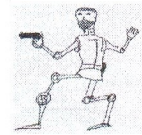
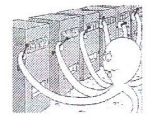
Credits: Francesco Trovò

Machine Learning
Multi-Armed Bandit

Lorenzo Bisi

23 May, 2019



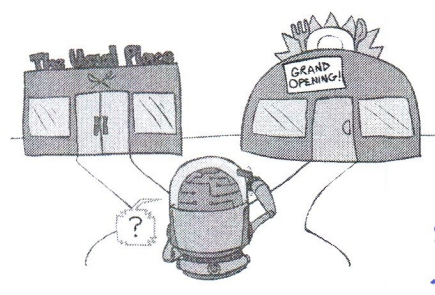


Outline

Outline

Traditional Motivating Example

Restaurant selection problem:



Should we go to the usual place or should we try something new? This is an usual problem in an interactive environment. (in an on-line learning, i.e. when we have to interact with the environment to get new infos)

Motivating Example

Beer selection problem:

- We enter a newly opened brewery
- We are allowed to chose among a set of available beers (one at a time)
- 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 (e.g., different bottles might have slightly different tastes)
- Your goal is twofold:
  - Find the beer you like the most
  - You do not want to get drunk in doing that (minimize the order of beers you do not like)

Exploration/Exploitation Dilemma

- The main point is that we are not sure about the value of each action $Q(a|s)$
- **Online** decision making make us face a fundamental choice:
  - **Exploration**: gather more information from unexplored/less explored options
  - **Exploitation**: select the option we consider to be the best one so far
- Depending on how much we are far-sighted we might make some sacrifice in the short-term to gain more in the future
  - Infinite time horizon: we want to gather enough information to find the best overall decision
  - Finite time horizon: we want to minimize the short-term loss due to uncertainty

If we choose only one of the two options we are never going to reach an optimal solution: if we always explore then we never bet on a single choice (we'll always try something new), while if we always exploit we have limited informations (we end up with a suboptimal solution)

Example of Real Applications

- 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

Common Approaches in RL

- ε-greedy:

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

  - Perform the greedy action except for a small amount of times
  - Does not achieve the optimal policy (if we keep $\varepsilon > 0$, to obtain the optimal policy: $\varepsilon \to 0$)
- Softmax (Boltzmann distribution):

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

  - Weights the actions according to its estimated value $\hat{Q}(a|s)$
  - τ is a temperature parameter which decreases over time

We use a fully stochastic policy: we take a random action, the probability for each action is weighted in a way that actions which have an higher function are selected more

by decreasing this parameter we're making the policy more deterministic (if $\tau \to 0$ then we reduce to the usual "max")

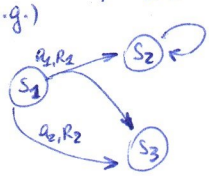
Even if these algorithms converge to the optimal choice, we do not know how much we lose during the learning process

In Reinforcement learning we focus on reaching the optimal. In Multi Armed Bandit we focus on how much we lose during the process.

Outline

From MDP to MAB

Reinforcement Learning has a dynamic: (e.g.)



Multi Armed Bandit has not:

(we clearly have some simplification w.r.t. the RL, however we still have, for instance, the exploration/exploitation dilemma)

We can see the Multi-Armed Bandit setting as a specific case of an MDP (Markov Decision Process)

- $\mathcal{S}$ is a set of states $\rightarrow$ single state $\mathcal{S} = \{s\}$
- $\mathcal{A}$ is a set of actions $\rightarrow$ **arms** $\mathcal{A} = \{a_1, \dots, a_N\}$
- P is a state transition probability matrix $\rightarrow P(s|a_i, s) = 1, \forall a_i$ — we're always going back to the starting state (whatever action we take)
- R is a reward function $\rightarrow R(s, a_i) = R(a_i)$ — it says what is the reward if we take an action
- γ is a discount factor $\rightarrow$ finite time horizon $\gamma = 1$
- μ^0 is a set of initial probabilities $\rightarrow \mu^0(s) = 1$

