Thompson Sampling and Upper Confidence Bound as solutions to the exploitation vs exploration dilemma

Nils Mattiß and Robin Sell
Machine Learning 2022 at UB
Gran Via de les Corts Catalanes, 585
nmattima25@alumnes.ub.edu, rsellsel145@alumnes.ub.edu

1 Introduction

1.1 Exploitation vs. Exploration

In the field of computer science, the trade-off between exploitation and exploration is a central concept that refers to the balancing act between learning about new options and maximizing the value of known options Audibert et al. [2009]. This trade-off is often encountered in scenarios where an agent must make choices in an uncertain or dynamically changing environment. On one hand, exploration allows the agent to gather information, learn about new options, and discover potentially valuable opportunities. On the other hand, exploitation involves focusing on known options and using the information and resources available to maximize the reward. Finding the right balance between exploration and exploitation is crucial for the success of the agent.

One well-known example of this trade-off is the multi-armed bandit (MAB) problem, which involves an agent facing a sequence of choices among several options, called ”arms”. The agent must decide which arm to play in each round. After that he receives a reward sampled from a distribution assigned to each arm with the goal of maximizing the total reward obtained over time. This requires finding a balance between exploring the different arms to gather more information and exploiting the current knowledge to maximize reward. It is particularly interesting in research areas such as online advertising, clinical trials, and recommendation systems Bouneffouf et al. [2020].

An interesting variation of this problem is the cascading bandit that involves a sequence of decisions, or ”rounds”, in which the agent must choose among several arms Kveton et al. [2015](Robin). However, unlike the traditional MAB problem, the cascading bandit problem (CBP) introduces the additional challenge of playing multiple arms at once by only getting one total reward. Specifically, bad items should ”cascade” down the ranking after being played a certain number of times - accordingly good items should ”cascade” up the rankings.

1.2 Solving Approaches

There are several ways to solve the MAB problem, each with their own benefits and limitations. Some popular methods include greedy algorithms Russo et al. [2018](Robin), which always play the arm with the highest estimated reward; epsilon-greedy algorithms Auer et al. [2002a], which mix exploration and exploitation by playing a random arm with a small probability; and exploration-exploitation algorithms, such as Thompson Sampling

(TS) Thompson [1933]Russo et al. [2018](Robin) and Upper Confidence Bound (UCB) Lai [1987]Auer et al. [2002a], which use more sophisticated methods to balance exploration and exploitation.

This paper takes a closer look at TS which works by maintaining a distribution over the possible rewards for each arm, and using these to sample a reward for each arm at each round. After interacting with the environment the distribution gets updated using Bayesian inference based on the made observations. Specifically, at each round t, the algorithm updates the posterior distribution for each arm i based on the previous observations $y_{1:t-1}$ and the corresponding arm selections $x_{1:t-1}$ up to round $t-1$, according to the following formula:

$$p(\theta_i|y_{1:t-1}, x_{1:t-1}) \propto p(y_{1:t-1}|\theta_i, x_{1:t-1}) * p(\theta_i)^1 \tag{1}$$

Here $p(\theta_i)$ is the prior distribution over the rewards θ_i, and $p(y_{1:t-1}|\theta_i, x_{1:t-1})$ is the likelihood of the observations given the reward probabilities and arm selections. Once the posterior distributions are updated, the algorithm samples a reward θ_i for each arm from it's corresponding reward distribution. Afterwards the arm with the highest sample gets played and its distribution gets updated according to the observed reward. Then the process repeats.

Unlike TS, which uses a Bayesian approach to update the distributions and compute the scores, UCB uses an optimization-based approach that leverages the concept of confidence bounds. Specifically, at each round t, the algorithm computes the score for each arm i as the upper confidence bound on the expected reward under the distribution, which can be written as:

$$\text{score}_i = \text{mean}_i + \sqrt{\frac{2 * \ln(t)}{n_i}} \tag{2}$$

Here, mean_i is the sample mean of the rewards observed for arm i up to round t, n_i is the number of times arm i has been selected up to round t, and t is the current round. The term $\sqrt{\frac{2*\ln(t)}{n_i}}$ represents the upper confidence bound on the variance of the reward distribution for arm i, and it decreases as the number of observations n_i increases Audibert et al. [2009].

