**IIIT Hyderabad** **Scribed By: Kushagra Chandak**
**-** **Aagam Shah**
**Topics in Machine Learning (CSE975)** **Lecture #7**
**Instructor: Dr. Naresh Manwani** **August 29, 2017**

---

**Outline.** *We introduce Reinforcement Learning with the Multi Arm Bandit (MAB) Learning algorithm and define its objective. Finally we give a brief introduction to the Upper Confidence Bound (UCB) algorithm.*

# 1 Reinforcement Learning

- Reinforcement learning (RL) is learning about what action to take when given a situation (state), so that we maximize our reward in the long run.
- In reinforcement learning, the learner does not receive any direct supervision. For example, it is not told which is a good action in a given state. Instead, it receives rewards or penalties for choosing an action in a given state. And the learner must discover which actions yield the most reward by trying them. Thus, the learning happens by interacting with environment and observing the results of these interactions.
- In the most interesting and challenging cases, actions may affect not only the immediate reward, but also the next situation and, through that, all subsequent rewards.
- This interestingly also mimics the fundamental way in which humans learn. As humans, we have a direct sensori-motor connection to our environment, meaning we can perform actions and witness the results of these actions on the environment.

## 1.1 An example : Chess

- In the popular board game, Chess, each player can perform a certain set of actions in a particular state. On performing any of these actions, the state of the board changes, also affecting the possible set of actions (from this state).
- Each player performs an action to try and get the best cumulative reward (which might be in the form of capturing an opponent's piece or some other form) with the final goal of capturing the king.
- Each such reward will be different for different action/state combination and will also depend on the player's and his/her opponent's previous actions.

Here we consider a simpler setting called the "Multi Armed Bandit Problem", which has only one state.

## 2 Multi Armed Bandit Learning

Multi-armed bandit learning is motivated from the famous old styled gambling machines (with arms to be pulled). A person want to maximize the money (reward) obtained by successively playing the gamble machines (the "arms" of the bandits). Also, the rewards corresponding to each machine are unknown ahead of time and each machine has a different and unknown distribution law for rewards. The machines have been set up such that successive plays of the same machine yield rewards that are independently and identically distributed.

### 2.1 A Practical Application: Ethical Clinical Trials

For a given disease, let there be $K$ kinds of treatment available. A medical practitioner would want to evaluate all $K$ possible treatments and find out which one is the most effective. The immediate reward here is the wellness of a patient.

### 2.2 The Algorithm

Let $n$ be the number of arms, $t$ be the trial number and $x_i^t$ be the reward obtained on pulling arm $i$ on trial $t$. Then a generic MAB algorithm works as follows.

**Algorithm 1** MAB algorithm

1: **for** $t = 1...T$ **do**
2: Select an arm $i_t \in \{1...n\}$
3: Environment selects a reward vector $\mathbf{x^t} = (x_1^t, ..., x_n^t) \in [0,1]^n$
4: Algorithm observes the selected reward $x_{i_t}^t$

Let $(i_1, x_{i_1}^1), ..., (i_T, x_{i_T}^T)$ be the sequence of arms played and corresponding rewards in $T$ trials. Given a sequence of arms played and rewards earned in $t-1$ trails, the MAB algorithm $A$ returns an arm $i_t \in \{1, ..., n\}$ to play in the $t^{th}$ trial. Therefore, this arm $i_t$, in general depends on the rewards $x_{i_1}^1, ..., x_{i_{t-1}}^{t-1}$.

### 2.3 The Objective

The objective of the MAB algorithm is to maximize the sum of rewards also called gain. Gain of $A$ on a sequence of reward vectors is given by:

$$G_T(A) = G_{(\mathbf{x}^1,...,\mathbf{x}^T)}[A] = \sum_{t=1}^{T} x_{i_t}^t$$

Gain of an individual arm $i$ over a sequence of reward vectors is given by:

$$G_T(i) = \sum_{t=1}^{T} x_i^t$$

The regret of an algorithm A over a sequence of reward vectors $(\mathbf{x}^1, \ldots, \mathbf{x}^T)$ is given by:

$$R_{(\mathbf{x^1},...,\mathbf{x^T})}(A) = \max_{i \in \{1,...,n\}} G_T(i) - G_T(A)$$

Thus, the objective of MAB algorithm $A$ is to minimize the regret $R_T(A)$.

# 3 Different MAB settings

## 3.1 Stochastic MAB

Each arm $i \in \{1, ..., n\}$ is associated with an unknown probability distribution $q_i$ on the space of rewards. At each trial $t$, reward $x_i^t$, $t = 1 \dots T$ for arm $i$ are drawn independently from $q_i$. Let $\mu_i$ denote the mean reward for arm $i$, that is $\mu_i = \mathbb{E}_{x_i \sim q_i}[x_i]$. $\mu_i$'s are unknown. Let $\mu^* = \max_{i \in \{1,...,n\}} \mu_i$ be the mean reward of the optimal arm. Let $\Delta_i = \mu^* - \mu_i$ be the difference between the mean reward of an optimal arm and the mean reward of arm $i$. Then expected regret is given as:

$$R_T(A) = \mathbb{E}_{(\mathbf{x^1}, ..., \mathbf{x^T})} \left[ \max_{i \in \{1,...,n\}} \sum_{t=1}^{T} x_i^t - \sum_{t=1}^{T} x_{i_t}^t \right]$$

