Generalized Thompson Sampling in Multi-armed Bandit and Reinforcement Learning

Heyuan Liu, Haoming Yang, Han Feng

December 12, 2018

1 Introduction

The idea of Thompson Sampling is generalized in [3] to contextual bandits, where the experts are weighted in an exponential fashion based on the difference between observed loss and the experts' mean prediction. Moreover, sub-linear regret bounds are proved under general structural assumptions on the loss function. We try to weaken the assumptions of that paper while accommodating the idea from Exp4 [2]. Our proposed setting is then applied to reinforcement learning problem, where we add expert advice in the posterior sampling learning of RL proposed in [1]. Our main results are

- Weaken the assumptions on experts in Generalized Thompson Sampling. Instead of using the mean prediction of all arms for all experts, we only use experts' mean estimates of the best suggested arm and experts' predicted distributions over arms (as in Exp 4). Our proposed loss function incorporates both information in this setting.
- Apply the idea of generalized Thompson Sampling to the expert learning of transition probability and learning of value function in reinforcement learning.

2 Preliminary

2.1 Contextual Bandits

Contextual bandit is a game between the learner and the environment, which can be either stochastic or adversarial. Denote a set of actions by $\mathcal{A}$ and a set of context by $\mathcal{X}$. In each round, the environment chooses a context $x_t \in \mathcal{X}$; we then select an action, or an arm, $a_t \in \mathcal{A}$ based on all history and x_t, resulting in a loss $l_t \in \{0, 1\}$ with expectation $\mu(x_t, a_t)$. Our selection of arms can be based on to a set of experts $\mathcal{E} = \{\mathcal{E}_1, \ldots, \mathcal{E}_K\}$. Each expert has his own guess of $\mu(x, a)$ and for expert k the guess is $f_k(x, a) : \mathcal{X} \times \mathcal{A} \to [0, 1]$. the experts can be greedy, which means given context x, expert k will suggest arm $\mathcal{E}_k(x)$ according to $\mathcal{E}_k(x) := \arg\min_a f_k(x, a)$. We next discuss two algorithms: Thompson Sampling can Generalized Thompson Sampling, that can be used to solve contextual bandits with experts.

2.2 Thompson Sampling on Experts

Algorithm 1 describes Thompson Sampling on contextual bandits problem with experts: start with a prior w_0 from probability simplex Δ_K. In each round we follow one expert's advice with the probability. After that we update the probability for each expert to be the correct one. For the algorithm to work we need a assumption that one of the experts has the correct guess and with out loss of generality we assume the correct expert is $\mathcal{E}_1$. That is, $f_1(x, a) \equiv \mu(x, a)$ for all $x \in \mathcal{X}$ and $a \in \mathcal{A}$.

Algorithm 1: Thompson Sampling on Experts

input : Prior $w_0 \in \Delta_K$

for $t = 0, \ldots, T$ **do**

Observe the context $x_t \in \mathcal{X}$

Select arm a_t according to the probability

$$p_{t,a} := \sum_{k=1}^{K} w_{t,k} \cdot \mathbf{1}\left\{\mathcal{E}_k(x_t) = a\right\}$$

Observe loss y_{t,a_t}

Update the posterior by

$$w_{t+1,k} := \frac{w_{t,k} \cdot f_k(x_t, a_t)^{l_{t,a_t}} \cdot (1 - f_k(x_t, a_t))^{1-y_{t,a_t}}}{\sum_{k'=1}^{K} w_{t,k'} \cdot f_{k'}(x_t, a_t)^{y_{t,a_t}} \cdot (1 - f_{k'}(x_t, a_t))^{1-y_{t,a_t}}}$$

If we further define

$$\tilde{w}_{t,k} := \exp\left(-\sum_{s=0}^{t-1} l(f_k(x_t, a_t), y_{t,a_t})\right), \quad \tilde{W}_t := \sum_{k=1}^{K} \tilde{w}_{t,k},$$

where

$$g(\theta, y) := -\mathbf{1}\left\{l = 1\right\} \cdot \log(\theta) - \mathbf{1}\left\{l = 0\right\} \cdot \log(1 - \theta).$$

It holds that $w_{t,k} = \frac{\tilde{w}_{t,k}}{\tilde{W}_t}$ for all $t \in [T]$ and $k \in [K]$. Hence, the update of posterior is similar to the multiplicative weights update with respect to loss of *inaccuracy* of predictions.