2 Related Work

There are many approaches that have been proposed to solve the MAB problem, and they can be broadly classified into three categories: Bayesian approaches, optimization-based approaches, and gradient-based approaches.

Bayesian approaches, such as Thompson Sampling (TS)Thompson [1933]Russo et al. [2018](Robin), have been introduced already. Specific variants that will be focused on in this paper are Meta-Thompson Wan et al. [2022](Nils) and Neural Thompson Zhang et al. [2020](Nils).

Optimization-based approaches, such as the Upper Confidence Bound (UCB) algorithm Lai [1987]Auer et al. [2002a] and the Gittins index Gittins [1979], use optimization principles to compute scores or "values" for each arm and select the arm with the highest score at each round.

Gradient-based approaches, such as the EXP3 respectively EXP4 Auer et al. [2002b] algorithm and the Hedge algorithm Littlestone and Warmuth [1994]Cesa-Bianchi et al. [1997]Freund and Schapire [1997], use gradient descent or "follow-the-leader" techniques to update the estimates of the reward probabilities for each arm based on the observed rewards. These algorithms have been shown to be effective in certain settings, but they can be sensitive to the learning rate and other hyper-parameters.

[1] $\propto$ means that the distributions are proportional to each other

In addition to these approaches, there are also algorithms based on reinforcement learning, Bayesian optimization, and information-theoretic principles that have been proposed to solve the MAB problem. Each algorithm has its own strengths and limitations, and the choice of algorithm depends on the specific characteristics of the problem at hand.

In addition to the basic MAB problem, there are also variations of the problem that involve additional layers of complexity. One such variation is the contextual bandit problem, which involves an additional layer of contextual information or features associated with each arm. Algorithms for solving the contextual bandit problem include the LinUCB algorithm Li et al. [2010], the Contextual Thompson Sampling (CTS) algorithm Agrawal and Goyal [2013]Abeille and Lazaric [2017], the Contextual Upper Confidence Bound (CUCB) algorithm Dani et al. [2008]Li et al. [2010], and the Contextual Hedge algorithm Auer et al. [1995], among others.

Another variation is the previously mentioned cascading bandit problem Kveton et al. [2015](Robin), in which the arms are organized in a hierarchy or ”cascade”, and the agent must make a sequence of decisions at each level of the cascade in order to select an arm. Algorithms for solving this problem include the Cascading Upper Confidence Bound (CUCB) algorithm Kveton et al. [2015](Robin) and the Cascading Thompson Sampling (CTS) algorithm Russo et al. [2018](Robin). Also the previously seen algorithms like Exp3, Hedge and LinUCB can be extended to this problem by extending the action space. These are then oftentimes used as benchmark.

For further research by the reader, there are more variants that have received significant attention in recent years like the adversarial bandit problem Auer et al. [1995] - in which the rewards for each arm are chosen by an adversarial player who tries to minimize the reward of the agent - and the combinatorial bandit problem Cesa-Bianchi and Lugosi [2012] - in which the agent must select a subset of arms at each round by now having, opposite to the cascading problem, no independence between arms anymore.

3 Proposal

We propose the (generalized) cascading bandit problem, which is a variation of the MAB problem and then present and use multiple algorithms to solve it.

3.1 Cascading Bandits

The generalized cascading bandit problem - generalized means that the distribution used is not yet specified - can be represented by a tuple $B = (E, p, K)$. Here, $E = 1, \dots, L$ is a ground set of L items, p is a probability distribution over a unit hypercube $\{0,1\}^E$, and $K < L$ is the number of recommended items Kveton et al. [2015](Robin).

