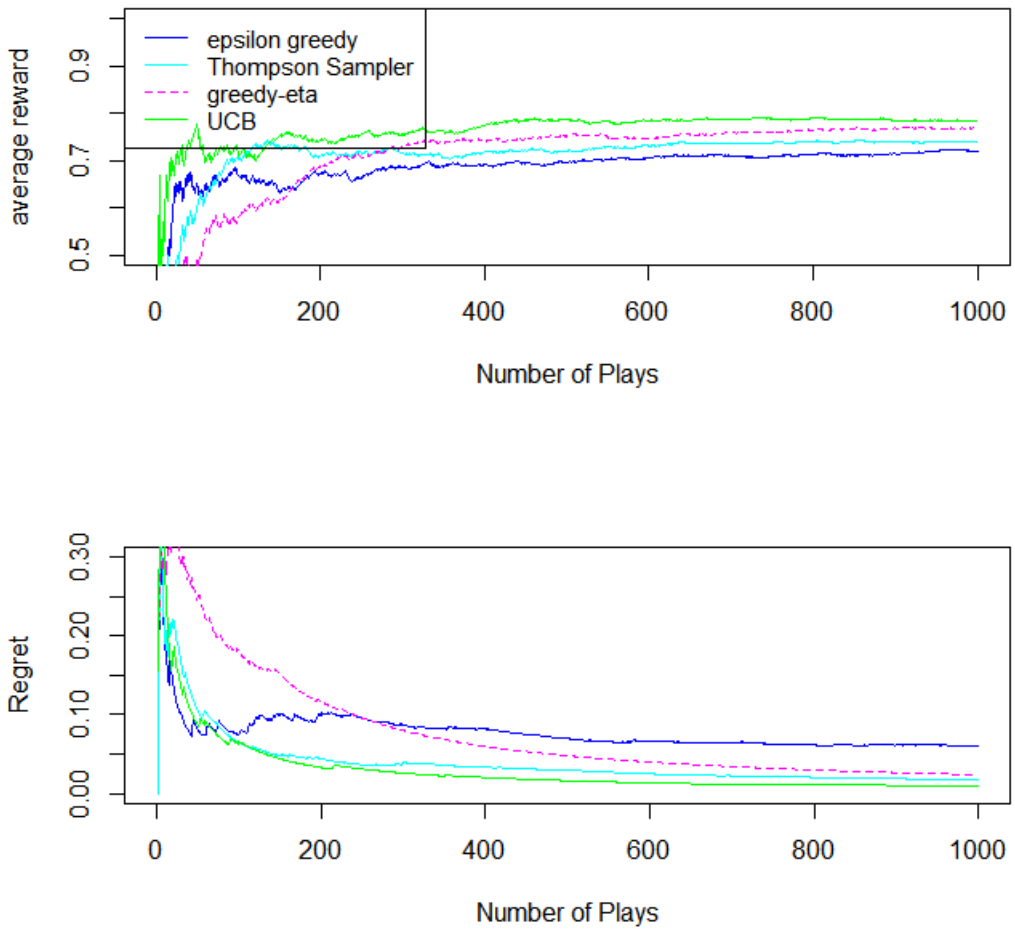

Figure 6: Comparison of all methods at their tuned parameters for 1000 iterations.

