RT1 Report: The Multi-Armed Bandit Problem and Thompson Sampling

James Neill
Supervisor: James Grant

February 2023

Overview

Consider a business deciding which advert should go on the homepage of their website each day. Some adverts will, on average, make more money than others – but the amount of money an advert will make will be different from day to day (as usage of the website fluctuates). With no prior knowledge of the quality of the adverts, how does the business decide which adverts to run?

This scenario is an example of the multi-armed bandit problem: we must continually take actions from a given set of options, making decisions based on the rewards of each action taken so far. We do not have any external information about the possible rewards of each action. Our goal is to find a method for decision-making that will, on average, result in the largest reward.

Continuing the scenario above, one method could be trying each advert once, choosing the one we like best, and then displaying it every day from then on. This strategy favors 'exploiting' – favoring the option with the greatest perceived reward. But if we make the wrong decision initially, we would make the wrong decision every day from then on. Another method could be to spend a whole year trying each advert the same number of times, determining which makes the most money on average, and choosing that advert from then on. This strategy favours 'exploring' – gathering lots of information about which takeaway is really the best, at the cost of having to spend a whole year making less money. We want to find a strategy that balances both exploring and exploiting.

In this report, we will focus on two strategies for decision making in the multi-armed bandit problem. The first is a 'greedy' algorithm, which works like the above examples. We have a fixed period of exploring, and then spend the rest of our time exploiting. The second algorithm uses a method called Thompson sampling, which continually balances exploration and exploitation by using the information gathered so far to build a clear image of the possible rewards (and how likely these possible rewards are) from each action. We will show that the Thompson sampling algorithm will lead to a greater average reward than the greedy algorithm, and investigate different methods for approximating the Thompson sampling algorithm when it is not possible to apply the method directly.

1 Multi-Armed Bandits

The multi-armed bandit (MAB) problem is a problem of choosing between different actions (called arms), each with a stochastic random reward from an unknown distribution. Our goal is to formulate a method of decision-making that will, on average, maximise the total reward. This method will split our time between 'exploring' - collecting information on which action will on average provide the largest reward - and 'exploiting' - taking the action that we have determined is best.

We can formulate this problem mathematically as follows: across T rounds, in each round we must choose one of K actions, where T is significantly larger than K. Choosing action $k \in \{1, \dots, K\}$ in round $t \in \{1, \dots, T\}$ gives reward $X_{k,t} \sim \nu_k$, where ν_k is a probability distribution with unknown parameters. The distribution ν_k does not depend on the current round t, so rewards are i.i.d. over time for each possible action.

Ideally, to ensure the greatest total reward, we would take the action with the greatest expected reward every round. In this case the expected reward each round is $\max_k\{\mathbb{E}(\nu_k)\}$. However, since we do not know the probability distributions ν_k, we will inevitably take some other set of actions $a_1, \dots, a_T$, where each $a_i \in \{1, \dots, K\}$. The expected reward in each round t is instead $\mathbb{E}(\nu_{a_t})$.

We define *regret* ρ as the difference between the expected total reward of the ideal case and the expected total reward of the actual case:

$$\begin{aligned}\rho(T) &= \sum_{t=1}^{T} \Big(\max_k\{\mathbb{E}(\nu_k)\} - \mathbb{E}(\nu_{a_t}) \Big), \\ &= T \cdot \max_k\{\mathbb{E}(\nu_k)\} - \sum_{t=1}^{T} \mathbb{E}(\nu_{a_t}).\end{aligned}$$

Our goal when choosing actions is to minimise this regret (i.e. to maximise our reward). In order to do this we will follow an algorithm to dictate which action to choose in each round; several algorithms are presented in the next section.

2 Algorithms

2.1 ε-Greedy Algorithm

The first algorithm we investigate is a 'greedy' algorithm – called greedy because after an initial exploration phase (where we select each arm an equal amount for εT rounds, given some ε, and calculate the average reward for each arm), we exploit the algorithm by choosing the arm with the greatest average reward. This algorithm is sometimes called ε-first or explore-then-commit.

