1 Multi Armed Bandit

Multi Armed Bandit(MAB) is a classic reinforcement learning problem that exemplifies the exploration–exploitation tradeoff dilemma. The name comes from imagining a gambler at a row of slot machines (sometimes known as "one-armed bandits"), who has to decide which machines to play, how many times to play each machine and in which order to play them, and whether to continue with the current machine or try a different machine. The multi-armed bandit problem also falls into the broad category of stochastic scheduling. Let's make an example to introduce the topic.

Example - Beer selection problem Suppose to enter in a newly opened brewery. We are allowed to choose among a set of available beers. The brewery have a rule. You can order a new beer only if you finished the previous one. After each beer, you assign a mark from 1 to 10 according to how much you liked it. It might happen that the value you assign to a beer varies. For example, one beer was expired or it had a production defect. We have two goals

- Find the beer you like the most
- Don't get wasted tasting beers

We can transpose this problem into an MDP. We can describe the value of each action(drinking a beer) in every state with a action-value function $Q(s, a)$. When we enter the brewery we don' know the Q function. We have to learn it **online**[1]. Every online decision problem make us face a fundamental choice

- **Exploration** gather more information from unexplored/less explored options. Choose a random beer to try something new. The new beer can be a revelation or be disgusting
- **Exploitation** select the option we consider to be the best one so far. Choose a type of beer we are sure we like

This balance is influenced by many factor. One of these is the time-horizon of our problem. Depending on how much we are far-sighted we might make some sacrifice in the short-term to gain more in the future. With **infinite time horizon** we have infinite steps(we are going several time to the brewery). We want to gather enough information to find the best overall decision. With **finite time horizon** we have a limited number of steps(We only go one time to the brewery). We want to minimize the short-term loss due to uncertainty. Usually MAB works with finite time horizon problems.

[1]Online means that we estimate the Q function interleaved with data gathering. In our case after every beer we try to estimate the action-value function

Example - Dilemma on real application To better visualize the exploration-exploitation dilemma, we can make some more examples.

- Clinical Trial
  - Exploration: Try new treatments
  - Exploitation: Choose the treatment that provides the best results
- Slot machine (a.k.a. one-armed bandit) selection
  - Exploration: Try all the available slot machines
  - Exploitation: Pull the one which provided you the highest payoff so far
- Game Playing
  - Exploration: Play an unexpected move
  - Exploitation: Play the move you think is the best
- Oil Drilling
  - Exploration: Drill at an unexplored location
  - Exploitation: Drill at the best known location

1.1 MAB problem

There are different types of MAB. They are fully characterized by the type of **reward**

- **Deterministic** we have a single value for the reward for each arm(We get the same reward from the same type of beer)
- **Stochastic** the reward of an arm is drawn from a distribution which is stationary over time(We get different rewards from the same type of beer)
- **Adversarial** an adversary chooses the reward we get from an arm at a specific round, knowing the algorithm we are using to solve the problem

We have seen in the previous chapters, how we can use MDPs to model a problem. It would be nice if we could map a MAB problem to a MDP. It turns out that we can see the Multi-Armed Bandit setting as a specific case of an MDP where

- S set of states. We have a single state, $S = \{s\}$
- A set of actions. Also called arms, $A = \{a_1, \ldots, a_N\}$
- P state transition probability matrix. $P(s|a_i, s) = 1$, $\forall a_i$
- R reward function. $R(s, a_i) = R(a_i)$

- γ discount factor. We have a finite time horizon, $\gamma = 1$
- μ^0 set of initial probabilities. $\mu^0(s) = 1$

In our case we have all the ingredients to build a MDP representation, except the reward function R. We have already seen how we can still solve the problem using RL using model-free control algorithms. Now let's focus on finding a policy(policy improvement) for our problem. We have seen several methods, for example ϵ-greedy.