3 Generalized Thompson Sampling

3.1 Algorithm and Implementation

Motivated by the multiplicative weight update interpretation, [3] considers a more general setting which incorporates other loss functions that measure the inaccuracy of predictions, for example, the square loss $g(\theta, l) := (\theta - l)^2$. We present the generalized Thompson Sampling algorithm in Algorithm 2.

Algorithm 2: Generalized Thompson Sampling

input : Prior $p_0 \in \Delta_K$, loss function $g(\cdot, \cdot)$, parameter η, γ

Initialize $w_0 = p_0$

for $t = 0, \ldots, T$ **do**

Observe the context $x_t \in \mathcal{X}$

Select arm a_t according to the probability

$$p_{t,a} := (1-\gamma) \sum_{k=1}^{K} \frac{w_{t,k}}{W_t} \cdot \mathbf{1}\left\{\mathcal{E}_k(x_t) = a\right\} + \frac{\gamma}{K}$$

Observe loss y_{t,a_t}

Update the posterior by

$$w_{t+1,k} := w_{t,k} \cdot \exp\left(-\eta \cdot l(f_k(x_t, a_t), y_{t,a_t})\right), \quad W_{t+1} := \sum_{k=1}^{K} w_{t+1,k}$$

By setting $g(\theta, l) = -\mathbf{1}\{l = 1\} \cdot \log(\theta) - \mathbf{1}\{l = 0\} \cdot \log(1-\theta)$, $\eta = 1$ and $\gamma = 0$, generalized Thompson Sampling in Algorithm 2 reduces to Thompson Sampling on Experts in Algorithm 1.

We implement Algorithm 2 on simulated sequences with different choice of time horizon T. The results are as follows. Here with probability $\gamma(T)$ we choose "expert 0", which is a uniform distribution over all arms.

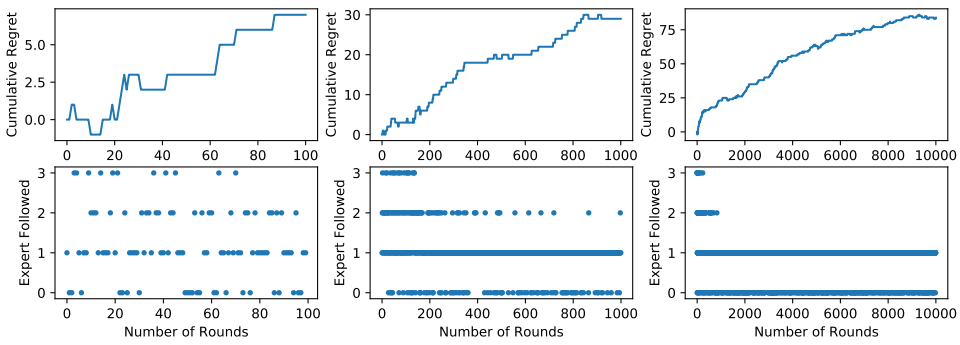

Figure 1: Performance of generalized Thompson Sampling with squared loss

3.2 Comparison with *exp4* and Limitations

We compare the motivations, aims and information needed between *exp4* and *generalized Thompson Sampling*. *exp4* only requires the advice of experts. The predictions of losses of all arms are not needed. The algorithm follows a policy that incurred least accumulative loss, which is estimated with the access to the loss of arms that we pulled. In the regret analysis, there is no assumption

on the *quality* of policies. Even if all policies are bad, the regret bound, which compares with the best policy in hindsight, still holds.

In contrast, *generalized Thompson Sampling* tries to find the policy with correct prediction. In every round it updates the posterior probability by the inaccuracy of prediction, which is calculated with every expert's predicted expected reward for each arm given the context. It requires massive information and no matter which arm we actually pulled, we can update the posterior probability of all policies. Therefore, the partially observed loss gives full feedback on the *quality* of experts.

