Statistical Learning for Decision - Thompson Sampling

STOR 608: Individual Report

Libby Daniells

1 Introduction

Multi-armed bandits provide a framework for the modelling of many real-world sequential decision problems. In such problems, an agent learns information regarding some uncertain factor as time progresses and uses this to make more informed choices. A common application of this is in slot machines, where a player must select one of K arms to pull, each of which outputs a random reward.

A K-armed bandit is set up as follows: the player has K arms to choose from (known as actions), each of which yields a reward payout defined by random variables, R_i, which output realisations r_i for arm i. These rewards are independently and identically distributed (IID) with some unknown mean μ_i that the player must learn through experimentation. In order to experiment, the player requires a policy regarding which action to take next. Formally, a policy is an algorithm that chooses the next arm to play using information on previous plays and the rewards they obtained (Auer et al., 2002).

When making the decision on which arm to pull next, the policy must weigh up the benefits of exploration versus exploitation. Exploitation refers to choosing an arm that is known to yield a desirable reward in short-term play. Whereas, exploration is the search for higher payouts in different arms, this can be used to optimize long term reward.

The focus of this report will be on Thompson Sampling (Thompson, 1933), which is a method that balances this exploitation vs. exploration in an efficient manner. The policy works by selecting an arm based on the Bayesian probability that it's expected reward payout is best.

In order to quantify a policy's success we use a measure known as regret. The regret of a policy after T rounds is defined as

$$\mathcal{R}(T) \ = \ \mu^* T - \sum_{j=1}^{K} \mu_j \mathbb{E}[T_j(T)] \quad \text{where } \mu^* = \max_{1 \leq i \leq K} \mu_i \tag{1.1}$$

and $T_j(T)$ is the number of times arm j has been played during the T rounds. Hence, regret is the loss due to the fact that the policy does not always choose the optimal arm (Auer et al., 2002).

This report will begin in Section 2 with a review of Thompson sampling as a method to tackle multi-armed bandit problems. In Section 2.1 we then explore it's applications to contextual based decision problems. We will conclude in section 3 with a simulation study to see how the method behaves in practice, whilst also exploring the effect of prior specification.

2 Thompson Sampling

We begin by defining the Thompson Sampling (TS) algorithm for the Bernoulli-bandit case in which rewards, r_k, take a value of either 0 (failure) or 1 (success). We have a set-up of $k \in \{1, \dots, K\}$ arms each with a probability of successful reward, θ_k. We would like to maximise the number of successes over the time period T.

We begin with a prior belief on our success probabilities θ_k which we take to be Beta distributed with density

$$p(\theta_k) \; = \; \frac{\Gamma(\alpha_k + \beta_k)}{\Gamma(\alpha_k)\Gamma(\beta_k)} \, \theta_k^{\alpha_k - 1} \, (1 - \alpha_k)^{\beta_k - 1}.$$

Each time we gain an observation we update this prior belief and thus use the posterior distribution as our prior for the succeeding arm pull. The Beta distribution is conjugate for the Bernoull, therefore the posterior will also be Beta distributed with updated parameters $(\alpha_{x_t}, \beta_{x_t}) \leftarrow (\alpha_{x_t} + r_t, \beta_{x_t} + 1 - r_t)$, where x_t is the action carried out at time t. For all arms not pulled at time t, we do not update the parameters as no new information regarding the θ_k's has been observed.

The process of Thompson sampling for the Bernoulli-bandit case is summarised in Algorithm 2.1.

Algorithm 2.1 Bernoulli TS

```
1: for $t = 1, 2, \dots$ do
2:     for $k = 1, 2, \dots, K$ do
3:         Sample $\hat{\theta}_k \sim \text{Beta}(\alpha_k, \beta_k)$
4:     end for
5:     $x_t \leftarrow \arg\max_k \hat{\theta}_k$
6:     Apply $x_t$ and observe $r_t$.
7:     $(\alpha_{x_t}, \beta_{x_t}) \leftarrow (\alpha_{x_t} + r_t, \beta_{x_t} + 1 - r_t)$
8: end for
```

(Russo et al., 2017)