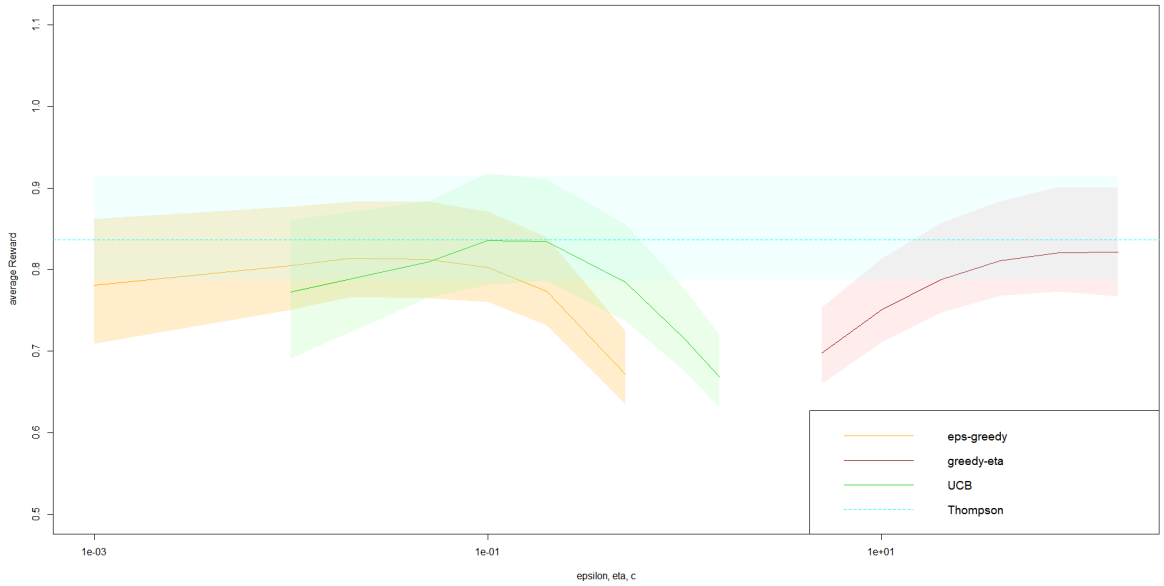

Figure 7: The average reward for horizon time $T = 1000$. **Uninformative** prior macro experiment ran for 500 simulations over each combination of parameters.

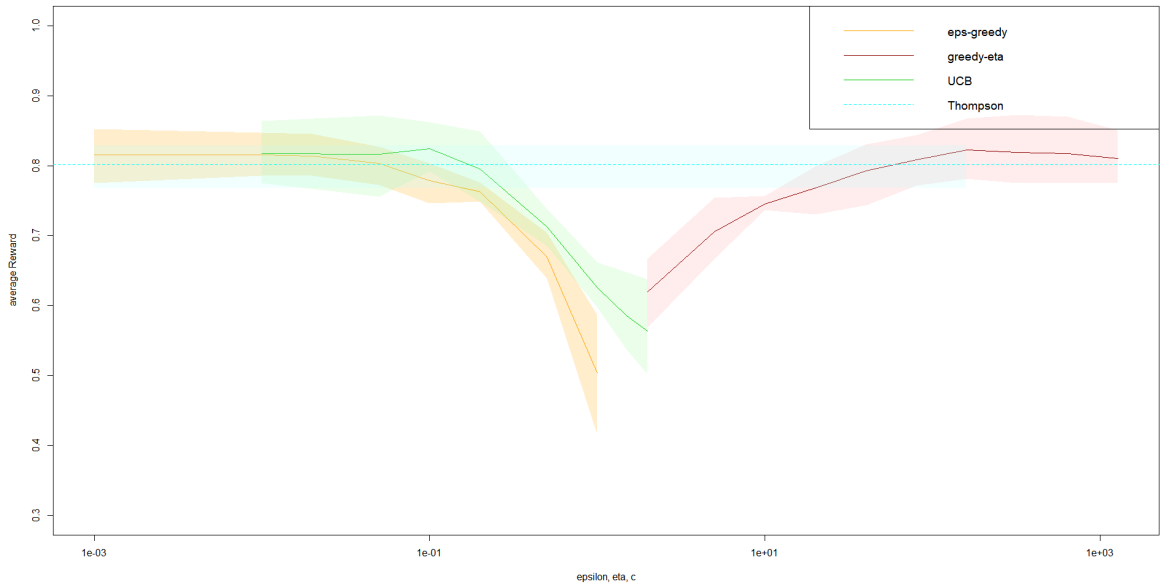

Figure 8: The average reward for horizon time $T = 1000$. **Informative** prior macro experiment ran for 500 simulations over each combination of parameters.

due to the deterministic nature of these algorithms as they have greater ability to exploit under informative prior information. Perhaps in this case, OBS would have been a better choice than standard Thompson sampling as it tends to exploit more.

