Statistical Techniques in Robotics (16-831, S22) Lecture #13 (Wednesday, March 02)

Thompsons Sampling, EXP3, EXP4

Lecturer: Kris Kitani *Scribes: Dakshit Agrawal, Meghdeep Jana*

1 Review

In the previous lecture, we discussed two multi-arm bandit(MAB) algorithms: Explore-Exploit and Upper Confidence Bound(UCB). We review these algorithms

	Context-free	Contextual
Stochastic environment	Explore-Exploit, UCB, Thompsons	linUCB
Adversarial environment	EXP3	EXP4

Figure 1: Types of MAB Algorithms

1.1 Explore-Exploit

The Explore-Exploit algorithm is shown in Algorithm 1.

Algorithm 1 Explore-Exploit[8]

Require: M

1: **for** $k = 1, \cdots, K$ **do**
2: **for** $m = 1, \cdots, M$ **do**
3: $a = k$
4: Receive(r)
5: $\hat{\mu}_k = \hat{\mu}_k + \frac{r}{M}$
6: **end for**
7: **end for**
8: **for** $t = KM, \cdots, T$ **do**
9: $a^{(t)} = \arg\max_k \widehat{\mu}'_k$
10: **end for**

The algorithm has two phases (a) Explore phase(lines 1-7) where we make KM pulls (b) Exploitation Phase(lines 8-10) where go from round KM to T. The regret is derived for each phase using the Hoeffding's inequality.

Definition 1. ***Hoeffding's inequality***[7]: Consider a one-dimension distribution ν with expectation μ, where any sample $r \sim \nu$ is bounded such that $r \in [0, 1]$. Given T i.i.d. samples ${r^{(t)}}_{t=1}^{T}$, we have that for any ϵ:

$$p(|\sum_{t=1}^{T} \frac{r^{(t)}}{T} - \mu| \geq \epsilon) \leq 2e^{-2T\epsilon^2} \tag{1}$$

We get the regret bound for explore phase as $R_{\text{explore}} = \mathcal{O}(KM)$ and for the exploit phase, we get $R_{\text{exploit}} = \sum_{t=KM+1}^{T}(\mu_{k^*}^{(t)} - \mu_{\hat{k}}^{(t)}) \leq (T - KM) \cdot 2\sqrt{\frac{\log(2/\delta)}{2M}}$. We get the total regret bound by adding the two regrets and upper bounding it.

$$R_{\text{explore-exploit}} \leq KM + 2T \cdot \sqrt{\frac{1}{M}} \tag{2}$$

The optimal M was found to be $(\frac{T}{K}^{2/3})$. From this we get the Asymptotic Regret bound as $R_{\text{explore-exploit}} = \mathcal{O}(K^{1/3}T^{2/3})$. Which shows that Explore-Exploit grows sublinearly with time and is a no-regret algorithm.

1.2 UCB

Using the Heoffding's inequality, a confidence bound is defined.

Definition 2. ***Confidence Bound:*** We get that the optimal value of ϵ for which the hoeffding's inequality is upper bounded by a specified value δ is:

$$\epsilon = \sqrt{\frac{log(2/\delta)}{2T}} \tag{3}$$

The UCB algorithm is shown in Algorithm 2.

Algorithm 2 UCB

Require: δ, T

for $t = 1, \cdots, T$ **do**
  if $t \leq k$ **then**
    $k = t$
  else
    $k = \arg\max_{k'} \left(\widehat{\mu}_{k'} + \sqrt{\frac{\log(2T/\delta)}{2T_{k'}^{(t-1)}}} \right)$
  end if
  Receive$(r^{(t)})$
  $T_k^t = T_k^{(t-1)} + 1$
  $T_{k'}^t = T_{k'}^{(t-1)}\ \forall k' \in [K] \setminus \{k\}$
  $\widehat{\mu}_k = \frac{1}{T_k^{(t)}} \left(T_k^{(t-1)} \widehat{\mu}_k + r_k^{(t)} \right)$
end for

For UCB, we get the asymptopic regret bound $R_{UCB} = \mathcal{O}(\sqrt{KT})$. This shows that UCB grows sub-linearly with time and is a no-regret algorithm.