$$\pi(a_i|s) = \begin{cases} 1-\epsilon & \text{if } \hat{Q}(a_i|s) = \max_{a\in A} \hat{Q}(s|s) \\ \frac{\epsilon}{|A|-1} & \text{otherwise} \end{cases} \tag{1}$$

We perform the greedy action with probability $1-\epsilon$, and a random action between the others with probability $\frac{\epsilon}{|A|-1}$. This approach converges to the optimal policy, only if ϵ converges to zero over time.
Another approach is the **softmax**

$$\pi(a_i|s) = \frac{e^{\frac{\hat{Q}(a_i|s)}{\tau}}}{\sum_{a\in A} e^{\frac{\hat{Q}(a|s)}{\tau}}} \tag{2}$$

Weights the actions according to its estimated value $\hat{Q}(a|s)$. τ is a parameter that decreases over time. Even if these algorithms converge to the optimal choice, we do not know how much we lose during the learning process.

1.1.1 Stochastic MAB

The definition of MAB problems with MDP is useful but is not the only one. A Multi-armed Bandit problem can be seen as a tuple $\langle \mathcal{A}, \mathcal{R} \rangle$

- $\mathcal{A}$ is a set of N possible arms(choices)
- $\mathcal{R}$ is an set of unknown random variable $\mathcal{R}(a_i)$, where $E[\mathcal{R}(a_i)] = R(a_i)$

The process we consider is the following. At each round t the agent selects a single arms a_{i_t}. Then the environment generates a reward r_{i_t} drawn from $\mathcal{R}(a_{i_t})$. Finally the agent updates its information by means of a history h_t(pulled arm and received reward).

The final objective of the agent is to maximize the cumulative reward over a given time horizon T

$$\sum_{t=1}^{T} r_{i_t,t}$$

where $r_{i_t,t}$ is the realization of the reward for the arm a_{i_t} we choose for the turn. Possibly we also want to converge to the option with largest expected reward if one considers $T \to \infty$. We can redefine our objective as to minimize the reward we lost through acting non-optimally.

The objective function can be reformulated in the following way. Define the expected reward of the optimal arm a^* as

$$R^* = R(a^*) = \max_{a \in A} E[\mathcal{R}(a)]$$

At a given time step t, we select the action a_{i_t}, and we incur in a loss of $\mathcal{R}(a^*) - \mathcal{R}(a_{i_t})$. The reward is stochastic, so it can be useful to calculate the average loss of the algorithm

$$E[\mathcal{R}(a^*) - \mathcal{R}(a_{i_t})] = R^* - R(a_{i_t})$$

This difference can be called **regret**. We want to minimize the expected regret suffered over a finite time horizon of T rounds

Definition 1.1 (Expected pseudo regret)**.**

$$L_T = TR^* - E\left[\sum_{t=1}^{T} R(a_{i_t})\right]$$

The expected value is taken w.r.t. the stochasticity of the reward function and the randomness of the used algorithm. Note that the maximization of the cumulative reward is equivalent to the minimization of the cumulative regret.

We can reformulate the cumulative regret in another way. We define the average difference in reward between a generic arm a_i and the optimal one a^* as $\Delta_i := R^* - R(a_i)$. Then we define the number of times an arm a_i has been pulled after a total of t time steps as $N_t(a_i)$.

$$\begin{aligned} L_T &= TR^* - E\left[\sum_{t=1}^{T} R(a_{i_t})\right] \\ &= E\left[\sum_{t=1}^{T} R^* - R(a_{i_t})\right] \\ &= \sum_{a \in A} E[N_t(a_i)](R^* - R(a_{i_t})) \\ &= \sum_{a \in A} E[N_t(a_i)]\Delta_i \end{aligned}$$

Note how we rewrote the summation over time to a summation over arms. At each time step we choose an arm. If we choose the arm a_i $N_t(a_i)$ times we have that $T = N_t(a_1) + \cdots + N_t(a_N)$. Looking at formula we can see that to minimize the regret we want to minimize the number of times we select a sub-optimal arm.