First we define the attraction probabilities $(w_t)_{t=1}^n$ to be an i.i.d. sequence of n weights from our distribution p with $w_t \in \{0,1\}^E$. We set $w_t(e)$ equal to 1 if and only if at time t the user is attracted by item e and set it to 0 otherwise. At time t, the learning agent computes and recommends a list of K items $A_t = (a_1^t, \dots, a_K^t) \in \Pi_K(E)$ based on the observations it has seen up until that point and where $\Pi_K(E)$ is the set of all K-permutations of set E. The user can then examine the list and select the first attractive item. If nothing seems appealing, the time increments to $t+1$ Kveton et al. [2015](Robin).

The reward of the agent at time t can be described like this:

$$f(A, w) = 1 - \prod_{k=1}^{K} (1 - w(a_k)). \tag{3}$$

The feedback received - in the specific case of the web search problem this would be the clicks of the user - can be described as

$$C_t = \arg\min\{1 \leq k \leq K : w_t(a_k^t) = 1\} \tag{4}$$

with the idea of setting $\arg\min\{\emptyset\} = \infty$ in case no item was chosen. Using the value of C_t, we can calculate the observed weights of all the recommended items at time t. It is worth noting that the weights of the items in the ground set E are independently distributed. An important assumption that we will now make is that the weights are distributed according to this:

$$p(w) = \prod_{e \in E} p_e(w(e)) \tag{5}$$

where p_e is a Bernoulli distribution with mean $\overline{w}(e)$. By making this assumption, we can now consider this problem to be a cascading bandit in which the weight at time t is independently drawn from the weights of the other items.

Now finally we get the expected reward for list $A \in \Pi_K(E)$ being $\mathbb{E}[f(A, w)] = f(A, \overline{w})$ which shows that it only depends on the attraction probabilities of individual items in A. Using the optimal list A^* we can now use the expected cumulative regret to evaluate the agent's policy. Therefore we define

$$R(n) = \mathbb{E}\left[\sum_{t=1}^{n} R(A_t, w_t)\right] \tag{6}$$

where $R(A_t, w_t) = f(A^*, w_t) - f(A_t, w_t)$ being the instantaneous stochastic regret of the agent at time t.

3.2 Upper Confidence Bound

3.2.1 Introduction

An Upper Confidence Bound (UCB) is a type of statistical confidence interval that provides an upper limit on the expected value of a random variable. It can be used to solve the problem faced in this paper. The idea behind an UCB algorithm is that at each time step, the agent maintains an upper confidence bound for each arm, which is an upper limit on the expected reward of the arm.

This upper limit is computed using the sample mean and variance of the rewards obtained from the arm so far, along with a confidence level. The UCB algorithm then selects the arm with the highest upper confidence bound as the next action to take. This allows the agent to explore arms that have not been pulled many times yet, while also taking advantage of the knowledge gained from previous actions to select arms with potentially high rewards.

3.2.2 Algorithm

The following steps outline the process for implementing the technique and provide a clearer understanding of the associated code. At first, choose the parameters α and β respectively w_0. These parameters define the initial behavior of the algorithm, where the weight of each item e is set to $w_0(e) = (\alpha_e)/(\alpha_e + \beta_e)$.

At each time step t, the UCB algorithm maintains an upper confidence bound $\mathrm{UCB}_t(e)$ for each item e. There are several ways to calculate the value of $\mathrm{UCB}_t(e)$. One method is as follows:

$$\mathrm{UCB}_t(e) = \hat{w}_{T_{t-1}(e)}(e) + c \cdot \sqrt{\frac{1.5 \cdot \log(t-1)}{T_{t-1}(e)}} \tag{7}$$

where $\hat{w}_s(e)$ is the average of s observed weights of item e at time $t-1$, c is a constant that controls the trade-off between exploration and exploitation, $\log(t-1)$ is the natural logarithm of $t-1$, and $T_{t-1}(e)$ is the number of times item e observed up to time $t-1$. The UCB algorithm then selects the list A_t with the highest upper confidence bound:

$$A_t = \underset{A \in \Pi_K(E)}{\arg\max} f(A, U_t). \tag{8}$$

The selected list A_t is taken, and the feedback C_t is observed to update the parameters α and β. The estimated expected reward $w_t(e)$ of item e is updated using the observed reward r and the number of times item e has been taken so far:

$$w_{T_t(e)} = \frac{T_{t-1}(e) \cdot \hat{w}_{T_{t-1}(e)}(e) + \mathbb{1}_{\{C_t = k\}}}{T_t(e)} \tag{9}$$

with $k = 1, ..., \min\{C_t, K\}$. The counting variable $T_t(e)$ of the number of times item e has been taken up to time t is incremented: $T_t(e) = T_{t-1}(e) + 1$.

3.3 Thompson Sampling

3.3.1 Introduction

Like the UCB algorithm, Thompson Sampling assumes a probability distribution over the parameters $(\theta_i)_{(i=1,...,k)}$ which determine the reward $(r_i(\theta_i))_{(i=1,...,k)}$ for each arm. Instead of playing the arm with the highest UCB, it samples a θ_i from the assumed distributions and plays the arm with the highest resulting reward. The distribution is then updated based on the observations made during the interaction with the environment. Both algorithms therefore focus on potentially optimal arms, rather than just those with high sample means.

3.3.2 Algorithm

As shown in the pseudo-code, the UCB and TS algorithms have many similarities and share many theoretical properties. The main difference is that the UCB algorithm uses the confidence interval as a measure of potential optimality, while TS samples from the distribution. Choosing a suitable class of distributions for p can allow for the analytical calculation of updates to the posterior distribution. For example, in the CBP discussed, we assume a Bernoulli distribution q over the rewards with success probability θ, which leads to modeling p as a Beta distribution. This is the conjugate distribution to the Bernoulli distribution, so updating p based on observations from q is analytically tractable. In other cases, numerical sampling methods such as Gibbs sampling may be used Johnson and Jones [2010].

```
input = $K, J, \alpha, \beta$
for   = 0,1,... do
  compute UCB for each item k
  play action with highest UCB
  observe signal and reward
  update params \alpha, \beta
end for
```

```
input = K, J, p
for   = 0,1,... do
  sample $\theta_k \sim p_k$ for each item k
  play $J$–highest $\theta$
  observe signal and reward
  update p as posterior
end for
```

3.3.3 Meta-Thompson

One major challenge with using TS is choosing an appropriate initial prior distribution (i.e. staying in/assuming conjugate classes, to have easy calculations or reverting to more complex numerical sampling methods). Given the correct prior TS can outperform most other algorithms. However, considering too many parameters can result in sampling from and updating of very high-dimensional distributions, leading to computational difficulties.

Thus Wan et al. [2022] proposes a more general framework for TS. They assume a setting where an agent receives a stochastic signal $Y_t \sim f(Y_t|A_t, \theta)$ according to a chosen action A_t and an unknown parameterθ. Afterwards he receives a deterministic reward $R_t = r(Y_t|\eta$ according to a known parameter η. In a Bandit setting with K items we can assume that

$\theta = (\theta_1, \ldots \theta_K)$, thus each item x_i gets its own parameter θ_i To maximize the rewards over time, there are two common approaches. The first one is the feature-agnostic approach to assume that the unknown parameter is a random observation of a distribution: $\theta \sim p$ which has its previously discussed flaws Russo et al. [2018]. The feature-determined approach assumes that there exists a parameterized model $g(\cdot|\gamma)$ such that $\theta_i = g(x_i|\gamma)$ with no stochastic errors at all. That is nearly impossible to get on real data sets even with very high dimensional γ.

Thus they decide to combine these methods to a feature-guided approach by drawing the parameter θ_i from a parameterized distribution $g(\cdot|x_i, \gamma)$, but only after sampling the parameter from another distribution $\gamma \sim Q$. Afterwards they perform the usual steps known from TS and update in the end Q and g separately. Note that one could also view this as sampling θ directly from a higher dimensional distribution and thus as a feature-agnostic approach. But this way it is possible to get much richer classes of possible distributions than in the classic feature-agnostic setting, because it only updates and draws from lower dimensional distributions.