2 Summary

We will first revise concepts of Probability Theory (Sec. 2.1). Using these concepts, we will derive Thompsons Sampling (Sec. 2.2). Afterwards, we will define terms (Sec. 2.3) followed by applying them to understand EXP3 (Sec. 2.4) and EXP4 (Sec. 2.5) algorithms as well as differentiate between the two (Sec. 2.6).

2.1 Probability Theory

2.1.1 Bayes Theorem

For an evidence e and a hypothesis H, Bayes theorem is defined as follows:

$$P(H|e) = \frac{P(e|H)P(H)}{P(e)} \tag{4}$$

The Bayes theorem consists of four probabilities as explained below:

Definition 3. *$P(H)$, known as **Prior**, defines how probable the hypothesis was before observing the evidence.*

Definition 4. *$P(e|H)$, known as **Likelihood**, defines how probable observing the evidence is given the hypothesis.*

Definition 5. *$P(e)$, known as **Marginal**, defines how probable observing the evidence is given all possible hypothesis $\left(\sum_{h \in \mathcal{H}} P(e|h)P(h)\right)$. This quantity is usually very hard to calculate.*

Definition 6. *$P(H|e)$, known as **Posterior**, defines how probable the hypothesis is given the observed evidence. It is not directly computable.*

2.1.2 Markov Assumption

For a random process, the **Markov property** [6] says that if we have the present value of the process, we won't get any additional information about the future behaviour of the process by gathering more information about the past. Mathematically, it means that the conditional distribution of future states of the random process given present and past states depends only on the present state and not at all on the past states (memoryless property).

$$P(\text{future} \mid \text{present}, \text{past}) = P(\text{future} \mid \text{present}) \tag{5}$$

Definition 7. *The term **Markov assumption** is used to describe a model where the Markov property is assumed to hold.*

2.1.3 Conjugate Priors

Eq. 4 can also be written as:

$$P(H|e) \propto P(e|H)P(H) \tag{6}$$

Definition 8. *If the prior $P(H)$ and posterior $P(H|e)$ have the same type of distribution, they are called* ***conjugate distributions*** *and the prior $P(H)$ is called the* ***conjugate prior*** *of the likelihood function [5].*

Theorem 9. *The Beta distribution is the conjugate prior of the Bernoulli distribution. Specifically, if we assume a Bernoulli likelihood with parameter r and Beta prior with parameters α and β, we get a Beta posterior with parameters $\alpha' = \alpha + r$ and $\beta' = \beta + 1 - r$.*

Proof.

$$\begin{aligned}
\text{Bernoulli distribution} \quad & p(r|\theta) = \theta^r(1-\theta)^{1-r} \\
\text{Beta distribution} \quad & p(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1} \\
\text{Gamma function} \quad & \Gamma(n) = (n-1)!
\end{aligned}$$

Using Eq. 6:

$$\begin{aligned}
p(\theta|r) &\propto p(r|\theta)p(\theta) \\
&\propto \theta^r(1-\theta)^{1-r}\theta^{\alpha-1}(1-\theta)^{\beta-1} \quad \text{(Plug in distributions)} \\
&\propto \theta^{r+\alpha-1}(1-\theta)^{1-r+\beta-1} \\
&\propto \theta^{(\alpha+r)-1}(1-\theta)^{(\beta+1-r)-1} \\
&\propto \theta^{\alpha'-1}(1-\theta)^{\beta'-1} \quad \text{(Beta distribution)}
\end{aligned}$$

2.2 Thompsons Sampling

Thompsons Sampling [10] is a strategy in the same class as Explore-Exploit (Sec. 1.1) and UCB (Sec. 1.2), i.e. it is one-shot, exhaustive and evaluative algorithm in a stochastic environment.