Algorithm 2.2 General TS

```
1: for $t = 1, 2, \dots$ do
2:     for $k = 1, 2, \dots, K$ do
3:         Sample $\hat{\theta}_k \sim p$
4:     end for
5:     $x_t \leftarrow \arg\max_k \mathbb{E}_{q_{\hat{\theta}_k}}[r(y_t)|x_t = k]$
6:     Apply $x_t$ and observe $r_t$.
7:     $p \leftarrow \mathbb{P}_{p,q}(\theta \in \cdot|x_t, y_t)$
8: end for
```

Although we have used the conjugate prior in this case, a prior of any distribution may be used. The downfall of this is that the posterior may follow a different distribution to the prior, so within the algorithm it will not be as simple as updating the parameters for the next arm pull, but potentially having to completely specify a new prior at each time step.

More generally TS can be applied to many distributions beyond the Bernoulli case. In general, when we apply action x_t we observe some outcome y_t which is drawn randomly from the distribution $q_\theta(\cdot|x_t)$. The reward, r_t, obtained from action x_t, is then given by $r_t = r(y_t)$ where r is a known reward function.

Working through Algorithm 2.2, we begin by sampling from some prior distribution p which reflects our beliefs of θ_k. We then select the action/arm that maximises the expected reward. From there, we apply x_t and observe some reward r_t, updating the prior to incorporate this observation.

2.1 Contextual Thompson Sampling

Consider the case of website advertisement where we are interested in click-through rate. This can be set up as a Bernoulli-bandit where rewards take a value of either 0 (failure to click on the ad) or 1 (successful click of the ad). This success rate will depend on the user viewing the advertisement. For example, some ads will appeal to a younger audience, while others to older viewers. We would therefore like a way to incorporate the user information into our bandit.

This is where contextual bandits comes in. At each time step a vector of information known as context, c_t is observed. Using this information we must select one of k actions from which a random reward (which is dependent on the context) is observed.

Pseudo-code for the algorithmic proceedure of contextual Thompson sampling is outlined in Algorithm 2.3 and is based off work by Elmachtoub et al. (2017). The main difference from the non-contextual case is that we require the prior for θ_k to be conditional on the observed context-reward pairs $\mathcal{D}_{t,k} = \{(c_s, r_{s,k}) : \ s \leq t-1, \ x_s = k\}$. We then select the action that maximise expected reward based on the sampled parameters, then observe a reward and update the context-reward pairs.

Algorithm 2.3 Thompson Sampling for Contextual Bandits

```
1: for $t = 1, 2, \ldots$ do
2:     Observe context vector $c_t$
3:     for $k = 1, 2, \ldots, K$ do
4:         Sample $\hat{\theta}_{t,k} \sim P(\theta_k | \mathcal{D}_{t,k})$
5:     end for
6:     $x_t \leftarrow \arg\max_k \mathbb{E}[r_k | \hat{\theta}_{t,k}, c_t]$
7:     Apply $x_t$ and observe $r_t$
8:     Update $\mathcal{D}_{t,x_t} = \mathcal{D}_{t,x_t} \cup (c_t, r_{t,x_t})$
9: end for
```

3 Simulation Study

3.1 Effect of Prior specification on non-contextual Thompson Sampling

Specification of a prior plays a large role in the effectiveness of the Thompson sampling algorithm and thus care must be taken over which prior to use. When specifying a prior distribution, it should be chosen to describe the set of all plausible values of our success parameter θ_k.

To test the effect of prior specification, we set up a two-armed Bernoulli-bandit with probabilities of success 0.55 and 0.5. We then applied Thompson sampling with 4 different prior's (plotted in

Figure 1) and simulated 1,000 times, calculating the mean and standard deviation of regret. The results are summarised in Table 1.