3.3.4 Neural-Thompson

Neural Thompson Sampling is another interesting approach to apply TS to optimize the parameters of a neural network. In this work Zhang et al. [2020] the θ becomes the flattened array of all the weights of a fully connected neural network $f(\cdot, \theta)$ which tries to map the feature vectors of items to the stochastic reward of playing it. But instead of applying standard optimisation methods to get the weights, they sample rewards $\tilde{r}_{t,k}$ from a normal distribution with the current output $f(x_i, \theta_t)$ of the network as mean and variance $\nu^2\sigma^2_{t,k}$ (ν is a chosen exploration parameter). Afterwards they update θ according to standard optimisation techniques regarding an l^2-regularised square loss minimisation problem. The $\sigma_{t+1,k}$ gets updated according to the gradient g of the network wrt. θ evaluated at the new θ_{t+1}.

By doing so they can proof a probabilistic regret bound for their algorithm in contextual bandit settings under reasonable assumptions (see Assumption 3.4 and Theorem 3.5 in Zhang et al. [2020]). In addition to that they show superior performance to other contextual bandit solvers on common known classification data sets such as MNIST.

4 Experiments and Results

4.1 Setting

In our experiments, we applied the cascading bandits method to two different scenarios. We randomly sampled a cascading bandit with 1000 and 50 arms, and with 100 and 10 attractions per instance, respectively. In each case, the attraction probability for each bandit instance, θ_k, was independently sampled from the Beta distribution with parameters α_k and β_k. The parameters (α_k, β_k) were set to $(1, 40)$ and the simulation included 250 time periods and 50 repetitions. For these experiments, we used the modified TS and UCB algorithms from the following source: `https://github.com/iosband/ts_tutorial`. We also implemented the Neural Thompson Sampling approach for the given scenarios with 1 hidden layer of size 24 and for computational reasons we allow the Network only to update for 200 gradient steps on the past 100 observations with a learning rate of 0.005. As a comparison we use a Fully connected Network of the same size and train it w.r.t. mean squared loss. All our experiments can be reproduced with the code from `https://github.com/stephbackv2/Workshop`. Due to computational restrictions we decided to not apply neural TS nor the comparison network to the 1000-100 setting but extend the time horizon there to 10000 steps.

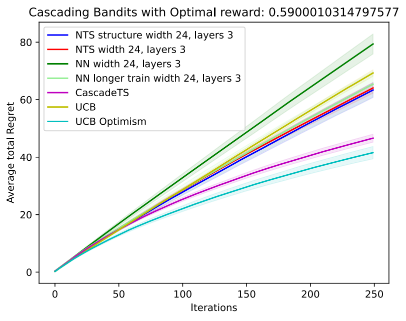

Figure 1: Total regret $(50, 10)$

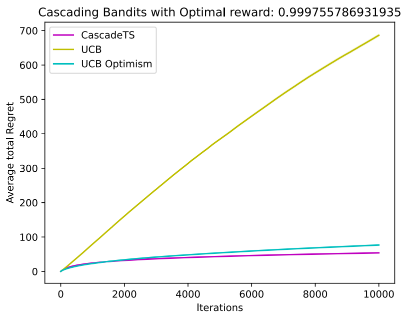

Figure 2: Total regret $(1000, 100)$

4.2 Results

As shown in Figure 2 for the $(K, L) = (1000, 100)$ scenario, the CascadeUCB performs worse than the CascadeTS method. This is because $f(A, w_t)$ is overly optimistic, as it assumes that the probability of a click for every item in A simultaneously takes on the highest attraction probability that is statistically possible. However, because the attractions of each item are statistically independent, it is unlikely that the agent significantly underestimated the attraction probability of every item in A. In contrast, the CascadeTS method samples the θ_k components independently for each item. While any sample might differ significantly from its mean, it is unlikely that the sampled attraction probability of every item in A will significantly exceed its mean. As a result, the variance in $f(A, w_t)$ more accurately reflects the level of uncertainty.