Each arm has a generative distribution conditioned on parameter θ and an action a from which a reward is sampled. Thus, we assume that each reward r is drawn from a parameterized distribution (likelihood function) given by:

$$r \sim p(r|a, \theta) \tag{7}$$

If the true parameters of the above likelihood function are available, the optimal arm for highest reward is given as:

$$a = \arg\max_k \mathbb{E}_{p(r|a_k,\theta_k^*)}[r|a_k, \theta_k^*] \tag{8}$$

However, we don't have the true parameters of the likelihood function! We need to estimate the true parameters so as to find the optimal arm for highest reward.

Given the parameter prior $P(\theta)$, we can use Bayes Rule (Eq. 4) to estimate the posterior distribution $P(\theta|h^{(t)})$ of the parameter θ given a history $h^{(t)}$ of actions and rewards and consequently get the best estimate of the parameter θ by:

$$\hat{\theta} = \arg\max_\theta p(\theta|h^{(t)}) \tag{9}$$

Mathematically,

$$p(\theta|h^{(t)}) \propto p(h^{(t)}|\theta)P(\theta) \tag{10}$$
$$p(\theta|h^{(t)}) \propto p(a^{(1)}, r^{(1)}, \ldots, a^{(t)}, r^{(t)}|\theta)p(\theta) \tag{11}$$

By Markov assumption (Def. 7), the reward at each time step depends on only the present action:

$$p(r^{(t)}|a^{(t)}, \ldots, r^{(1)}, a^{(1)}, \theta) = p(r^{(t)}|a^{(t)}, \theta) \tag{12}$$

Eq. 11 can be rewritten using Eq. 12 as:

$$p(\theta|h^{(t)}) \propto \prod_t p(r^{(t)}|a^{(t)}, \theta)p(\theta) \tag{13}$$

Eq. 13 can be written in recursive form:

$$p(\theta|h^{(t)}) \propto p(r^{(t)}|a^{(t)}, \theta)p(\theta|h^{(t-1)}) \tag{14}$$

Hence, the best estimate of the parameter can be calculated incrementally (combining Eq. 9 and Eq. 14):

$$\hat{\theta} = \arg\max_\theta p(r^{(t)}|a^{(t)}, \theta)p(\theta|h^{(t-1)}) \tag{15}$$

Thus, Thompsons Sampling Multi-Arm Bandit (Algorithm 3) consists of the following steps repeated T times:

1. Sample parameter θ for each arm k from prior distribution of that arm (effectively posterior distribution of that arm in previous time step as can be seen in Eq. 14).
2. Select the best arm a_k by using Eq. 8.
3. Take the action and recieve a reward r.
4. Update the parameter estimate (posterior) of only the pulled arm using Eq. 14.

Algorithm 3 Thompsons Sampling

for $t = 1, 2, \ldots, T$ **do**
  $\theta_k \sim p(\theta_k|h_k) \quad \forall\, k$ ▷ Sample from posterior
  $a_{\hat{k}}^{(t)} = \arg\max_k \mathbb{E}_{p(r|a_k,\theta_k)}[r|a_k, \theta_k]$ ▷ Predict
  RECEIVE $(r^{(t)})$ ▷ Get sampled reward
  $p(\theta_{\hat{k}}|h_{\hat{k}}) \propto p(r^{(t)}|a_{\hat{k}}^{(t)}, \theta_{\hat{k}})p(\theta_{\hat{k}}|h_{\hat{k}})$ ▷ Update posterior
end for

Although the posterior update process is incremental, it is still inefficient. We can parameterize the distributions using conjugate distributions (Def. 8) to make the posterior update process efficient. Specifically, we have seen in Theorem 9 that if we assume a Bernoulli likelihood with parameter r

and Beta prior with parameters α and β, we get a Beta posterior with parameters $\alpha' = \alpha + r$ and $\beta' = \beta + 1 - r$. Algorithm 4 updates Thompsons Sampling to use this fact.