The algorithm is as follows (where $x_{1:t}^{(k)}$ is the subset of $x_1, \dots, x_t$ where arm k is chosen, and $s_{k,t}$ is the length of $x_{1:t}^{(k)}$):

Algorithm 1 ε-Greedy

Require: $\varepsilon \in (0, 1)$

for $t \in \{1, \ldots, \lfloor \varepsilon T \rfloor\}$ **do**
  Let $k = t \bmod K$.
  If $k = 0$, instead let $k = K$. ▷ Cycling through $1, \ldots, K$
  Let $x_t = X_{k,t}$.
end for
for $k \in \{1, \ldots, K\}$ **do**
  Let $y_k = \sum x^{(k)}_{1:\lfloor \varepsilon T \rfloor} / s_{k,\lfloor \varepsilon T \rfloor}$. ▷ Finding the average reward
end for
Let $k' = \text{argmax}_k\{y_k\}$.
for $t \in \{\lfloor \varepsilon T \rfloor + 1, \ldots, T\}$ **do**
  Let $x_t = X_{k',t}$. ▷ Only choosing the 'best' arm
end for

We will now show that this algorithm takes $O(n)$ time (adapted from Chapter 6 of Lattimore and Szepesvári (2020)). For each $k \in \{1, \ldots, K\}$, let $\Delta_k = \max_l\{\mathbb{E}(\nu_l)\} - \mathbb{E}(\nu_k)$. Then we can write regret as

$$\begin{aligned}
\rho(T) &= \sum_{t=1}^{T} \left(\max_k \{\mathbb{E}(\nu_k)\} - \mathbb{E}(\nu_{a_t}) \right) \\
&= \sum_{t=1}^{T} \mathbb{E}\left(\sum_{k=1}^{K} \Delta_k \mathbb{I}(k = a_t) \right) \\
&= \sum_{k=1}^{K} \Delta_k \mathbb{E}\left(\sum_{t=1}^{T} \mathbb{I}(k = a_t) \right).
\end{aligned}$$

Then when we use the ε-Greedy algorithm, we have

$$\begin{aligned}
\rho(T) &= \sum_{k=1}^{K} \Delta_k \left(\mathbb{E}\left(\sum_{t=1}^{\lfloor \varepsilon T \rfloor} \mathbb{I}(k = a_t) \right) + \mathbb{E}\left(\sum_{t=\lfloor \varepsilon T \rfloor + 1}^{T} \mathbb{I}(k = a_t) \right) \right) \\
&= \sum_{k=1}^{K} \Delta_k \left(\left\lfloor \frac{\varepsilon T}{K} \right\rfloor + \mathbb{I}(k < \varepsilon T / K) + (T - \lfloor \varepsilon T \rfloor) \mathbb{P}(k = k') \right).
\end{aligned}$$

Since $\mathbb{P}(k = k')$ is non-zero for all k, we see that the regret is $O(T)$.

2.2 Thompson Sampling Algorithm

The second algorithm we investigate is Thompson Sampling (TS). This algorithm was originally developed by William Thompson in Thompson (1933). In this algorithm, we calculate the posterior distribution for the parameters of the distributions of each arm (given some prior distribution, chosen based on the particular problem). Each round, we take one sample from each arm's distribution. Whichever arm that provided the sample that gives the largest expected reward is selected, and the posterior distribution of this arm is updated with the new datapoint.

The algorithm is as follows (from Chapter 4 of Russo et al. (2018)):

Algorithm 2 Thompson Sampling

for $t \in \{1, \dots, T\}$ **do**
  for $k \in \{1, \dots, K\}$ **do**
    Sample $\tilde{\theta}_k$ from $p(\theta_k | x_{1:t-1}^{(k)})$. ▷ Sampling from the posterior
    Let $y_k = \mathbb{E}(X_{k,t}; \tilde{\theta}_k)$.
  end for
  Let $k' = \text{argmax}_k\{y_k\}$.
  Let $x_t = X_{k',t}$.
  Update the posterior distribution of $\theta_{k'}$ with x_t.
end for

This is a better algorithm that the greedy algorithm, because it considers the whole posterior distribution of the parameters, rather than just the average – balancing both exploration and exploitation over time. From Agrawal and Goyal (2012) we see that the regret for the Thompson Sampling algorithm grows $O(\log T)$, significantly better than the $O(T)$ of the greedy algorithm.