## 3.2 Pseudo Regret

Since the expected regret is difficult to handle, we minimize pseudo regret given as follows.

$$\widetilde{R}_T(A) = \max_{i \in \{1,...,n\}} \mathbb{E}_{(\mathbf{x^1}, ..., \mathbf{x^T})} [\sum_{t=1}^{T} x_i^t - \sum_{t=1}^{T} x_{i_t}^t]$$

Note that the pseudo-regret is upper bounded by the expected regret, and therefore an upper bound on the pseudo-regret does not imply an upper bound on the expected regret (i.e. an upper bound on the pseudo regret is a weaker statement than an upper bound on the expected regret). When there is randomization in the algorithm, we also take expectation over this randomness in defining the above quantities. In order to minimize the pseudo-regret, we essentially try to find an optimal arm, i.e. an arm with highest mean reward $\mu^*$. Various strategies have been proposed to do this; and in most cases, we make use of sample means of the rewards obtained with different arms to estimate the true mean $\mu_i$.

## 3.3 Adversarial MAB *(to be studied in a later lecture)*

No probabilistic assumptions on rewards $x_i^t$. Rewards can be generated by an adversary. We will talk more about this in the upcoming classes.

# 4 Action Value Methods

In the action value methods, we basically maintain average reward corresponding to each action (arm). We then use these average rewards to define the policy to choose an arm.
$\mu_i$ are also termed as action values. $\hat{\mu}_i^t$ is the empirical estimate of $\mu_i$ till time $t$ and is called the estimate of action value $\mu_i$.

$$\hat{\mu}_i^t = \frac{\sum_{s=1}^{t-1} x_i^s \mathbb{I}_{\{i_s = i\}}}{\sum_{s=1}^{t-1} \mathbb{I}_{\{i_s = i\}}} \tag{1}$$

### 4.1 Efficient implementation of $\hat{\mu}_i^t$

Given $\hat{\mu}_i^{t-1}$, $N_i^{t-1}$ and $x_{i_t}^t$, we can find $\hat{\mu}_i^t$ as follows.

$$\begin{aligned}
\hat{\mu}_i^t &= \frac{1}{\sum_{s=1}^{t} \mathbb{I}_{\{i_s=i\}}} \sum_{s=1}^{t} x_i^s \mathbb{I}_{\{i_s=i\}} \\
&= \frac{1}{\sum_{s=1}^{t-1} \mathbb{I}_{\{i_s=i\}} + \mathbb{I}_{\{i_t=i\}}} \left[ x_i^t \mathbb{I}_{\{i_t=i\}} + \sum_{s=1}^{t-1} x_i^s \mathbb{I}_{\{i_s=i\}} \right] \\
&= \frac{1}{N_i^{t-1} + \mathbb{I}_{\{i_t=i\}}} \left[ x_i^t \mathbb{I}_{\{i_t=i\}} + \hat{\mu}_i^{t-1} \sum_{s=1}^{t-1} \mathbb{I}_{\{i_s=i\}} \right] \\
&= \frac{1}{N_i^{t-1} + \mathbb{I}_{\{i_t=i\}}} \left[ x_i^t \mathbb{I}_{\{i_t=i\}} + \hat{\mu}_i^{t-1} \sum_{s=1}^{t} \mathbb{I}_{\{i_s=i\}} - \hat{\mu}_i^{t-1} \mathbb{I}_{\{i_t=i\}} \right] \\
&= \hat{\mu}_i^{t-1} + \frac{\mathbb{I}_{\{i_t=i\}}}{N_i^{t-1} + \mathbb{I}_{\{i_t=i\}}} [x_i^t - \hat{\mu}_i^{t-1}]
\end{aligned}$$

where $N_i^{t-1} = \sum_{s=1}^{t-1} \mathbb{I}_{\{i_s=i\}}$ is the number of times arm $i$ is pulled in $t-1$ trials.

### 4.2 The Exploitation-Exploration Dilemma

The main issue here is to trade-off between exploration and exploitation. If an agent has tried a certain action in the past and got good rewards, then its action value so far is good. Then choosing this action based on its past performance is called *exploitation*. On the other hand, there may be other action (arm) which can have better value (mean reward) than the current best. Thus, it is desirable to try other possibilities in the hope of producing better rewards. This process if called *exploration*.

We can't exploit all the time as it may lead us to choose suboptimal action (arm) all the time as we don't know whether there is an arm which is better than the current best. On the other hand, we cant explore always as there are many suboptimal arms and exploring them all will again lead us to poor gains.

Thus, we need to make a balance between exploration and exploitation. The most common way to achieve a nice balance is to try a variety of actions while progessively favouring those that stand out as producing the most reward. In the next sections, we look and analyze various simple approaches to solve the Multi-Armed Bandit problem.

