From today onwards we will be discussing exploration. Before discussing it in full generality, let’s first consider the simpler context of bandit problem, where episode experience terminates after one step.

We denote,

**Environment:** $E(\mathcal{O}, \mathcal{A}, \mathcal{H}, \rho)$

**Episode experience:** $(\mathcal{O}_{t,0}, \mathcal{A}_{t,0}, \mathcal{O}_{t,1}) = \mathcal{H}_{t,1}$, where $\mathcal{O}_{t,1} \sim \rho(\bullet | \mathcal{H}_{t,0}, \mathcal{A}_{t,0})$ and $\mathcal{O}_{t,0}$ would be included in $\mathcal{H}_{t,0}$. All that matters in terms of modeling environment is the probability of sampling each possible observation given our initial history.

**Reward function:** $\mathcal{R}_{t,1} = r(\mathcal{H}_{t,1}) = r(\mathcal{O}_{t,0}, \mathcal{A}_{t,0}, \mathcal{O}_{t,1})$

Sometimes the problem is referred to as “contextual” as the first observation can be thought of as a context for the problem. But for simplicity we will assume $\mathcal{O}_{t,0} = \text{null}$ and therefore is irrelevant. In short, we have a simple mapping of $\mathcal{A}_t, \mathcal{O}_t \rightarrow \mathcal{H}_t, \mathcal{R}_t$.

## 1 Bernoulli bandit

We consider the specific example of Bernoulli bandit problem where action space $\mathcal{A} = \{1, \ldots, k\}$ describes possible arms to pull, observation takes value $\mathcal{O} = \{\text{null}, 0, 1\}$ and $\rho(\mathcal{H}_0, \mathcal{A}_0) = \theta, \mathcal{A}_0$ describes the ‘probability of success’.

A concrete example of this problem can be a game of tossing $k$ coins (biased differently) and trying to maximize number of heads. The environment can be identified by $\theta \in [0, 1]^k$, which can be obtained from our experience of getting a series of heads or tails.

### 1.1 RL algorithm

Denote the data accumulated over episodes (e.g. which coins were tossed and their outcomes) as $\mathcal{D}_t$.

With $\mathcal{D}_t$ as input, we want a black box RL algorithm s.t. $\mathcal{D}_t \rightarrow \text{algorithm} \rightarrow \pi_t \rightarrow A_{t+1}$

There are two ways to address this problem.

The native approach: try to learn what is $\theta$ and the algorithm just spits the optimal action for that environment.

The more robust approach: since we cannot know $\theta$ in advance, we should look for an algorithm which behaves as if an adversary is choosing a $\theta$.

For example, if we want to design a RL agent for posting banner ads and we have no idea what would be the clicking rates. In order to eke out as much value as possible, our optimizing strategy shouldn’t focus too much on one specific clicking rate. For example, an optimized algorithm for clicking rate being 90% may be very rewarding, but it is extremely rare to that environment. Instead, it is better to optimize for a distribution of possible environments as if an “adversary” gets to pick the environment.

Here the prior distribution aims to capture the range of $\theta$ that you want your agent to work well on. Let’s say prior is $P(\theta_k \in \bullet) \sim \text{unif}(\{0, 1\})$ (or equivalently beta(1,1))

Then we can identify the posterior distribution to be from a beta distribution: $P(\theta_k \in \bullet | \mathcal{D}_t) \text{ beta}(\alpha_{t,k}, \beta_{t,k})$

for some $\alpha, \beta$.

Since $\mathcal{O}_t \in \{0, 1\}$ denotes whether success of failure, let’s say $R_t = \mathcal{O}_t$.

Then we can update $\alpha_{t,k}$ and $\beta_{t,k}$ as:

$$
(\alpha_{t+1,k}, \beta_{t+1,k}) = \begin{cases} 
(a_{t,k}, b_{t,k}), & \text{if } A_t \neq k \\
(a_{t,k}, b_{t,k}) + (R_{t+1}, 1 - R_{t+1}), & \text{if } A_t = k 
\end{cases}
$$
i.e. $\alpha$ and $\beta$ count number of successes and failures.

Then we have

$$E[\theta_k|D_t] = \frac{a_{tk}}{(a_{tk} + b_{tk})}$$

For the described problem setup, we can have the following RL algorithms:

1.1.1 greedy:

We can simply choose action to maximize the expected reward:

$$A_{t+1} = \arg \max_{a \in \mathcal{A}} E[R_{t+1}|D_k, A_{t+1} = a] = \frac{\alpha_{la}}{(\alpha_{la} + \beta_{la})}$$

The drawback of this method is: if good $\theta_k$ is not well-explored, we may never know it is good. For example, in Fig. 1, if we are given the posterior of $\theta_1$ and $\theta_2$, $\theta_1$ has quite some chance of being better than $\theta_2$, but a greedy algorithm will always pick action 2 over 1 and therefore increase confidence over $\theta_2$ without learning about $\theta_1$.

![Figure 1: Example of posterior distributions](image)

1.1.2 dithering:

To address the problem with greedy algorithm, we can simply add some noise to decision to make sure we try all the actions:

$$A_{t+1} = \begin{cases} 
\arg \max_{a \in \mathcal{A}} E[R_{t+1}|D_t, A_{t+1} = a] & \text{w.p. } 1 - \epsilon \\
\text{uniform(}A) & \text{w.p. } \epsilon 
\end{cases}$$

Here we do overcome the drawback of greedy algorithm as action 1 will be explored even though action 2 has a higher expected return. The drawback is: if we are given the posterior of $\theta_3$ and $\theta_2$ (Fig. 1), $\theta_3$ is better than $\theta_2$ almost for sure, so exploring $\theta_2$ with probability $\epsilon$ is quite a waste of effort.
1.1.3 Boltzmann (smarter dithering):

To improve dithering, we sample action from an exponential distribution where $\theta$’s with low expected rewards are explored exponentially less frequently:

$$A_{t+1} \sim \exp\left(\frac{1}{\epsilon}E[R_{t+1}|D_t, A_{t+1} = a]\right)$$

It won’t waste too much effort on ”bad-for-sure” actions. However, it still treats less explored $\theta$’s unfairly (like greedy) if their expected reward is mistaken to be low in the early game.