Different Kind of MAB

The only thing we need to have a full definition of the problem is the characterization of the **reward** $R(a_i)$:

- Deterministic: we have a single value for the reward for each arm (trivial solution) — In this case the solution is **trivial**: we try each action once and then we determine which is the best
- Stochastic: the reward of an arm is drawn from a distribution which is stationary over time, e.g., $R(a_i) = \mu_i$ if the distribution is a Bernoulli $Be(\mu_i)$ (the $R(a_i)$ is equal to the mean of the distribution in the $Be(\cdot)$ case)
- 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 — the adversary knows the algorithm we're using, however it doesn't know which is the realization of our strategy (notice that in this case the best action may change over time)

Usually we **don't** know the distribution: we just know that the mean of the rewards is the mean of the distribution and so we want to choose the arm with the higher mean (of course we won't just look at the mean, we also have to understand what we lose during the process)

Outline

Multi-Armed Bandit Setting

A Multi-Armed Bandit problem is a tuple $\langle \mathcal{A}, \mathcal{R} \rangle$

action space and reward function

- $\mathcal{A}$ is a set of N possible arms (choices)
- $\mathcal{R}$ is an set of **unknown** random variable $\mathcal{R}(a_i)$ ($\mathbb{E}[\mathcal{R}(a_i)] = R(a_i)$), assume $\mathcal{R}(a_i) \in [0, 1]$

The process we consider is the following:

- At each round t the agent selects a single arms a_{i_t}
- The environment generates a reward $r_{i_t,t}$ drawn from $\mathcal{R}(a_{i_t})$
- The agent updates her information by means of a history h_t (pulled arm and received reward)

Objective of a MAB Algorithm

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 $T \to \infty$

Alternative Goal: Minimize the Regret

the regret of not having taken the optimal action

The objective function can be reformulated in the following way:

- Define the expected reward of the optimal arm a^* as:

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

- 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 loss due to the fact that the chosen action may be a suboptimal one

- On average (w.r.t. the reward stochasticity) the algorithm loses:

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

Expected Pseudo Regret Definition

We want to minimize the expected regret suffered over a finite time horizon of T rounds

Expected Pseudo Regret

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

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

Another Regret Definition

Another way of reformulating the cumulate regret is:

- Define the average difference in reward between a generic arm a_i and the optimal one a^* as $\Delta_i := R^* - R(a_i)$
- Define the number of times an arm a_i has been pulled after a total of t time steps as $N_t(a_i)$

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

i.e., we want to minimize the number of times we select a suboptimal arm

Lower Bound for Stochastic MAB

- The definition of regret in terms of Δ_i implies that any algorithm performance is determined by the **similarity** among arms
- The more the arms are similar, the more the problem is difficult

It is possible to show the following:

Theorem (MAB lower bound, Lai & Robbins 1985)
Given a MAB stochastic problem any algorithm satisfies:

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

where $KL(R(a_i), R(a^*))$ is the Kullback-Leibler divergence between the two Bernoulli distributions $\mathcal{R}(a_i)$ and $\mathcal{R}(a^*)$

KL is a measure of similarity. if the two distribution are very similar → KL will be very small (small denominator, higher lower bound, higher regret) → similar arms lead to some problems (difficulties)

Pure Exploitation Algorithm

- Always select the action s.t. $a_{i_t} = \arg\max_a \hat{R}_t(a)$ where the expected reward for an arm is:

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

select the action for which the sampled mean of the rewards computed so far is maximal (we're choosing the action that so far proved to be the optimal)

- This strategy is the one which is trying to minimize the cumulated regret in a straightforward way
- Might not converge to the optimal action
- We are not considering the uncertainty corresponding to the $\hat{R}_t(a)$ estimate

due to the fact that we only saw a limited amount of samples

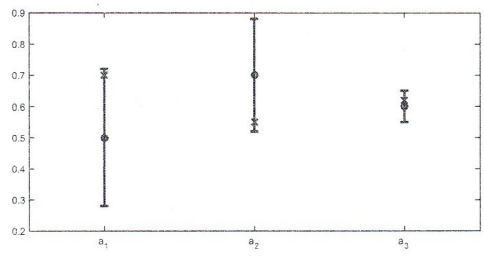
Hint: We need to provide an explicit bonus for **exploration**