Figure 1: Prior Distributions

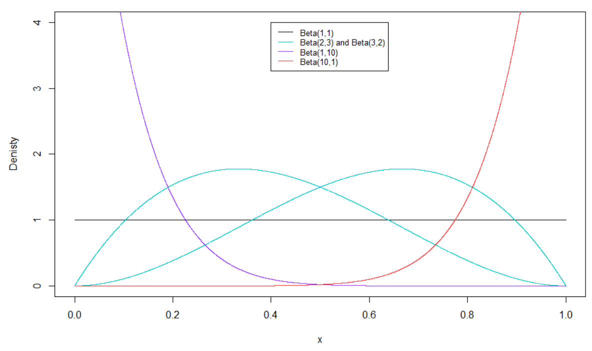


Table 1: Regret Summaries

	Mean Regret	Sd
Beta(1,1)	12.391	10.498
Beta(3,2) Beta(2,3)	10.834	10.207
Beta(1,10)	17.472	21.660
Beta(10,1)	14.080	7.662

In the slot machine case, the player often has no intuition on the values of θ_k for each arm. This lack of knowledge is usually described using an uninformative Beta(1,1) prior as it takes uniform values across the entire [0,1] support. This flat prior promotes exploration.

However, if some information regarding the θ_k's *is* known, we can centre our prior distributions around these values in order to reduce regret. In our simulation $\theta = (0.55, 0.5)$, so we can specify Beta(3,2) and Beta(2,3) priors which are centred around these values. This reduces regret as we require less exploration to identify the optimal arm and can exploit this to maximise reward.

At each time step the parameters of the prior are updated, causing the distribution to change shape and favour a certain action. Consider a case where we have specified the prior to have it's mass about 0 (e.g. Beta(1,10)). This is a misspecified prior as it does not describe the plausible values of θ appropriately. In this case, it takes longer and is more difficult to move the distributions and learn about our θ values, ultimately leading to a higher regret. Similarly, consider a prior which has most of it's mass around the value 1 (e.g. Beta(10,1)). In Table 1 we see the Beta(10,1) gives lower mean regret than the Beta(1,10) prior. This is likely because the Beta(10,1) provides a slightly better description of our θ values and so takes less time to learn which arm is optimal.

All of these cases can have outliers where mean regret is much higher than we would expect. For example, regret may be increased if we begin by pulling the sub-optimal arm and receive reward 1. We are likely to exploit this further and continue to pull this arm, falsely leading us to believe the sub-optimal arm is actually optimal. Or alternatively, we may begin by selecting the optimal arm and observe a failure reward. We then may exploit the sub-optimal arm under the false belief that it will give us a higher reward.

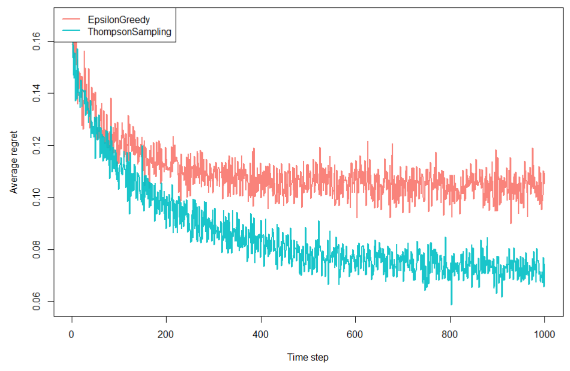
To summarise, unless we have concrete information regarding the value of θ, it is safest to utilize a Beta(1,1) uninformative prior. As seen in Table 1, the Beta(1,1) prior gives the second lowest regret, only behind the case where we are highly certain of our reward probabilities (which is unlikely to be the case in real-life applications), whereas, the mean regret is much higher if we have accidentally misspecified the prior.

3.2 Thompson Sampling vs. ϵ-Greedy for Contextual Bandit Problems

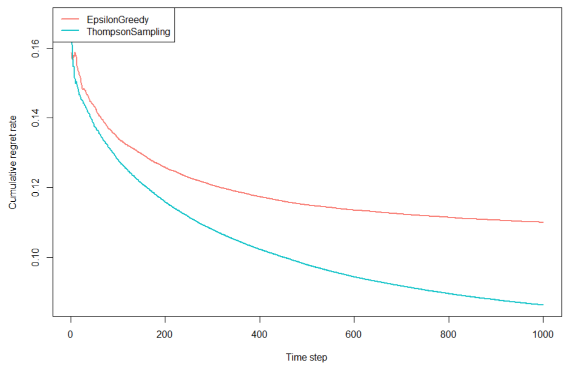
We take the website advertisement example as outlined in Section 2.1, where we have three advertisements and at each time step we receive the age of the viewer. The success rate of the advertisement will depend on the age group of the audience. We define three age groups: Under 21's, $22-40$ and Over 40's; and set probabilities of success for each arm as $\theta = (0.3, 0.5, 0.1), (0.6, 0.55, 0.4)$ and $(0.7, 0.2, 0.6)$ for each age group respectively. We simulated the age of the viewer to have equal probability of lying in each of the age categories.