1.1.4 Upper confidence bound (UCB):

Alternatively, we can use higher end of the confidence interval to decide what action to choose.

In Fig. 1, $\theta_1$ is not well explored, so despite a lower expectation, it gives higher upper confidence bound than the rest. We will keep choosing action 1 until we have more confidence on how bad $\theta_1$ really is.

The quantitative definition for UCB can have many variations. Here we use:

$$U_i(k) = (1 - 1/l)^{th} \text{quantile}$$

and the choice of action would be:

$$A_{t+1} = \arg \max_{a \in A} U_i(a)$$

In this way, as $l$ gets larger, we will compare higher quantiles. It encourages us to occasionally return to a bad $\theta$ just in case it might be good.

1.1.5 Thompson sampling (TS):

Here we first create a conjecture for each action according to the data through all episodes (up to $l^{th}$ episode) by sampling:

$$\hat{\theta}_{t,k} \sim \text{beta}(\alpha_{t,k}, \beta_{t,k})$$

and then choose:

$$A_{t+1} = \arg \max_{a \in A} \hat{\theta}_{t,k}$$

In Fig. 1, this means we will almost always choose action 3 (red), almost never choose action 2 (green) and occasionally choose action 1 (blue), which makes sense.

1.2 Summarized in more generic terms

We have an environment that determines the ”mapping” (more precisely, a distribution of mapping) from action to reward: $A_{t+1} \xrightarrow{E} R_{t+1}$ where $E$ is defined by $\theta \in \mathbb{R}^k$. Denote expected reward as:

$$\hat{r}(a, \theta) = E[R_{t+1}|\theta, A_t = a]$$

then we can summarize UCB and TS as follows:

1.2.1 UCB:

1. Construct confidence set $\Theta_t \subseteq \mathbb{R}^k$ based on $D_t$ (the details can be left to some subroutines)

2. Based on $u_l(a) = \max_{\theta \in \Theta_t} \hat{r}(a, \theta)$, choose $A_{t+1} \in \arg \max_{a \in A} u_l(a)$
1.2.2 TS:
1. Sample $\hat{\theta} \sim P(\theta \in \bullet | D_t)$
2. Choose $A_{t+1} \in \arg \max_{a \in A} \bar{r}(a, \hat{\theta})$

1.3 How good they are:
1.3.1 UCB:
Let $A^* = \arg \max_{a \in A} \bar{r}(a, \theta)$, then
\[
\bar{r}(A^*, \theta) - \bar{r}(A_{t+1}, \theta) = \bar{r}(A^*, \theta) - u_t(A_{t+1}) + u_t(A_{t+1}) - \bar{r}(A_{t+1}, \theta) \\
\leq \bar{r}(A^*, \theta) - u_t(A^*) + u_t(A_t + 1) - \bar{r}(A_{t+1}, \theta)
\]
where the second line follows as $A_{t+1}$ is chosen to maximize $u_t$. The first term is a ‘pessimism term’ (since expected reward is lower than UCB with high probability). The second term is confidence width, which will go to zero if the optimal action is sampled for sufficient times.

This interpretation of terms should be very reminiscent of RTVI. This is the gist of UCB algorithms and many papers are often about different ways to quantify the convergence rate of the confidence width.

1.3.2 TS:
We can consider upper confidence bound as defined above but on what data we get by following TS algorithm.

Lemma 1.
\[
E[u_t(A_{t+1})] = E[E[u_t(A_{t+1})|D_t]] \quad \text{(since inner expectation should be same as } E[u_t(A^*)|D_t]) \\
= E[u_t(A^*)]
\]

Applying the lemma, we have
\[
E[\bar{r}(A^*, \theta) - \bar{r}(A_{t+1}, \theta)] = E[\bar{r}(A^*, \theta) - u_t(A_{t+1})] + E[u_t(A_{t+1}) - \bar{r}(A_{t+1}, \theta)] \\
= E[\bar{r}(A^*, \theta) - u_t(A^*)] + E[u_t(A_{t+1}) - \bar{r}(A_{t+1}, \theta)]
\]
where the first term is a pessimism term $\leq 0$ (for sure since we are taking expectation) and second term is confidence width (going to zero as we sample).

We would want to assign weights to environments properly so as to minimize expected ‘regret’.
For the Bernoulli bandit case, the regret would be:
\[
Regret(L) = E[\sum_{l=1}^{L}(\bar{V}_* - \bar{V}_{\pi_l})] \\
\leq \hat{O}(\sqrt{KL})
\]
where $\pi$ and $\pi_l$ refer to the optimal policy and our policy at episode $l$, respectively.
Here, the main difference between UCB and TS is: TS allows us to say something about the expectation of regret, whereas UCB bounds the regret inside the expectation to also obey the inequality with high probability.
Since the analysis for TS algorithm also uses involves upper confidence bound, most analysis techniques of those UCB papers can apply for TS as well. The various definitions of UCB in the papers are very restrictive to the particular problems (hard to be statistically and computationally efficient), whereas TS would in some sense be more robust.