9 Comparison

Most of the different approaches to solving the multi-armed bandit problem presented in this report have some trade-off to improving the regret, whether that be an extra parameter to tune, extra knowledge such as knowing a finite horizon time or prior information, or increased risk. In this section, the methods will compared based on the simulation in this report, other simulations within the literature, and theoretical guarantees.

Thompson sampling, in its simplest form, does not have any parameter to tune. This property, and the random nature of Thompson sampling makes it very robust. For example to things like delayed feedback. In fact, even when the feedback was delayed only by 1 iteration, it has been shown [27] that the difference in regret before and after introducing delayed feedback of standard UCB is 2.65 times that of Thompson sampling.
The optimistic Bayesian sampling (OBS) algorithm is shown to perform only marginally better than the standard Thompson sampling asymptotically [27] when applied to the Yahoo! data set to maximise the number of clicks for online personalised news article recommendations based on one single trial over the whole data set. In contrast, OBS has been applied by [16] to the same data set, but they investigated the short term performance by considering the variance of the regret over multiple simulations and a short time horizon. In this setting, they show a decreased regret when using OBS over Thompson sampling, indicating a potential short term benefit.
The introduction of the α parameter in the reshaped posterior Thompson sampler algorithm has large impacts on the regret. Figure 9 (left) shows that $\alpha < 1$(more exploitation) gave better expectations of the regret, but the cost is a larger variance. We can see that for $\alpha = 2$ for example, the probability of a very large regret is much smaller and is therefore less risky.
There has recently been a lot of work into the theoretical properties of Thompson sampling in the literature. Granmo (2008) [14] considers the two-armed Bernoulli bandit case and shows the probability of playing the best arm converges to 1. Agrawal and Goyal [**?**] (2011) considers the k-armed Bernoulli bandit and proves an optimal rate of regret asymptotically. Since then however, Agrawal

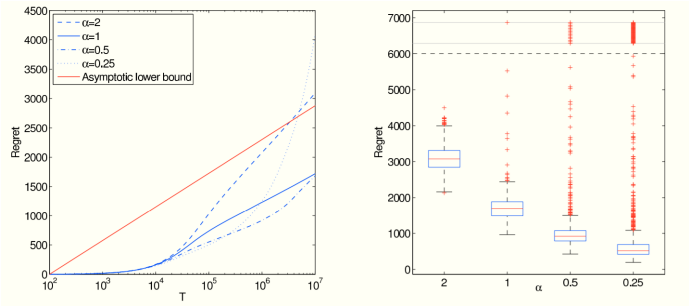

Figure 9: The parameters of the beta distribution for the posterior of the Bernoulli bandit are divided by α. Left: average regret. Right: distribution of regret over 1000 simulations.

and Goyal (2012,2014) and Russo and Van Roy (2014, 2016 [25]) have proven bounds for the finite time regret of Thompson sampling by forming a link between finite time regret in UCB algorithms, allowing them to "establish a number of new results on posterior sampling by leveraging prior work on UCB algorithms."
It is commonly announced in the literature that Thompson out performs UCB, but this is not always the case. When the UCB is tuned it can outperform Thompson, however in real applications it is difficult to tune without prior knowledge. Olivier Chapelle and Lihong Li [27] applied Thompson sampling to simulated and real data, and showed that it achieves "state-of-the-art results, and in some cases significantly outperforms other alternatives like UCB."