Two Formulations

- Frequentist formulation
  - $R(a_1), \ldots R(a_N)$ are **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 **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 approach:

Optimism in face of Uncertainty



- The more we are uncertain on a specific choice
- The more we want the algorithm to explore that option
- We might lose some value in the current round, but it might turn out that the explored action is the best one

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** (e.g., more than $1-\delta$, $\delta \in (0,1)$)

- The bound length $B_t(a_i)$ depends on how much information we have on an arm, i.e., 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**)

If we choose an arm many many times we want almost to rely on the empirical mean. If we choose an arm few times we want the bound to be big (to underly the uncertainty)

Hoeffding Inequality Bound

In order to set the upper bound we resort to a classical concentration inequality:

Hoeffding Bound

Let $X_1, \ldots, X_t$ be i.i.d. random variables with support in $[0,1]$ and identical mean $\mathbb{E}[X_i] =: X$ and let $\bar{X}_t = \frac{\sum_{i=1}^{t} X_i}{t}$ be the sample mean. Then:

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

probability that the true mean exceeds the empirical one of a certain value u

obviously the probability gets smaller and smaller as we have $t \rightarrow \infty$ (smaller probability if we have (collect) more samples)

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

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

However, how do we select $B_t(a_i)$? We start from fixing:

Computing the Upper Bound

- Pick a probability p that the real value exceeds the bound:

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

- Solve to find $B_t(a_i)$:

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

- Reduce the value of p over time, e.g., $p = t^{-4}$

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

with this we can guarantee: $\mathbb{P}(R(a_i) > \hat{R}_t(a_i) + B_t(a_i)) \leq t^{-4}$

- Ensure to select the optimal action as the number of samples increases:

$$\lim_{t\to\infty} B_t(a_i) = 0 \Rightarrow \lim_{t\to\infty} U_t(a_i) = R(a_i)$$

UCB1 (Upper Confidence Bound 1)

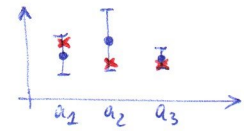
- For each time step t
- Compute $\hat{R}_t(a_i) = \frac{\sum_{i=1}^{t} r_{i,t}\, \mathbb{1}\{a_i = a_{i_t}\}}{N_t(a_i)} \quad \forall a_i$
- Compute $B_t(a_i) = \sqrt{\frac{2 \log t}{N_t(a_i)}} \quad \forall a_i$
- Play arm $a_{i_t} = \arg\max_{a_i \in A} \left(\hat{R}_t(a_i) + B_t(a_i) \right)$

Theorem (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:

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

EXPLORATION / EXPLOITATION: we're **not** choosing the best arm so far, but the arm with the best upper confidence bound:



- empirical mean
- true mean

In this case we're going to choose a_2 because it has the higher upper bound. Since we're probably going to choose it many time, the upper bound will eventually become closer to the true mean, and then it'll be convenient to choose a_1.

Other UCBs

- UCBV → Bernstein inequality
- KLUCB → Chebichev upper bound
- BayesUCB → Upper bounds based on beta distribution
- ...

Bayesian approach:

Thompson Sampling

Designed by William R. Thompson in 1933
General Bayesian methodology for online learning

- 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 $\hat{r}_1, \ldots, \hat{r}_N$
- **Pull** the arm a_{i_t} with the highest sampled value $i_t = \arg\max_i \hat{r}_i$
- **Update** the prior incorporating the new information

Thompson Sampling for Bernoulli Rewards

the possible rewards are 0/1

- The prior conjugate distributions are $Beta(\alpha, \beta)$ and the Bernoulli
- Start from a prior $f_i(0) = Beta(1, 1)$ (uniform prior) for each arm a_i
- Keep a distribution $f_i(t) = Beta(\alpha_t, \beta_t)$ incorporating information gathered from each arm a_i
- Incremental update formula for the pulled arm a_i:
  - 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 two parameters of a Beta distribution **Beta(α, β)** represents:
α = # successes (# 1's)
β = # failures (# 0's)