Algorithm 4 Bern-Beta Thompsons Sampling

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\theta_k \sim p(\theta; \alpha_k, \beta_k) \quad \forall\, k$ ▷ Sample from posterior
3: $a_{\hat{k}}^{(t)} = \arg\max_k \mathbb{E}_{p(r|a_k,\theta_k)}[r|a_k, \theta_k]$ ▷ Predict
4: RECEIVE $(r^{(t)})$ ▷ Get sampled reward
5: $\alpha_{\hat{k}} = \alpha_{\hat{k}} + r^{(t)}$ ▷ Update posterior
6: $\beta_{\hat{k}} = \beta_{\hat{k}} + 1 - r^{(t)}$ ▷ Update posterior
7: **end for**

Thompsons Sampling has lower empirical regret as compared to UCB (Sec. 1.2), especially when the number of time steps is large [4]. Thompsons Sampling is a no-regret algorithm with a regret bound of $O(\sqrt{KT \log T})$ where K is the number of arms and T is the number of time steps for which we run the algorithm.

2.3 Preliminiaries for Adversarial MAB Algorithms

2.3.1 Adversarial Environment

In this section, we switch from a stochastic environment to an adversarial environment for the MAB problem. In an adversarial setting, the environment can observe the learning agents' actions and can decide the feedback(reward) based off it. This removes the assumption that rewards follow a certain distribution and thus require algorithms than can provide a generalized solution to the MAB problem. We discuss two such algorithms, Exponential-Weight Update for Exploration and Exploitation(EXP3) for context-free and Exponential-Weighted Update for Exploration and Exploitation with Experts(EXP4) for contextual MAB.

2.3.2 Unbiased Estimator

We first define the Unbiased Estimator that is key to the EXP algorithms. An unbiased estimator of a parameter is an estimator whose expected value is equal to the parameter[1]. More formally, if the estimator S is being used to estimate a parameter θ , then S is an unbiased estimator of θ if :

$$\mathbb{E}[S] = \theta \tag{16}$$

For our Adversarial MAB problem, we need to define a reward estimator. For our problem, we need an unbiased reward estimator. If we define our reward estimator as follows:

$$c_i^{(t)}(a^{(t)}) = \mathbf{1}[a^{(t)} = 1] \cdot r^{(t)} \tag{17}$$

We get that the expectation of our estimator over T rounds doesn't equal the true reward for Multi-arm bandit case.

To make our estimator unbiased, we define a custom reward estimator as follows:

$$c_i^{(t)}(a^{(t)}) = \mathbf{1}[a^{(t)} = 1] \cdot r^{(t)} \cdot \frac{1}{p_i} \tag{18}$$

The $1/p_i$ term here is known as the *Inverse probability weight* or *importance sampling term* and $c_i(a)$ denotes the reward esimator for the i^{th} arm. The $1/p_i$ helps in forming an unbiased reward estimator.

We assume a 2-arm case here to show that the estimator is unbiased. The expected value of the estimator becomes

$$\mathbb{E}_{p(a)}\left[c_1^{(t)}\left(a^{(t)}\right)\right] = p\left(a^{(t)} = 1\right) \cdot c_1^{(t)}(1) + p\left(a^{(t)} = 2\right) \cdot c_1^{(t)}(2) \tag{19}$$

$$= p\left(a^{(t)} = 1\right) \mathbf{1}\left[a^{(t)} = 1\right] \cdot r^{(t)} \cdot \frac{1}{p_1} + p\left(a^{(t)} = 2\right) \mathbf{1}\left[a^{(t)} = 1\right] \cdot r^{(t)} \cdot \frac{1}{p_1} \tag{20}$$

$$= (0.5) \cdot 1 \cdot r^{(t)} \cdot \frac{1}{0.5} + (0.5) \cdot 0 \cdot r^{(t)} \cdot \frac{1}{0.5} \tag{21}$$

$$= r^{(t)} \tag{22}$$

This can be extended to multi-arm case to show that the reward estimator is unbiased.

2.4 EXP3

The EXP3 algorithm is shown in Algorithm 5.

