

Multi-armed bandits

Or an MDP with one state

General information

About this lecture

Multi-armed bandits

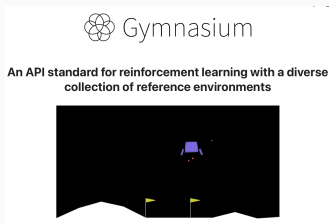
A Bayesian Approach: Thompson Sampling

General information

Contact: guillaume.lagarde@labri.fr

Some references:

- Reinforcement Learning: An Introduction, Sutton and Barto, 2018
- Deepmind Course on RL, D. Silver
- MVA Course on Reinforcement Learning, A. Lazaric



Gymnasium:

<https://gymnasium.farama.org>



AlphaGo the movie:

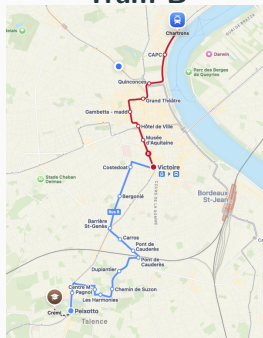
About this lecture

Which route to take?

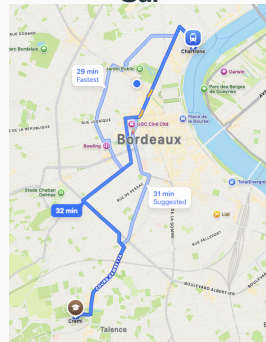
Bike



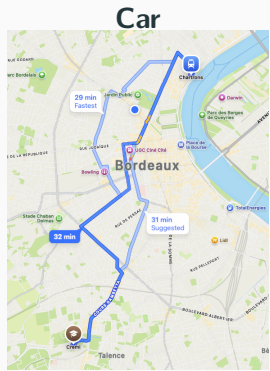
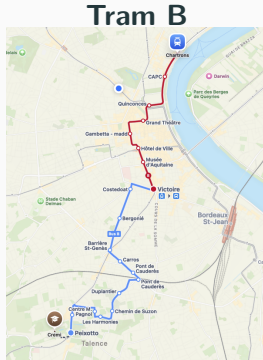
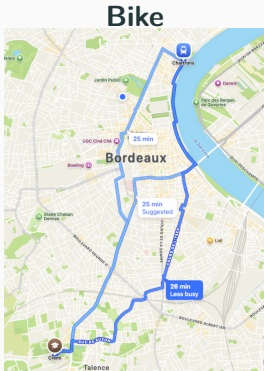
Tram B



Car

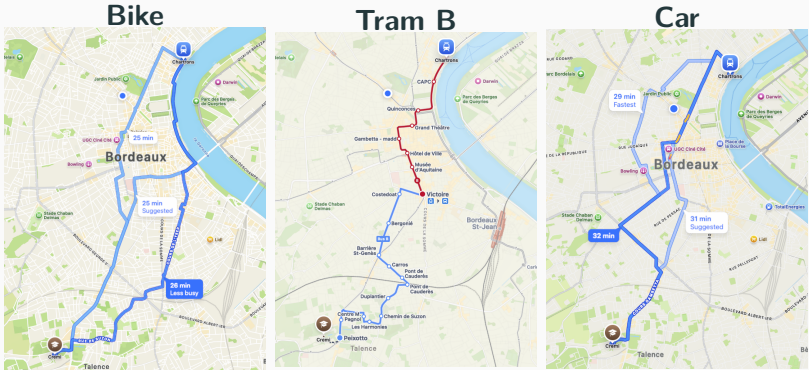


Which route to take?



- 7 possible routes to go from Chartrons to the University
- Suppose no app to tell you which route is the fastest
- You can **try a route** each day and see how long it takes
- You want to **converge** to the best route as fast as possible

Which route to take?



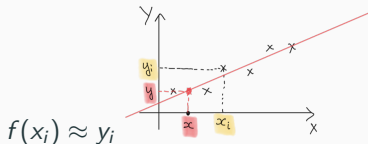
- 7 possible routes to go from Chartrons to the University
- Suppose no app to tell you which route is the fastest
- You can **try a route** each day and see how long it takes
- You want to **converge** to the best route as fast as possible

Tension between **Exploration** and **Exploiting**.

What RL is not

- **Supervised learning**

- Training data: input-output pairs (x_i, y_i) (labeled data)
- Goal: learn a function f that maps inputs to outputs.