## 5 Greedy Approach

We begin with the simplest method called Greedy Approach where we don't *explore* but only *exploit*. The idea is to make best decision given current information and not look for any new information.

We know that the true value of an action is the mean reward received when that action is selected. Let $N_i^t$ be the number of times $i$-th arm is selected in $t$ trials. Then the mean reward is

$$\hat{\mu}_i^t = \frac{1}{N_i^t} \sum_{s=1}^{t} x_i^s \mathbb{I}_{\{i_s=i\}}$$

If $N_i^t = 0$, we define $\mu_i^t$ instead as some default value like $\mu_i^t = 0$. As $N_i^t \longrightarrow \infty$, $\mu_i^t$ converges to $\mu_i$ by *law of large numbers*. In greedy approach, we pick an arm at trial $t$ with highest current action value (average reward).

$$i_t = \arg \max_{i \in \{1,\ldots,n\}} \hat{\mu}_i^{t-1}$$

An advantage of this method is that it's very simple to understand and implement. The biggest disadvantage is that this method always exploits current knowledge to maximize immediate reward and does not explore other arms (actions) which might be potentially optimal.

## 5.1 $\epsilon$-greedy approach

The exploration-exploitation dilemma is the biggest problem in the stochastic MAB setting. There's always some trade-off between exploiting the current knowledge to pick the arm that has highest reward (as in greedy approach), and exploring further the other arms to get a better reward in future. Since Greedy Approach, as described above, only *exploits*, we need a better strategy and the solution is to perform exploration and exploitation simultaneously, which is done by the $\epsilon$-greedy heuristic.

The $\epsilon$-greedy algorithm is similar to the greedy one, except it has an extra parameter $\epsilon$ which is sometimes also called exploration probability. The idea is very simple. First, pick a parameter $0 < \epsilon < 1$. Then, at each step greedily pick the arm with highest empirical mean reward with probability $1 - \epsilon$ (*exploitation phase*), and pick a random arm with probability $\epsilon$ (*exploration phase*). If $\epsilon$ is a function of time (instead of a constant), we can get logarithmic bound instead of a linear bound. If $\epsilon_t = n/d^2t$ then it can be proved that the regret grows logarithmically like $n \log T/d^2$ provided $0 < d < \min_{i \neq i^*} \Delta_i$ and algorithm with this $\epsilon$ performs well in practice.

**Algorithm 2** $\epsilon$-greedy algorithm

**Input:** $\epsilon \in (0, 1)$
**for** $t = 1$ to $T$ **do**
  Toss a coin with probability of success $\epsilon$.
  **if** *success* **then**
    explore: choose an arm uniformly at random
  **else**
    exploit: choose the arm with the highest average reward so far

### 5.1.1 Remarks

As the number of time steps increases, each arm would get selected infinitely many times (thanks to exploration phase) and the sample estimates of mean rewards will converge to the actual mean rewards (*by law of large numbers*). Thus, probability of selecting the optimal action converges to greater than $1 - \epsilon$. In $\epsilon$-greedy, the exploration schedule does not depend on the history of the observed rewards. Whereas it is usually better to *adapt* exploration to the observed rewards.

# 6 Upper Confidence Bound (UCB) Algorithm (to be continued in the next lecture)

## 6.1 Brief introduction

So far, we have seen algorithms which either selects the arm with highest average reward or sometimes decides to explore and pick an option that currently doesn't seem the best. But they don't keep track of *how much they know* about any of the arms available to them. Only the rewards are recorded by these algorithms. We might end up *less* (or no) exploring the options and that can give us sub optimal rewards. In UCB algorithm we keep track of how much we know about each arm in terms of confidence intervals around the mean rewards.

Basic idea: Suppose there are 2 suboptimal actions, but one of them has less uncertainty than the other (perhaps one is sampled more number of times than the other). More certain the action, less likely is its value better than the optimal (greedy) choice. The estimate for other action is very uncertain and the chances that its true value is better than the optimal action are quite high. So clearly, it is sensible to explore the second choice than the first one. This exploration is done by estimating a confidence interval for each action's value and the action with highest upper limit of confidence interval is selected. This is the basic idea behind UCB which encourages exploration of actions that are more uncertain. It achieves regret that grows logarithmically with the number of actions taken.

As an arm is sampled more and more, the confidence interval around $\mu_i$ (sample mean) becomes tighter (because of the $\sqrt{\frac{\alpha lnt}{2N_i^{t-1}}}$ factor), giving more accurate estimate of the true mean reward.

**Algorithm 3** UCB algorithm

1: Parameter $\alpha > 0$
2: **for** $t = 1...T$ **do**
3: Select an arm $i_t \in \arg\max_{i \in \{1...n\}}[\hat{\mu}_i^{t-1} + \sqrt{\frac{\alpha \ln t}{2N_i^{t-1}}}]$
4: Algorithm observes reward $x_{it}^t$

# References


[1] Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto.

[2] http://profs.sci.univr.it/farinelli/courses/ddrMAS/slides/Bandit.pdf

[3] Regret Analysis of Stochastic and Non-stochastic Multi-armed Bandit Problems by Sebastian Bubeck and Nicolo Cesa-Bianchi