We can find a lower bound of the regret

Theorem 1.1. *Given a MAB stochastic problem, any algorithm satisfies*

$$\lim_{T \to \infty} L_T \geq log(T) \sum_{a_i | \Delta_i > 0} \frac{\Delta_i}{KL(R(a_i), R(a^*))} \tag{3}$$

where $KL(R(a_i), R(a^*))$ is the Kullback-Leibler divergence between the two Bernoulli distributions $\mathcal{R}(a_i)$ and $\mathcal{R}(a^*)$. The important thing to notice is that the cumulative regret depends on the time horizon T. So we can't have a constant regret over all time steps. We can show that the Kullback-Leibler divergence is proportional to Δ^2. This means that the argument of the summation on the right hand side is proportional to $\frac{1}{\Delta_i}$. If we have a small Δ_i we have a higher lower bound and vice versa. This can explained by the fact that, if all the arms are very close to the optimal action, it will be difficult to choose which of the arm is the best.

Now let's discuss some algorithm which can came close to this lower bound. The simplest algorithm we can think of always select the action such that $a_{i_t} = \underset{a}{argmax} \hat{R}_t(a)$ where the expected reward for an arm is

$$\hat{R}_t(a_i) = \frac{1}{N_t(a_i)} \sum_{j=1}^{t} r_{i,j} \mathbf{1}(a_i = a_{i_j})$$

This algorithm is called **pure exploitation algorithm**. It might not converge to the optimal action. We can construct a counter example. Suppose to have Bernoulli returns. At the beginning, we choose the optimal arm, and it returns zero(note that the rewards are stochastic). If we get other samples, and we never choose the optimal action again, and one of the non-optimal function gets a reward of one, the optimal arm will never be chosen. To adjust the algorithm we need to consider the uncertainty corresponding to the $\hat{R}_t(a)$ estimate. One way to do that is by providing an explicit bonus for exploration.

There are two different formulations

- **Frequentist formulation** $R(a_1), \ldots, R(a_N)$ are considered as unknown parameters. A policy selects at each time step an arm based on the observation history
- **Bayesian formulation** $R(a_1), \ldots, R(a_N)$ are considered as random variables with prior distributions $f_1, \ldots, f_N$. A policy selects at each time step an arm based on the observation history and on the provided priors

Frequentist algorithms Let's start with the frequentist approach. The main take away of this approach is that the more we are uncertain on a specific choice, the more we want the algorithm to explore that option. Doing so, we might lose some value in the current round, but it might turn out that the explored action is the best one. An example is the upper confidence bound approach. Instead of using the empiric estimate we consider an upper bound $U(a_i)$ over the expected value $R(a_i)$. More formally, we need to compute an upper bound

$$U(a_i) := \hat{R}_t(a_i) + B_t(a_i) \geq R(a_i)$$

with high probability. The bound length $B_t(a_i)$ depends on how much information we have on an arm, for example the number of times we pulled that arm so far $N_t(a_i)$. Small $N_t(a_i) \rightarrow$ large $U(a_i)$ (the estimated value $\hat{R}_t(a_i)$ is uncertain). Large $N_t(a_i) \rightarrow$ small $U(a_i)$

(the estimated value $\hat{R}_t(a_i)$ is accurate). In order to set the upper bound we resort to a classical concentration inequality