Upper-Confidence-Bound approaches and their variants can provide detailed theoretical properties on the bounds of its performance. Figure 10 details the performance of each of these over 1000 simulations. Compared to posterior sampling, UCB algorithms can be more difficult to implement but they benefit from having strong bounds on the expected regret.
Thompson sampling and UCB methods are both myopic algorithms, meaning they aim to reduce uncertainty in the currently played arm without explicitly considering future rewards. This means that in most cases they are easier to implement than a belief-lookahead method.
[20] introduced an ϵ_t-greedy which lets ϵ go to zero with rate $1/t$. In their simulation, they show that an optimally tuned ϵ_t-greedy algorithm "performs almost always best". With exceptions being when there are several non-optimal arms whose reward expectations differ a lot. This is because it has an *undirected* exploration approach and chooses the sub-optimal arms randomly, meaning there is potential to accumulate large regret when exploring. In most cases UCB-Tuned has similar regret to a well tuned ϵ_t-greedy, but is not as sensitive to the variability of the reward distributions of the arms because it uses myopic index-based exploration. UCB2 is always slightly worse than UCB-Tuned.
Introducing parameters into algorithms can help to improve performance, but it is preferable for such parameters to be insensitive since in most applications the optimum parameters are not known prior to testing. UCB2 is insensitive to its parameter given it is small (≈ 0.001). The ϵ-greedy algorithm on the other hand is very sensitive to its parameter and there is no single value that works well for all possible reward distributions. The choice of prior also effects the optimum value of ϵ. When no knowledge of prior or reward distribution is available, choosing a value of ϵ is very difficult.

Figure 10 shows that a lower expected value of regret, for example OBS and UCB-Tuned, coincided with a larger spread and higher probability of a very large regret. This was generally the case for all other algorithms presented in the literature. Thus, if the application desires risk-averse solutions then MOSS and UCB-V provide suitable solutions. One exception is KL-UCB, which has the lowest expected regret but also has a better spread of regret than OBS and UCB-T. OBS always performed better than Thompson sampling. It also shows a small short term boost in performance over Thompson sampling, and in applications such as web-based click data, a small increase in performance can have a large impact[34].

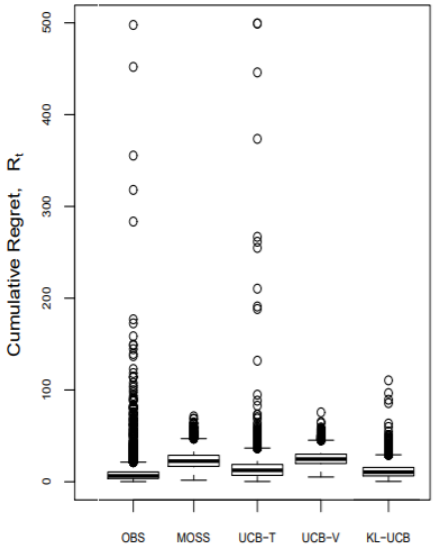

Figure 10: Comparison of the performance of Optimistic Bayesian Sampling, MOSS, UCB-Tuned, UCB-V and KL-UCB. From a simulation produced by[8]

9.1 Delayed Feedback

In real applications, the information about the rewards is often received in chunks over a certain epoch. For example in web-based learning problems, we may be interested in whether or not an advert has been clicked, and whilst the server is relaying this information we still need to provide adverts to other people. The delay in feedback τ is not necessarily constant, and so it is possible to receive feedback from actions in a different order to which they were played.
The benefit of Thompson sampling is its stochasticity makes it more robust to delayed feedback than deterministic algorithms [27], although recently some UCB methods have been adapted to optimise for delayed feedback.
Joulani and György (2013) worked on a UCB-type queue based method, Queued Partial Monitoring with Delayed Feedback (QPM-D). They have shown that for adversarial bandit problems, delay increases the regret in a multiplicative way, and in the stochastic problems considered in this report the delay increases the regret in only an additive way, such that