2.3 Other Algorithms

There are various other algorithms that can be used to make decisions in the multi-armed bandit problem. For example, the UCB1 (upper confidence bound) algorithm balances exploration and exploitation by taking into account both the average reward and the number of visits so far to each arm. From Auer et al. (2002) we see that regret for the UCB1 algorithm also grows $O(\log T)$.

2.4 Bernoulli Example

A simple example of the multi-armed bandit problem is Bernoulli bandits: the reward from each of the K arms follows a Bernoulli distribution with mean θ_k (the arms being i.i.d. of each other).

If each θ_k has a Beta prior distribution with parameters α_k, β_k, then the posterior distribution for θ_k after t rounds is

$$\begin{aligned} p(\theta_k | x_{1:t}^{(k)}) &\propto p(\theta_k) L(x_{1:t}^{(k)} | \theta_k) \\ &\propto \theta_k^{\alpha-1}(1-\theta_k)^{\beta-1}\theta_k^{\sum x_{1:t}^{(k)}}(1-\theta_k)^{s_{k,t}-\sum x_{1:t}^{(k)}} \\ &= \theta_k^{\alpha+\sum x_{1:t}^{(k)}-1}(1-\theta_k)^{\beta+s_{k,t}-\sum x_{1:t}^{(k)}-1}. \end{aligned}$$

This means the posterior distribution is

$$\text{Beta}\Big(\alpha + \sum x_{1:t}^{(k)}, \beta + s_{k,t} - \sum x_{1:t}^{(k)}\Big),$$

which means that when updating the posterior distribution in the Thompson Sampling algorithm, we can easily update the posterior without having to recalculate the parameters from

scratch (we add 1 to the first parameter when we have a reward of 1, and add 1 to the second parameter when we have a reward of 0).

In Section 4, we will investigate this problem empirically, showing that the ε-Greedy algorithm is $O(T)$ and that the Thompson Sampling algorithm is $O(\log T)$. This example has been explored further empirically in Chapelle and Li (2011).

3 Thompson Sampling Approximations

3.1 Sampling Difficulties

In the Thompson Sampling algorithm we sample from the posterior distribution for the parameters, given the data so far. However, with some complicated distributions, sampling from the posterior distribution may not be possible. In this section we will investigate different methods for Thompson Sampling in this case (as presented in Chapter 5 of Russo et al. (2018)). We will specifically be using the Bernoulli example above – this means we can empirically compare the approximations with how Thompson Sampling actually performs in Section 4.

3.2 Bootstrap Approximation

The first approximation algorithm that we investigate uses bootstrapping to replace the posterior distribution in Thompson Sampling, based on the first algorithm in Eckles and Kaptein (2014). For each arm k, we create J replicates of the parameters of the posterior (in this case, the Beta distribution has parameters α and β). At each point in time we randomly choose one replicate for each arm and choose the arm with the greatest posterior estimate for the mean; we then update all the replicates for the chosen arm each with probability 1/2.

Algorithm 3 TS Bootstrap Approximation

Let $\alpha_{k,j} = 1$ and $\beta_{k,j} = 1$ for all $k \in \{1, \dots, K\}$, $j \in \{1, \dots, J\}$.
for $t \in \{1, \dots, T\}$ **do**
    for $k \in \{1, \dots, K\}$ **do**
        Sample j_k uniformly from $\{1, \dots, J\}$. ▷ Randomly choosing one replicate
        Let $y_k = \alpha_{k,j_k}/(\alpha_{k,j_k} + \beta_{k,j_k})$. ▷ Estimating the mean of the posterior
    end for
    Let $k' = \text{argmax}_k\{y_k\}$.
    Let $x_t = X_{k',t}$.
    for $j \in \{1, \dots, J\}$ **do**
        Sample i uniformly from $\{0, 1\}$.
        if $i = 1$ **then** ▷ Updating the replicates with probability 1/2
            Let $\alpha_{k',j} = \alpha_{k',j} + x_t$.
            Let $\beta_{k',j} = \beta_{k',j} + 1 - x_t$.
        end if
    end for
end for

3.3 Laplace Approximation

The second approximation method that we consider is the Laplace approximation: we approximate the complicated posterior using a normal distribution (which is clearly easy to sample from).