In order to evaluate the performance of the Contextual Thompson sampling algorithm we compare it to the simple ϵ-greedy algorithm. In this policy we take the arm that maximises reward with probability $1 - \epsilon$ and select the arm randomly ϵ of the time. Once observations are obtained we update our estimates of success probabilities in each arm. It is known as greedy as it aims to maximise short term reward and thus tends to lack exploration.

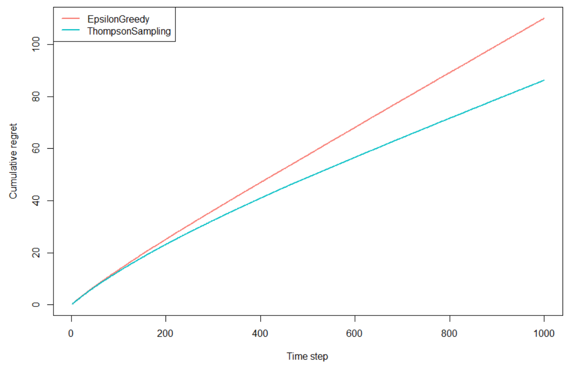
We ran the simulation using both the Thompson and ϵ-greedy methods 10,000 times, each with a 1000 length run time, setting the ϵ parameter to be 0.4 and using a Beta(1,1) prior for TS. Over each simulation we computed regret as a measure of performance.



(a) Average Regret



(b) Cumulative Regret Rate



(c) Cumulative Regret

Figure 2: Contextual Thompson Sampling Simulation

From Figure 2 we observe that Thompson sampling is far superior to the ϵ-greedy method for contextual bandits, giving on all accounts a lower regret value. Figure 2a is a visualization of how average regret changes as time progresses. The average regret starts high as the policy is exploring the environment to learn which arm gives optimal reward, creating a higher chance of selecting the sub-optimal arm. It then begins to level off as more information regarding success probabilities is obtained, leading to the exploitation of the arm which produces optimal reward, hence reducing regret. A similar trend is observed in Figure 2b, which describes the cumulative regret rate as time progresses.

Figure 2c plots cumulative regret, where we see that the gradient for the Thompson sampling method begins to decrease. If we were to run this simulation for longer, we would expect this line to level off as the user learns and begins to select the optimal action a larger proportion of the time. Over the 10,000 simulations, TS obtains a mean cumulative regret of 86.178 (sd 18.224) compared to 109.993 (sd 16.941) for the ϵ-greedy method.

The example and code used in this section are based off that work by van Emden and Kaptein (2018).

4 Conclusion

To conclude, Thompson sampling is an efficient algorithm for solving stochastic multi-armed bandit problems in both the contextual and non-contextual settings. It successfully balances the trade-off between exploration and exploitation by initially exploring during uncertainty and exploiting as more information is gathered. This minimises regret and maximises reward. However, careful considerations need to be made when selecting a prior distribution in order to avoid misspecification and the consequencing increased regret. From our simulation, we concluded that, in cases of uncertainty surrounding our parameter value θ, a Beta(1,1) prior is preferred in order to avoid misspecification.

References

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256. 1

Elmachtoub, A. N., McNellis, R., Oh, S., and Petrik, M. (2017). A practical method for solving contextual bandit problems using decision trees. *CoRR*. 3

Russo, D., Van Roy, B., Kazerouni, A., Osband, I., and Wen, Z. (2017). A tutorial on thompson sampling. 2

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. 1

van Emden, R. and Kaptein, M. (2018). contextual: Evaluating contextual multi-armed bandit problems in r. 6