$$\mathcal{R}(T) = O(\sqrt{KTlogT} + K\mathbb{E}[\tau])$$

Mandel et al (2015) continues this idea and introduces an assumption of bounded delays to develop the Stochastic Delayed Bandit (SDB) algorithm. Both algorithms work by placing rewards for each arm into separate queues and receiving it once a base algorithm plays that arm.
Recently, Pike-Burke et al. (2018) have considered the Bandits with Delayed, Aggregated Anonymous Feedback (MABDAAF), where the reward is received as a sum of the outcomes of the actions played during each delay epoch. Their algorithm uses knowledge of $\mathbb{E}[\tau]$ and assumptions about the bound on τ, unlike Joulani et al. (2013). They prove that if the expected delay epoch is known there is "no extra price to be paid" [33] for only receiving aggregated anonymous feedback.

10 Concluding Remarks

There are many solutions to the multi-armed bandit problem proposed in the literature, many of which claim to be optimal. Whilst exploring the abundance of approaches to solving this problem, I have found the best solutions depends on the specifics of the problem, and must be tailored to get the best results. The algorithm chosen will depend on knowledge of prior information, the horizon time, the desired risk, and its ease of implementation.
As these problems are solved others are created, such as introducing delayed feedback or non-stationary

rewards. Solutions to these further milestones will help to deal with the complexities of real online data.

References

[1] Villar, S., Bowden, J., Wason, J. (2015) *Multi-armed Bandit Models for the Optimal Design of Clinical Trials: Benefits and Challenges.* Statist. Sci. 30 , no. 2, 199–215.

[2] Pavlov, I. P. (1927). *Conditioned reflexes: an investigation of the physiological activity of the cerebral cortex.* Oxford, England: Oxford Univ. Press.

[3] James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). *An introduction to statistical learning.*

[4] Yixin Chen, Wang, J. and Krovetz, R. (2005). *CLUE: cluster-based retrieval of images by unsupervised learning.* IEEE Transactions on Image Processing, 14(8), pp.1187-1201.

[5] Bennett, J., Lanning, S., (2007). *The Netflix Prize.* Proceedings of KDD Cup and Workshop 2007.

[6] Gittins, J. C., (1979). *Bandit Processes and Dynamic Allocation Indices*, Journal of the Royal Statistical Society, Series B, Vol. 41, No. 2. pp. 148177

[7] Whittle, P. (1988). *Restless bandits: Activity allocation in a changing world.* Journal of applied probability, pages 287298

[8] May, B. C., Leslie, D. S., (2011) *Simulation studies in optimistic Bayesian sampling in contextual-bandit problems.* Technical Report 11:02, Statistics Group, Department of Mathematics, University of Bristol.

[9] Brezzi, M., Lai, T. L. (2000) *Incomplete learning from endogenous data in dynamic allocation.* Econometrica ; 68(6):15111516.

[10] Scott, S., *A modern Bayesian look at the multi-armed bandit.* Applied Stochastic Models in Business and Industry, 26:639658, 2010.

[11] Powell, WB,. (2007) *Approximate Dynamic Programming: Solving the Curses of Dimensionality.* Wiley: New York.

[12] Brezzi, M., Lai, T. L. (2002) *Optimal learning and experimentation in bandit problems.* Journal of Economic Dynamics and Control; 27:87108.

[13] Dirkx, R., (2018) *Optimizing Infill Drilling Decisions Using Multi-Armed Bandits: Application in a Long-Term, Multi-Element Stockpile.* Mathematical Geosciences, Volume 50, Issue 1, pp 3552

[14] Granmo, O. C., (2010) *Solving two-armed bernoulli bandit problems using a bayesian learning automaton.* International Journal of Intelligent Computing and Cybernetics, 3(2):207234.

[15] Thompson, W. R., (1933) *On the likelihood that one unknown probability exceeds another in view of the evidence of two samples.* Biometrika, 25(34):285294.

[16] May, B. C., Korda, N., Lee, A., Leslie, D. S.,(2011) *Optimistic Bayesian sampling in contextual-bandit problems.* Technical Report 11:01, Statistics Group, Department of Mathematics, University of Bristol. Submitted to the Annals of Applied Probability.