Algorithm 5 EXP3 Algorithm[2]

Require: $\gamma \in [0, 1]$

1: $\boldsymbol{w}^{(1)} \leftarrow \{w_k^{(1)}\}_{k=1}^K$ ▷ weights over actions
2: **for** $t = 1, \cdots, T$ **do**
3: $\quad \boldsymbol{p}^{(t)} = \frac{\boldsymbol{w}^{(t)}}{\sum_k w_k^{(t)}}$ ▷ probability over actions
4: $\quad k \sim$ MULTINOMIAL$(\boldsymbol{p}^{(t)})$ ▷ take and draw action
5: $\quad a^{(t)} = a_k$
6: $\quad$ RECEIVE $(r^{(t)} \in [0, 1])$ ▷ get reward
7: $\quad w_k^{(t+1)} \leftarrow w_k^{(t)} \exp\left(\gamma \cdot r^{(t)}/p_k^{(t)}\right)$ ▷ update weights
8: **end for**

We can see that the EXP3 algorithm is almost exactly the same as the RWMA algorithm. Though the EXP3 has two key differences than the RWMA algorithm. Firstly, EXP3 receives partially observable rewards, thus leading to the weight update of only the arm that was pulled at that time step whereas the hedge algorithm received fully observable rewards. Secondly, the update step (line 7) for the weights is an exponential update. The term $r^{(t)}/p_k^{(t)}$ in the update step comes from the unbiased estimate of the reward. This unbiased estimate is used to convert the partially observed loss function to a fully observed loss function which helps in setting up the problem as *prediction with one expert advice*.

The intuition behind this is that, depending on the relative value of reward to the probability of pulling the arm k, we are changing the probability of pulling the arm k for the next time step . So for a tiny value of $p_k^{(t)}$, the arm k will be exploited if it's reward was high. In the same way, if the reward of the arm k was low then exploration will follow.

We get the asymptotic regret for EXP3 as

$$R_{EXP3} = O(\sqrt{TKlogK}) \tag{23}$$

$$\text{when } \gamma = \sqrt{\frac{logK}{TK}} \tag{24}$$

Where T is the time horizon and K is the total number of arms. This shows that EXP3 grows sub-linearly with T and is a no-regret algorithm.

2.5 EXP4

The environment for EXP4 is different than EXP3 as now we assume that the state is now contextual, where context is sampled from infinite space at each time step. The EXP4 algorithm is shown in Algorithm 6.

Algorithm 6 EXP4 Algorithm[2]

Require: $\gamma \in [0, 1], T$

1: $\boldsymbol{w}^{(1)} \leftarrow 1 \in R^N$ ▷ weights over experts
2: **for** $t = 1, \cdots, T$ **do**
3: RECEIVE $(\boldsymbol{X}^{(t)} \in R^{N \times K})$ ▷ advice from N experts
4: $\boldsymbol{q}^{(t)} = \frac{\boldsymbol{w}^{(t)}}{||\boldsymbol{w}||^2} \cdot \boldsymbol{X}^{(t)} \in \Delta^K$ ▷ probability over actions
5: $k^{(t)} \sim$MULTINOMIAL$(\boldsymbol{q}^{(t)})$ ▷ draw action
6: RECEIVE $(r^{(t)})$ ▷ get reward
7: $\hat{\boldsymbol{r}}^{(t)} = \frac{r^{(t)}}{q_k^{(t)}} I[k = k^{(t)}] \in I^K$ ▷ reward over all arms
8: $\boldsymbol{g}^{(t)} = \boldsymbol{X}^{(t)} \cdot \hat{\boldsymbol{r}}^{(t)} \in R^N$ ▷ per expert reward
9: $w_n^{(t+1)} \leftarrow w_n^{(t)} \exp\left(\gamma \cdot g_n^{(t)}\right) \quad \forall n$ ▷ update
10: **end for**

EXP4 is mostly similar to EXP3 except that it uses expert advice to make better decisions. Here, we are maintaining the weights over experts (there are N experts) whereas for EXP3 we were maintaining weights over arms. In line 3, we receive context $(X^{(t)})$ from the environment, where each row of $X^{(t)}$ is a probability distribution. In line 5, we construct $q^{(t)}$ vector of dimension K which is the probability distribution over experts. Line 7, is the inverse probability reward estimator to form the fully observable reward. We then calculate the gain $(g^{(t)})$ which is how well each expert performed. We use this gain in line 9 to perform an exponential update to the weight of each expert.