Definition 1.2 (Hoeffding Bound)**.** *Let $X_1, \dots, X_t$ be i.i.d. random variables with support in $[0,1]$ and identical mean $E[X_i] =: X$ and let $\bar{X}_t = \frac{\sum_{i=1}^{t} X_i}{t}$ be the sample mean. Then*

$$P(X > \bar{X}_t + u) \leq e^{-2tu^2}$$

We will apply this inequality to the upper bounds corresponding to each arm

$$P(R(a_i) > \hat{R}_t(a_i) + B_t(a_i)) \leq e^{-2N_t(a_i)B_t(a_i)^2} \tag{4}$$

We want to find B_t. To do so, we fix the probability of the bound

$$e^{-2N_t(a_i)B_t(a_i)^2} = p$$

We solve to find $B_t(a_i)$

$$B_t(a_i) = \sqrt{\frac{-log(p)}{2N_t(a_i)}}$$

the probability p represents the probability of the expect value overcome the bound. If we perform many steps these probabilities sums up and almost surely the bound will be broken. To solve this problem, we can shrink p over time. For example the probability can be equal to $p = t^{-4}$

$$B_t(a_i) = \sqrt{\frac{2log(t)}{N_t(a_i)}} \tag{5}$$

This ensures to select the optimal action as the number of samples increases. $\lim_{t\to\infty} B_t(a_i) = 0 \Rightarrow \lim_{t\to\infty} U_t(a_i) = R(a_i)$
This algorithm is called UCB1. For each time step t we do three steps.

1. Compute
$$\hat{R}_t(a_i) = \frac{\sum_{i=1}^{t} r_{i,t}\mathbf{1}(a_i = a_{i_t})}{N_t}, \quad \forall a_i$$
2. Compute
$$B_t(a_i) = \sqrt{\frac{2log(t)}{N_t(a_i)}}, \quad \forall a_i$$
3. Play arm
$$a_{i_t} = \underset{a_i \in A}{argmax}\left(\hat{R}_t(a_i) + B_t(a_i)\right)$$

From that we can derive the specific upper bound of UCB1

Theorem 1.2 (UCB1 Upper Bound, Auer & Cesa-Bianchi 2002)**.** *At finite time T, the expected total regret of the UCB1 algorithm applied to a stochastic MAB problem is*

$$L_T \leq 8log(T) \sum_{i|\Delta_i>0} \frac{1}{\Delta_i} + \left(1 + \frac{\pi^2}{3}\right) \sum_{i|\Delta_i>0} \Delta_i$$

Remember that the lower bound we have found before is general for all the algorithms. Instead, for the upper bound, every algorithm has a different one. For UCB we have that the the two bounds depends on the same order, $log(T)$. This is very nice, the upper bound is good.

Bayesian algorithms The first Bayesian algorithm we see is **Thompson sampling**. It is a general Bayesian methodology for online learning. It is structured as follow. Consider a Bayesian prior for each arm $f_1, \ldots, f_N$ as a starting point. At each round t sample from each one of the distributions $\hat{r}_i, \ldots, \hat{r}_N$. Pull the arm a_{i_t} with the highest sampled value $i_t = \underset{i}{argmax}\{\hat{r}_i\}$. Update the prior of the pulled arm, incorporating the new information. Let's see in more detail this algorithm. The prior conjugate distributions are $Beta(\alpha, \beta)$. In particular we take $f_i(0) = Beta(1, 1)$ for each arm a_i. After we pulled the arm a_i, we get a Bernoulli reward. Then, we update the prior incorporating information gathered. After the update we still have a $Beta$ distribution. In detail we have,

- In the case of a success occurs $f_i(t+1) = Beta(\alpha_t + 1; \beta_t)$
- In the case of a failure occurs $f_i(t+1) = Beta(\alpha_t; \beta_t + 1)$

The upper bound of the algorithm is

Theorem 1.3 (Thompson Sampling Upper Bound, Kaufmann & Munos 2012)**.** *At time T, the expected total regret of Thompson Sampling algorithm applied to a stochastic MAB problem is*

$$L_T \leq O\left(\sum_{i|\Delta_i>0} \frac{\Delta_i}{KL(\mathcal{R}(a_i), \mathcal{R}(a^*))}(log(T) + log(log(T)))\right)$$

Note that this upper bound have the same order and constant of the lower bound. This means that we can't build a better bound.