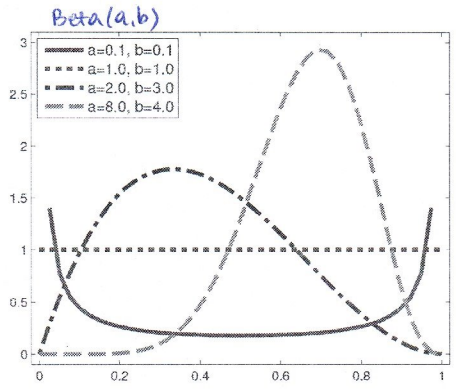
Theorem (Thompson Sampling Upper Bound, Kaufmann & Munos 2012)

At time T, the expected total regret of Thompson Sampling algorithm applied to a stochastic MAB problem is:

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

The exploration/exploitation trade off is represented into the updating + sampling from the priors/posteriors.

Examples of Beta Distributions



Outline

Adversarial MAB Setting

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 $r_{i,t}$, $\forall i$
- The agent gets reward $r_{i_t,t}$
- The final objective of the agent is to maximize the cumulative reward over a time horizon T:

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

Adversarial Regret Definition

- We cannot compare the cumulated 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)

Weak Regret

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

rewards choosing a constant action

rewards that we collected

We are comparing the policy with the best constant action

Lower Bound

Theorem (Minimax Lower Bound)

Let sup be the supremum over all distribution of rewards such that, for $i \in \{1, \dots, N\}$ the rewards $r_{i,1}, \dots, r_{i,N}$ and $r_{i,j} \in \{0, 1\}$ are i.i.d., and let inf be the infimum over all forecasters. Then:

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

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

Why is this a inf sup $\mathbb{E}[L_T]$? We're trying to minimize (inf) the regret, while the adversary is trying to maximize (sup) it. That's why this is presented as min-max lower bound

it's not $\log(T)$ anymore → the other performed better in terms of lower bound (we want it as low as possible)

EXP3

- Variation of the Softmax algorithm
- Probability of choosing an arm:

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

where:

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

η tells us at which speed we should update, the overall value is > 1 if the reward is positive (and so we increase the weight of the action) and < 1 if it's negative

EXP3 Upper Bound

Theorem (EXP3 Upper Bound)

At time T, the expected total regret of EXP3 algorithm applied to an adversarial MAB problem with $\beta = \eta = \sqrt{\frac{N \log N}{(e-1)T}}$ is:

$$\mathbb{E}[L_T] \leq O(\sqrt{TN \log N}),$$

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

Outline

An Example

- 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 MAB techniques over each one of the breweries
- Called **Contextual MAB**

we don't have a single state, we have multiple states for which we cannot control the transition

however once we are in a state (in a context) then we can use the stochastic MAB approach

Other Type of Bandit Problems

- Arm identification problem: we just want to identify the optimal arm with a given confidence, without caring about regret
- 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 $\mathcal{A}$ which is not finite (discrete)
- Unimodal bandits: the arms are ordered in such a way that their expected value increases monotonically until some unknown point and than decease monotonically
- Expert setting: we are allowed also to see the reward which would have given the not pulled arm each turn (online learning problem)

References

An exhaustive review of most of the existing bandit techniques and results is:

Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems

By Sebastien Bubeck and Nicolò Cesa-Bianchi

References - Regret Bounds for RL

Do we have **regret** bounds for RL algorithms?

- Regret should be defined in another way (using **value function**)
- Most of the results are on **tabular** MDPs[1] (MDP with a finite number of states and actions)
- Recent developments in continuous actions/states setting[2]
- Most approaches based on the **OFU principle** (as UCB)

[1] For an overview, see: Zhang, Zihan, Xiangyang Ji, and Simon S. Du. "Is reinforcement learning more difficult than bandits? a near-optimal algorithm escaping the curse of horizon." arXiv preprint arXiv:2009.13503 (2020).

[2] Papini, Matteo, et al. "Optimistic policy optimization via multiple importance sampling." International Conference on Machine Learning. PMLR, 2019.