We get the regret bound for EXP4 as:

$$R_{EXP4} \leq \sqrt{KTlogN} \tag{25}$$

Where T is the time horizon, K is the total number of arms and N is number of experts. This shows that EXP4 also grows sub-linearly with T and is a no-regret algorithm.

2.6 EXP3 vs EXP4

The difference between EXP3 and EXP4 is the $logK$ and the $logN$ term in their respective regret bounds. If the number of arms are very large, then we might want to solve the context-free problem posed as a contextual bandit problem and use EXP4 algorithm. One example would be ad placement where number of arms are huge. So the regret bound for EXP3 will be large. If we can have a smaller N, say a neural network that suggests ads, we can solve the ad placement problem as a contextual bandit problem and guarantee a lower regret bound.

References

[1] A. Adhikari. Unbiased estimators. [Online: http://stat88.org/textbook/notebooks/Chapter_05/04_Unbiased_Estimators.html].

[2] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[3] R. Bonnefoi, L. Besson, C. Moy, E. Kaufmann, and J. Palicot. Multi-armed bandit learning in iot networks: Learning helps even in non-stationary settings. *CoRR*, abs/1807.00491, 2018.

[4] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[5] W. contributors. Conjugate prior. *Wikipedia, the free encyclopedia, 2022*, Online; accessed 02-Mar-2022.

[6] W. contributors. Markov property. *Wikipedia, the free encyclopedia, 2022*, Online; accessed 02-Mar-2022.

[7] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[8] T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

[9] 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.

[10] 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, 12 1933.

3 Appendix

3.1 Real-Life Use Cases of Thompsons Sampling

Thompsons Sampling has been used in many real-life applications, several of which have been covered in [9]. The authors also provide their code that can be run locally (GitHub Link). We go through a few use cases in the following subsections.

3.1.1 News Article Recommendation

Given K news articles, the recommendation algorithm tries to learn to show personalized news articles to its users. Each news article is associated with a parameter θ. There are a sequence of users from which the algorithm learns. For every user (time step t), the recommendation algorithm looks at the user's feature vector (encodes the user's information like recent reading activity, demographics, etc.) and selects a news article. The recommender then observes whether the user clicked the news article or not (reward $r \in 0, 1$). A simulation in a simplified environment of 3 news articles is provided here.

3.1.2 Product Assortment

Given N products, an agent tries to determine a subset of these products that would get them the maximum profit. At any period of time (time step t), an agent chooses a subset of products to showcase. The profit (r) depends on the demand θ of each product conditioned on whether other products are in the subset. A simulation in a simplified environment of 6 products is provided here.

3.1.3 Cascading Recommendation

Given K websites, a search engine needs to show J websites in an ordered sequence. Every time a user searches on the search engine (time step t), the search engine shows J websites based on a attraction vector θ that contains the probability that the user finds a particular website attractive. The search engine receives a positive reward (r) if the user visits any of the websites in the list of J articles. A simulation in a simplified environment of K=1000 and J=100 is provided here.

3.1.4 Active Learning with Neural Networks

Active learning not only generates actions that maximizes the immediate reward, but it also probes the environment to generate data that accelerates learning. While using neural networks for these problems, computing the posterior distribution is intractable. Thompsons sampling proves useful in approximating the posterior distribution. An incremental Thompsons sampling fits well with the incremental learning procedure of neural networks.

3.1.5 IoT Networks

A recent work [3] evaluates the performance of several multi-arm bandit algorithms in IoT networks, and finds that both UCB and TS have near-optimal performance even when their underlying i.i.d. assumption is violated by the many "intelligent" end-devices.