1.1.2 Adversarial MAB

With adversarial MAB the reward is no longer given by the environment, but is returned by an adversary player. A Multi-Armed Bandit Adversary setting is a tuple $\langle \mathcal{A}, \mathcal{R} \rangle$

- $\mathcal{A}$ is a set of N possible arms (choices)
- $\mathcal{R}$ is a reward vector for which the realization $r_{i,t}$ is decided by an adversary player at each turn

The process we consider is the following

- At each time step t the agent selects a single arms a_{i_t}
- At the same time the adversary chooses rewards $r_{i,t}$
- The agent gets reward $r_{i,t}$
- The final objective of the agent is to maximize the cumulative reward over a time horizon T

$$\sum_{t=1}^{T} r_{i_t,t}$$

We cannot compare the accumulated regret we gained with the optimal one. Moreover, the fact that there is an adversary choosing the regret does not allow to use deterministic algorithms(e.g., UCB). This is due to the fact that, the return is no longer characterized by a random variable, and so it's not stochastic. Instead, we can use the **weak regret**.

Definition 1.3 (Weak regret)**.**

$$L_T = \max_i \left(\sum_{t=1}^{T} r_{i,t} \right) - \sum_{t=1}^{T} r_{i_t,t}$$

We are comparing the policy with the best constant action. As we did before, we can construct a lower bound for all the algorithms

Theorem 1.4 (Minimax Lower Bound)**.** *Let sup be the supremum over all distribution of rewards such that, for $i \in \{1, \dots, N\}$ the rewards $r_{i,1}, \dots, r_{i,T}$ and $r_{i,j}$ are i.i.d. and let inf be the infimum over all forecasters. Then*

$$inf\ sup\ E[L_T] \geq \frac{1}{20}\sqrt{TN}$$

where the expectation is taken with respect to both the random generation rewards and the internal randomization of the forecaster

The first non-deterministic algorithm we see is **EXP3**. This algorithm derives from the softmax algorithm. The probability of choosing an arm(policy) is

$$\pi_t(a_i) = (1-\beta)\frac{w_t(a_i)}{\sum_j w_t(a_j)} + \frac{\beta}{N} \tag{6}$$

where w is a weight calculated as

$$w_{t+1}(a_i) = \begin{cases} w_t(a_i) e^{\eta \frac{r_{i,t}}{\pi_t(a_i)}} & \text{if } a_i \text{ has been pulled} \\ w_t(a_i) & \text{otherwise} \end{cases}$$

and β is a constant term.
For this algorithm, the upper bound is

Theorem 1.5 (EXP3 Upper Bound)**.** *At time T, the expected weak regret of EXP3 algorithm applied to an adversarial MAB problem with* $\beta = \eta = \sqrt{\frac{Nlog(N)}{(e-1)T}}$ *is*

$$E(L_T) \leq O\big(\sqrt{TNlog(N)}\big)$$

where the expectation is taken with respect to both the random generation rewards and the internal randomization of the forecaster

1.2 Generalized MAB problem

In the beer selection problem you now have a set of breweries. Each night your friend decides which one to pick. Once you are in a specific brewery you are free to pick a beer of your choice. Is this a Bandit problem? If we fix the brewery, it is a stochastic MAB. Since we do not control the transition from one brewery to another, we can use independent MAB techniques over each one of the breweries. This change in environment(different breweries) needs a new type of MAB called **contextual MAB**. There exist other type of MAB like

- **Budget-MAB** we are allowed to pull arms until a fixed budget elapses, where the pulling action incurs in a reward and a cost
- **Continuous Armed Bandit** we have a set of arms A which is not finite

We can also have other sequential decision settings

- **Arm identification problem** we just want to identify the optimal arm with a given confidence, without caring about the regret
- **Expert setting** we are also allowed to see the rewards of the not pulled arms each turn (online learning problem)