3.3 Extension: Generalized Thompson Sampling with Partial Prediction

We propose a setting between *exp4* and *generalized Thompson Sampling*. Given a context, each expert will give his advice and the expected loss for that arm. The experts provide more than which arm to pull (the case in *exp4*), while hiding the mean estimate of arms not suggested (unlike *generalized Thompson Sampling*. In short, we observe experts' advise but partially observe the inaccuracy of predictions. In *exp4* case, for all action $a \in \mathcal{A}$ we use the estimate loss

$$\hat{y}_{t,a} := \begin{cases} \frac{y_{t,a}}{p_{t,a}}, & a = a_t, \\ 0, & \text{otherwise}, \end{cases} \tag{3.1}$$

and update the loss of each policy π by

$$\hat{Y}_{t+1,\pi} := \hat{Y}_{t,\pi} + \langle \hat{y}_t, \xi_{t,\pi} \rangle, \tag{3.2}$$

where $\xi_{t,\pi}$ is a distribution over all actions. Motivated by this, we propose a estimation inaccuracy function

$$\hat{l}_{t,a}(\theta, y) := \begin{cases} \frac{l_{t,a}(\theta,y)}{p_{t,a}}, & a = a_t, \\ 0, & \text{otherwise}. \end{cases} \tag{3.3}$$

and update the inaccuracy of each policy $\mathcal{E}_k$ based on context x by

$$\hat{L}_{t+1,\mathcal{E}_k} := \hat{L}_{t,\pi} + \hat{l}_{t,\mathcal{E}_k(x_t)}(f_k(x_t, \mathcal{E}_k(x_t)), y_t). \tag{3.4}$$

where $l_{t,a}$ can be any loss function. Our loss function penalizes the inaccuracy in estimation, as in generalized Thompson Sampling, while remains optimistic in the arms not sampled, as Exp4 does. On a side note, we can also consider a mixed weights update, which is a weighted sum over the loss incurred and the inaccuracy of experts. Since we differentiate the experts not only on the best arm advice, but also on the best mean estimate, we expect the algorithm better able to find the optimal expert. The details of our proposed algorithm is presented in Algorithm 3.

3.4 Discussion of step size and mixing constant

From [3], the mixing constant $\gamma \propto K^{2/3}T^{-1/3}\beta^{1/3}$ is proposed, where β is an upper bound on the loss, K is the number of expert and T is the time horizon. This constant is picked based on a balancing of two terms in the regret estimate: γT and $K\sqrt{\frac{\beta}{\gamma}T\sum_{t=1}^{T}\mathbb{E}_{r,a}[\hat{l} - \hat{l}^*]}$, where l^* is the loss of the best expert. The learning rate parameter η is set to be a constant that only depends on

Algorithm 3: Generalized Thompson Sampling with Partial Predictions

input : Prior $p_0 \in \Delta_K$, loss function $g(\cdot, \cdot)$, parameter η, γ
Initialize $w_0 = p_0$
for $t = 0, \ldots, T$ **do**
Observe the context $x_t \in \mathcal{X}$
Select arm a_t according to the probability

$$p_{t,a} := (1-\gamma) \sum_{k=1}^{K} \frac{w_{t,k}}{W_t} \cdot \mathbf{1}\left\{\mathcal{E}_k(x_t) = a\right\} + \frac{\gamma}{K}$$

Observe loss y_{t,a_t}
Update the posterior by

$$w_{t+1,k} := w_{t,k} \cdot \exp\left(-\eta \cdot \hat{l}(f_k(x_t, a_t), y_{t,a_t})\right), \quad W_{t+1} := \sum_{k=1}^{K} w_{t+1,k}$$

the loss, and with that choice, we are able to quickly pick up the best expert and make sure the $\sum_{t=1}^{T} \mathbb{E}_{r,a}[\hat{l} - \hat{l}^*]$ is finite for all T. In our extension the situation is similar. We fixed a constant learning rate, and the mixing constant decreases with time. We may furthermore adapt the step size and mixing constant as we did in lecture. γ should be decreased once the aggregate mixing loss exceeds a certain budget. The hard part in the adaptation is to faithfully estimate the mixing loss. We left the adaptation as future work.

4 Generalized Thompson Sampling in Reinforcement Learning

4.1 From Bandit to Reinforcement Learning

Multi-armed Bandit (MAB) problem can be understood as a Markov Decision Process (MDP) where a single state is not altered during the pulling of arms. On the other hand, the general problem of Reinforcement Learning (RL) can be understood as solving many MAB problems at the same time in all states — the arms at a state are exactly the actions to take and states are different in the sense that they do separate bandit problems. This observation is the basis of our application of generalized Thompson Sampling to reinforcement learning.

Specifically, recall in RL the goal is to find a policy that maximize the total expected reward over trajectories in a MDP. We can split RL into 2 learning tasks: (1) learning of dynamics, including learning of state value function using sampled rewards and learning of transition probabilities using sampled states; (2) learning of policies, which amounts to the planning of a route between states to gain maximum cumulative rewards. This split is not without restrictions: there is an underlying relation between sets of rewards and sets of transition probabilities, that is, rewards-probabilities sample pairs are drawn from some fixed space, and not all pairs can be sampled. We will focus on task 1, and consider task 2 solvable with dynamic programming methods.