If we are trying to sample from $p(\theta|x)$, then a second-order Taylor expansion for $\log p(\theta|x)$ around its mode θ_0 is

$$\log p(\theta|x) \approx \log p(\theta_0|x) - \frac{v}{2}(\theta - \theta_0)^2 + O(\theta^4),$$

where $v = -\frac{\mathrm{d}^2}{\mathrm{d}\theta^2} \log p(\theta|x)|_{\theta=\theta_0}$. This means we have

$$p(\theta|x) \propto \exp\left(-\frac{v}{2}(\theta - \theta_0)^2\right),$$

so we can approximate $p(\theta|x)$ by a Normal distribution with mean θ_0 and variance v^{-1} (see Chapter 27 of MacKay (2005)).

In the Bernoulli bandit case, we are estimating the Beta distribution, so for each arm k we have

$$\theta_{k,0} = \frac{\alpha - 1}{\alpha + \beta - 2}, \qquad v_k = \frac{\alpha_k - 1}{\theta_{k,0}^2} + \frac{\beta_k - 1}{(1 - \theta_{k,0})^2}.$$

Note that when $\alpha_k = \beta_k = 1$, we have instead that $\theta_{0,k} = 1/2$ and $v_k = 0$. We then let the variance of the Normal distribution equal 1 (since v_k^{-1} is undefined).

We can now follow the same algorithm as standard Thompson Sampling, except sampling each $\tilde{\theta}_k$ from a $\mathrm{N}(\theta_{0,k}, v_k^{-1})$ distribution instead of $p(\theta_k|x_{1:t-1}^{(k)})$ – the same as Algorithm 2, changing only line 3.

3.4 Other Approximations

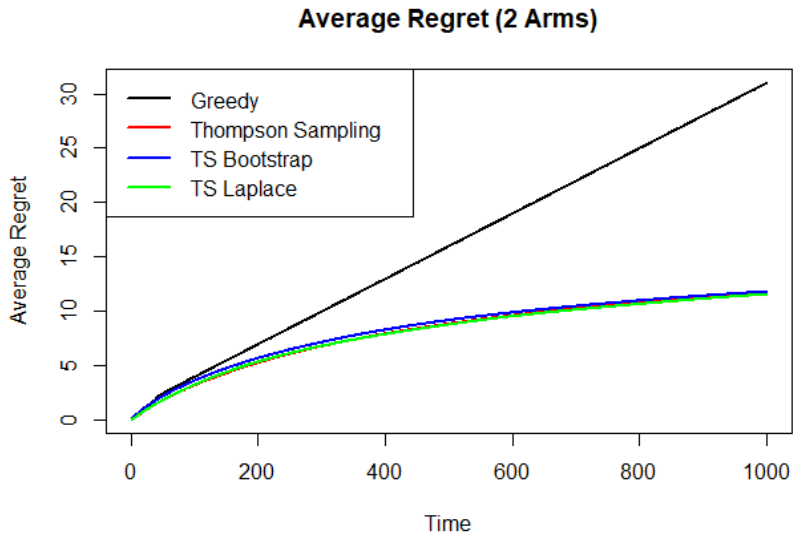
There are various other methods that can be used to approximate Thompson sampling: in Mazumdar et al. (2020), Langevin Markov chain Monte Carlo is used to approximate the posterior distribution the Thompson sampling algorithm, and in Riquelme et al. (2018), Thompson sampling is approximated using Bayesian neural networks.