Even when the UCB algorithm is optimally suited to the problem (with a value of c=0.05), it still performs worse than the CascadeTS method. One potential reason for the inferior performance of the UCB algorithm may be the shape of the confidence sets it uses. The algorithm utilizes hyper-rectangular confidence sets, as the set of statistically plausible attraction probability vectors is represented by a Cartesian product of item-level confidence intervals. However, the Bayesian central limit theorem suggests that ”ellipsoidal” confidence sets may be a better choice Dani et al. [2008]. Specifically, as data is collected, the posterior distribution of θ can be accurately approximated by a multivariate Gaussian, the level sets of which are ellipsoidal.

In the second scenario of Figure 1, with $(K, L) = (50, 10)$, the optimally-suited CascadeUCB (with a value of c=0.1) performs better than the CascadeTS method. This may be because hyper-rectangular sets provide poorer approximations of ellipsoids as the dimension increases Dani et al. [2008]. The small advantage in this case may be due to the fact that it is specifically tuned for the given scenario and time horizon.

Neural Networks aren't performing well on stochastic reward tasks directly. Our results show that too, but we can see, that the TS updates perform better than the standard gradient descent updates. It is worth noting, that these updates are computationally very expensive, basically all the run-time of the code goes to the neural TS algorithms.

5 Conclusion

The proposed exploration-exploitation algorithms demonstrated good performance on the presented problem. However, the neural networks did not perform well, likely due to their assumption of small noise and difficulties in predicting 0-1 solutions with the chosen loss function. On the other hand, the Neural Thompson approach showed promise, although it was very time-consuming. While it did not outperform the traditional TS method on this particular problem, it could potentially be useful in more complex scenarios. It is worth considering other approaches, such as using TS to guide the training of a neural network, for future research Russo et al. [2018].

References

M. Abeille and A. Lazaric. Linear thompson sampling revisited. In *Artificial Intelligence and Statistics*, pages 176–184. PMLR, 2017.

S. Agrawal and N. Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International conference on machine learning*, pages 127–135. PMLR, 2013.

J.-Y. Audibert, R. Munos, and C. Szepesvári. Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19): 1876–1902, 2009.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of IEEE 36th annual foundations of computer science*, pages 322–331. IEEE, 1995.

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

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002b.

D. Bouneffouf, I. Rish, and C. Aggarwal. Survey on applications of multi-armed and contextual bandits. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2020.

O. Cappé, A. Garivier, O.-A. Maillard, R. Munos, and G. Stoltz. Kullback-leibler upper confidence bounds for optimal sequential allocation. *The Annals of Statistics*, pages 1516–1541, 2013.

N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422, 2012.

N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *Journal of the ACM (JACM)*, 44(3):427–485, 1997.

W. Chu, L. Li, L. Reyzin, and R. Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 208–214. JMLR Workshop and Conference Proceedings, 2011.

V. Dani, T. P. Hayes, and S. M. Kakade. Stochastic linear optimization under bandit feedback. 2008.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):148–164, 1979.

A. A. Johnson and G. L. Jones. Gibbs sampling for a Bayesian hierarchical general linear model. *Electronic Journal of Statistics*, 4(none):313 – 333, 2010. doi: 10.1214/09-EJS515. URL `https://doi.org/10.1214/09-EJS515`.

B. Kveton, C. Szepesvari, Z. Wen, and A. Ashkan. Cascading bandits: Learning to rank in the cascade model. In *International conference on machine learning*, pages 767–776. PMLR, 2015.

T. L. Lai. Adaptive treatment allocation and the multi-armed bandit problem. *The Annals of Statistics*, pages 1091–1114, 1987.

L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

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

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

R. Wan, L. Ge, and R. Song. Towards scalable and robust structured bandits: A meta-learning framework. *arXiv preprint arXiv:2202.13227*, 2022.

W. Zhang, D. Zhou, L. Li, and Q. Gu. Neural thompson sampling. *arXiv preprint arXiv:2010.00827*, 2020.