Bandit algorithms are a basic implementation of task 1. In the simplest setting, say we are now at state S_1, forget about which state we will reach after taking this action, focus on how to play this game well if we are allowed to pull only once given previous pulling history and values of each states. Consider arms as actions and rewards of pulling each arm as instantaneous reward plus the expected value of the next states. After the arm is pulled, we transition to a new state and solve another bandit problem. Such transition is captured in transition probabilities, which is assumed to be stationary but unknown (like bandits). The conceptual steps from bandits to RL are sketched below, where we remove restrictions on reward distribution and transition probabilities one at a time

1. Every action has only 2 transition probabilities and exactly reward is zero or one (bernoulli bandit).
2. Every action has only 2 transition probabilities and reward is discrete (discrete bandit)
3. Every action has more than 2 transition probabilities with non-stationary rewards from some distribution. (general RL)

The last, and the most general case contains a lot of transitions and needs a lot of data to learn, so we will instead focus on the simplified setting of case 2.

4.2 Problem Setup

For the underlying dynamics, we assume 5 states and 2 actions, and for any action we take, 2 states may be reached. For the reward, we assume they are discrete, taking values over $\{0, \ldots, 9\}$. If context information is available, we assume there is always one correct expert taking the context and gives correct answers.

The sections to follow mirror the previous development of contextual bandits. We will first present a general algorithm called Posterior Sampling for Reinforcement Learning (PSRL) [1] and see how Thompson Sampling extends to RL in learning the transition probabilities. Next we describe how contextual bandit ideas can be extended to RL. Finally propose an empirical algorithm that combines PSRL with General Thompson Sampling. We shall see that the nature of PSRL is more explored based than General Thompson Method, and therefore our combination could be promising.

4.3 PSRL Algorithm

Our division of RL learning tasks gives rise to two levels of explore-exploit trade-offs. The first level is exploration vs. exploitation of actions and dynamics at a state, and the second level is exploring different states vs. exploiting one states deeper. Level 2 is an planning problem where the MDP is assumed to be correct and best policies at every state is found using algorithms from Dynamic Programming. The PSRL Algorithm specializes the first level, where the goal is to collect maximum rewards and learn good models for the current round. It imitates Thompson Sampling and keeps a prior on each (S, A) pairs using Dirichlet distribution. The posterior update is calculated with state sampling. At the first round, the prior is set to be uniform over the next state that can be transitioned to. With epoch increasing, the posterior will finally converge to true result as long as

our second level sampling can generate each (S, A) pairs infinitely frequently. One may terminate the epoch early if some states are visited too often. The algorithm is outlined in Algorithm 4.

Algorithm 4: Posterior Sampling for Reinforcement Learning (PSRL)

for *episode* $e = 1, \ldots, E$ **do**
- Sample all (S, A) pairs to get a MDPs model
- Solve MDPs and obtain current optimal policy π_e
- Follow the policy π_e for the remaining epoch, terminate the epoch early if necessary
- Update the posterior using the samples from the current epoch

We implemented this algorithm and compared the rewards and regret against the uniform policy. We find that PSRL is able to achieve low regret and is able to detect the best policy as times goes on. Interestingly, PSRL may be worse than random policy for the first several rounds.

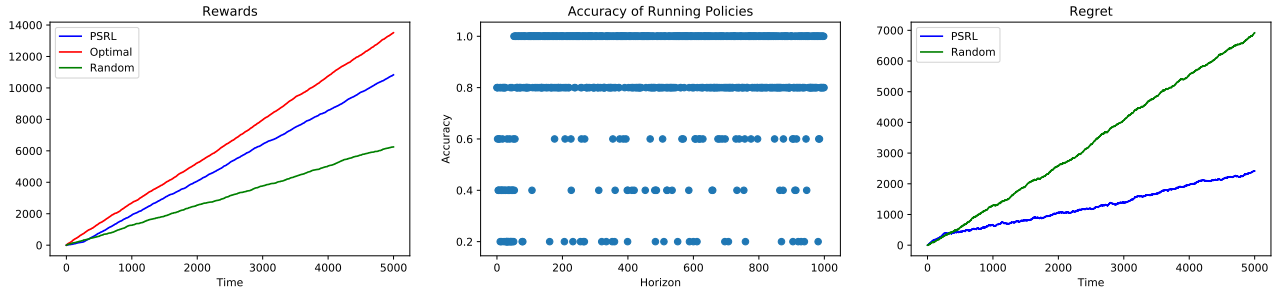

Figure 2: Rewards, Regrets and Accuracy under problem setup with 1000 episodes of PSRL