4 Empirical Comparison

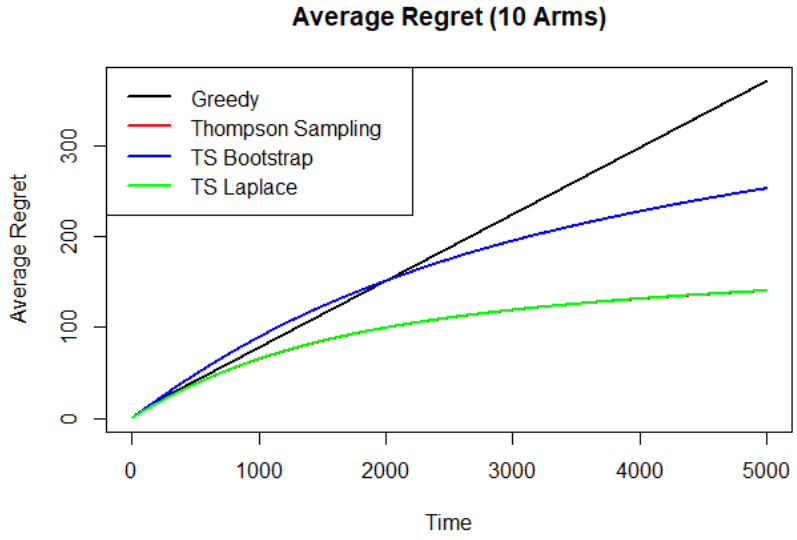
We now compare the growth of regret over time between the ε-Greedy algorithm, Thompson Sampling algorithm, and the two TS approximation algorithms on the Bernoulli bandit problem. Each algorithm is averaged over 1000 macroreplications. For the Greedy algorithm we use $\varepsilon = 0.05$, and for the TS bootstrap approximation algorithm we use 100 replicates per arm. In all cases we assume no initial knowledge, and so use a flat $\mathrm{Beta}(1, 1)$ prior.

We first consider a case with two arms: one with mean 0.5, and the other with mean 0.6. We run each algorithm for 1000 time-points. The plot below show the total regret over time – we see that the greedy and Thompson sampling algorithms behave as we expect; the former growing

$O(T)$ and the latter growing $O(\log T)$. We also see that both TS approximation algorithms perform very well (both have approximately the same regret as standard Thompson sampling).



The other case we consider has ten arms: nine with mean 0.5, and one with mean 0.6. Since we have more arms, it will take longer to differentiate between them, so we run the algorithms for 5000 time-points. Again we see the regret for the greedy and Thompson sampling algorithms grow as we expect and that the Laplace approximation to Thompson sampling performs very well. However, we also see that in this case the bootstrap approximation to Thompson sampling performs significantly worse than standard Thompson sampling (while still growing $O(\log T)$). In Eckles and Kaptein (2014) we see that the regret of the bootstrap approximation approaches the regret of standard Thompson sampling as the number of bootstrap replications increases.



5 Conclusion

In this report, we have explored several methods for decision-making in the multi-armed bandit problem, including the ε-greedy method and through Thompson sampling. By measuring regret, the difference between the expected total reward of the theoretically best course action and the expected total reward of the actions we take, we found that Thompson sampling performs significantly better than the ε-greedy method (the regret grows slower over time). We have also explored several methods for approximating Thompson sampling when we cannot apply the method directly, including using bootstrapping and using a Laplace approximation. Through an empirical comparison we see that the Laplace approximation incurs less regret over time than the bootstrap approximation in the Bernoulli bandit example.

References

Agrawal, S. and Goyal, N. (2012). Analysis of Thompson sampling for the multi-armed bandit problem. *Conference on Learning Theory*, 23:39.1–39.26.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256.

Chapelle, O. and Li, L. (2011). An empirical evaluation of Thompson sampling. *Advances in Neural Information Processing Systems*, 24.

Eckles, D. and Kaptein, M. (2014). Thompson sampling with the online bootstrap. *arXiv preprint arXiv:1410.4009*.

Lattimore, T. and Szepesvári, C. (2020). *Bandit algorithms*. Cambridge University Press.

MacKay, D. J. (2005). *Information theory, inference and learning algorithms*. Cambridge University Press.

Mazumdar, E., Pacchiano, A., Ma, Y., Jordan, M., and Bartlett, P. (2020). On approximate Thompson sampling with Langevin algorithms. *International Conference on Machine Learning*, 1:6797–6807.

Riquelme, C., Tucker, G., and Snoek, J. (2018). Deep Bayesian bandits showdown: An empirical comparison of Bayesian deep networks for Thompson sampling. *arXiv preprint arXiv:1802.09127*.

Russo, D. J., Van Roy, B., Kazerouni, A., Osband, I., Wen, Z., et al. (2018). A tutorial on Thompson sampling. *Foundations and Trends in Machine Learning*, 11(1):1–96.

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