[17] Garivier, A., Capp, O. (2011) *The KL-UCB Algorithm for Bounded Stochastic Bandits and Beyond* Conference On Learning Theory 24 Jul. 2011 pp.359-376

[18] Phillips, J. M. (2012) *Chernoff-Hoeffding Inequality and Applications* CoRR, abs/1209.6396.

[19] Auer, P., Ortner, R. (2010) *UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem.* Periodica Mathematica Hungarica 61, 1 (2010), 5565.

[20] Auer, P., Cesa-Bianchi, N., Fischer, P. (2002) *Finite time analysis of the multiarmed bandit problem.* Machine Learning (2002) 47: 235. https://doi.org/10.1023/A:1013689704352

[21] Audibert, J. Y., Munos, R., Szepesvri, C. (2007) *Tuning Bandit Algorithms in Stochastic Environments.* In: Hutter M., Servedio R.A., Takimoto E. (eds) Algorithmic Learning Theory. ALT 2007. Lecture Notes in Computer Science, vol 4754. Springer, Berlin, Heidelberg

[22] Audibert, J. Y., Bubeck, S. (2010). *Regret Bounds and Minimax Policies under Partial Monitoring.* Journal of Machine Learning Research. 11. 2785-2836.

[23] Auer, P., Cesa-Bianchi, N., Shawe-Taylor, J. (2006) *Exploration versus exploitation challenge.* In 2nd PASCAL Challenges Workshop. Pascal Network, 2006.

[24] Russo, D., Van Roy, B. (2013) *Learning to optimize via posterior sampling.* CoRR, abs/1301.2609, 2013.

[25] Russo, D., Van Roy, B. (2016) *An Information-Theoretic Analysis of Thompson Sampling.* Journal of Machine Learning Research, Vol. 17, pp. 1-30, 2016.

[26] Burnetas, A.N., Katehakis, M.N. (1996) *Optimal adaptive policies for sequential allocation problems.* Advances in Applied Mathematics, 17(2):122142, 1996.

[27] Li, L., Chapelle, O. (2011) *An Empirical Evaluation of Thompson Sampling* Neural Information Processing Systems, 2249-2257.

[28] Capp, O., Garivier, A., Maillard, O., Munos, R., Stoltz, G. (2012) *Kullback Leibler Upper Confidence Bounds for Optimal Sequential Allocation.* Annals of Statistics, Institute of Mathematical Statistics, 2013, 41 (3), pp.1516-1541.

[29] Gao, W., Zhou, Z. (2013) *On the doubt about margin explanation of boosting*, Artificial Intelligence, vol 203, 1 - 18

[30] Joulani, P., Gyorgy, A., and Szepesvari, C. (2013). *Online learning under delayed feedback.* In Proceedings of The 30th International Conference on Machine Learning, 14531461.

[31] Travis Mandel, Yun-En Liu, Emma Brunskill, Zoran Popovic. (2015) *The Queue Method: Handling Delay, Heuristics, Prior Data, and Evaluation in Bandits* Association for the Advancement of Artificial Intelligence.

[32] T.L. Lai and H. Robbins. (1985) *Asymptotically efficient adaptive allocation rules.* Advances in applied mathematics, 6:422.

[33] Ciara Pike-Burke, Shipra Agrawal, Csaba Szepesvari, Steffen Grunewalder. (2018)*Bandits with Delayed, Aggregated Anonymous Feedback.* eprint arXiv:1709.06853

[34] Thore Graepel, Joaquin Quinonero Candela, Thomas Borchert, Ralf Herbrich. (2010) *Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsofts Bing search engine.* In Proceedings of the Twenty-Seventh International Conference on Machine Learning (ICML-10), pages 1320, 2010.

[35] Kendall E. Atkinson, (1989) *An Introduction to Numerical Analysis*, John Wiley & Sons, Inc, ISBN 0-471-62489-6