4.4 Contextual Bandits and RL

How to incorporate context information into RL? As mentioned previously, part of RL involves solving a separate bandit problem at each state, if we ignore the change of states and only care about how to do well at a given state. We apply the idea from Generalized Thompson Sampling and sketch a preliminary Algorithm 5. The algorithm is essentially local in that whenever we reach a state, we use local expert information to make a decision and update the local information. The algorithm ignores the trade-off between instantaneous reward and total reward. That is there is one always correct expert, he may insist you go somewhere with large rewards in future while suffer a larger local regret. As a result, the algorithm may stuck at local optimum, and it needs to be complemented by a global strategy.

4.5 An Empirical Algorithm

Algorithm 5 cannot balance between getting a good reward now and going to a good state that rewarded more in future. To solve this problem, we need a modified reward by adding the expected

Algorithm 5: Contextual Bandit Algorithm in a RL set-up

for *Round* $e = 1, \ldots, T$ **do**
- Receive contextual information and expert advice of the current state
- Use General Thompson Sampling to decide which action to choose at this state
- Observe transition reward and move on to next state
- Update the weight of each expert with the observed reward and transition

value of the next state and the transition rewards to feed into General Thompson Method. We also need to learn transition probabilities. Recall that PSRL can learn good state transition probabilities and collect decent accumulated rewards, it is natural to ask if we can combine them together to get something better: using PSRL as a global exploring method and General Thompson as local exploiting method. Both are known to guarantee sub-linear regrets, so we have reason to believe that combined they work well. We describe the algorithm in Algorithm 6

Algorithm 6: Our Empirical Algorithm

for *epoch* $e = 1, \ldots, E$ **do**
- Run General Thompson Sample for T_e rounds
- Run PSRL with a designed prior based on previous observations for P_e rounds
- Run Value Iteration based on previous samples to obtain expected rewards for each states
- Keep running General Thompson with updated rewards for all rounds left in the epoch
- Start next epoch with adjusted P_e and T_e

We have not implemented this idea but we will explain the motivation behind the algorithm. Since there is an always a correct expert, the main task is how to find out this expert as quickly as we can and follow his prediction. Instead of starting with PSRL which fumbles in the underlying world inefficiently as seen in Figure 2, we use the context as a guide to at least achieve good local understanding of experts. After some data from such exploration, so we can use PSRL with a designed prior that encourages exploration of other arm while not suffer great loss. The designed prior also need to encourage exploring of the transition probabilities for those actions rarely recommended by the General Thompson method: say if two arms A is sampled with $Pr(A) = 0.8$, and B is sampled with $Pr(B) = 0.2$, it can has a prior of $(A, B) = (0.2, 0.8)$ feed into PSRL for state action pairs. When we are confident about the world and about the correct experts we should rely on, we should stick to his policy as soon as possible, say by decreasing PSRL exploring time P_e.

5 Summary and Future Work

Our project extended the ideas of generalized Thompson Sampling to contextual bandit with partial observations and applied the idea to reinforcement learning. Some unresolved problems include

- Regret Analysis of our proposed methods. From the implementation we expect sub-linear

regret bound to hold. But the proof seems non-trivial.

- Adapt general Thompson methods to more continuous rewards. Our setting assumed discrete Bernoulli reward, but as the generalization in reinforcement learning shows, we need it to be able to work with value function of states, hence a continuous reward space is needed.
- Adaptive choice of hyper-parameters. The generalized framework includes both a mixing parameter γ and a learning rate η. They seem to depend on an estimated loss compared with the best expert in hindsight.
- Adaptive choice of epoch length and P_e, T_e portion of every epoch.

References

[1] S. Agrawal and R. Jia. Optimistic posterior sampling for reinforcement learning: Worst-case regret bounds. page 11. 00017.

[2] S. Bubeck and N. Cesa-Bianchi. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, Dec. 2012. 01058.

[3] L. Li. Generalized thompson sampling for contextual bandits. *arXiv preprint arXiv:1310.7163*, 2013.