- **Unsupervised learning**

- Training data: input data x_i (no labels)

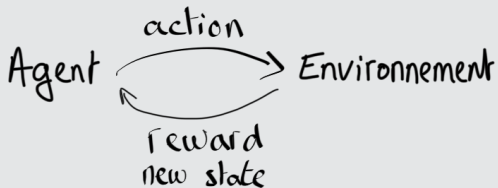
- Goal: find patterns in the data



- active learning
- learn without examples of the optimal behavior
- reward-based learning

Goal

Optimize rewards through repeated actions



Applications

- Game playing, video or board games (AlphaGo, AlphaStar, ...)
- Autonomous driving (robotics in general)
- Program synthesis (code generation)
- Recommendation systems (Netflix, ...)
- Clinical trials (optimize treatments)

Find solutions

- a program that plays go well
- a program that drives a car

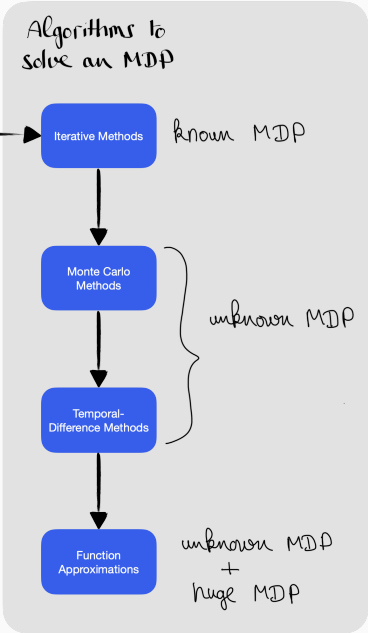
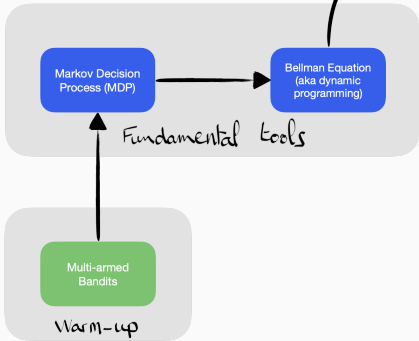
Adapt online

- to the opponent
- to unforeseen events

RL provides algorithms to solve these problems

Multi-armed bandits

■ you're here





- arm = machine = action
- K arms: $1, 2, \dots, K$ Each machine has a **unknown reward distribution**: \mathcal{D}_k , with mean $\mu_k = \mathbb{E}[r_k]$. Rewards are bounded in $[0, 1]$.

Scenario

- for $t = 1, 2, \dots$
 - choose an arm $a_t \in \{1, 2, \dots, K\}$
 - receive a reward $r_t \sim \mathcal{D}_{a_t}$

We play either for a fixed number of steps T or indefinitely (finite or infinite horizon).

Why bandits?

- Simplest form of RL, yet interesting!
- Good warm-up before tackling more complex problems
- Basic building block for more complex RL problems
- Actions don't change the state of the environment
- Actions impact only **immediate reward**



- arm = machine = action
- K arms: $1, 2, \dots, K$
- Each machine has a **unknown reward distribution**: \mathcal{D}_k , with mean μ_k

Goal

- find the best arm: $a^* = \arg \max_{k=1}^K \mu_k$
- maximize the **cumulative reward** over T time steps

$$R(T) = \sum_{t=1}^T r_t$$

- arm = machine = action
- K arms: $1, 2, \dots, K$
- **Unknown reward distribution:** \mathcal{D}_k , with mean μ_k

Goal

- find the best arm: $a^* = \arg \max_{k=1}^K \mu_k$
- we denote $\mu^* = \max_{k=1}^K \mu_k = \mu_{a^*}$
- minimize the **regret** over T time steps

Regret

The **regret** over T rounds is the difference between the best expected total reward and the expected total reward of the realized actions

$$L(T) = T \cdot \mu^* - \mathbb{E}\left[\sum_{t=1}^T r_t\right]$$

Random Bandit

Showtime!

Find a strategy σ that minimizes the regret

$$\min_{\sigma} L(T) = \max_{\sigma} \mathbb{E}[R(T)]$$

Find a strategy σ that minimizes the regret

$$\min_{\sigma} L(T) = \max_{\sigma} \mathbb{E}[R(T)]$$

Algorithms

- Explore-then-Exploit
- ϵ -greedy
- UCB
- Thompson Sampling

Difficulties

Make choices based on incomplete statistics

Trade-off between

- **exploitation**: maximize performance based on current knowledge
- **exploration**: increase knowledge to improve future decisions

Best long-term strategy may involve short-term sacrifices.

Some notation

- r_t : reward at time t

-

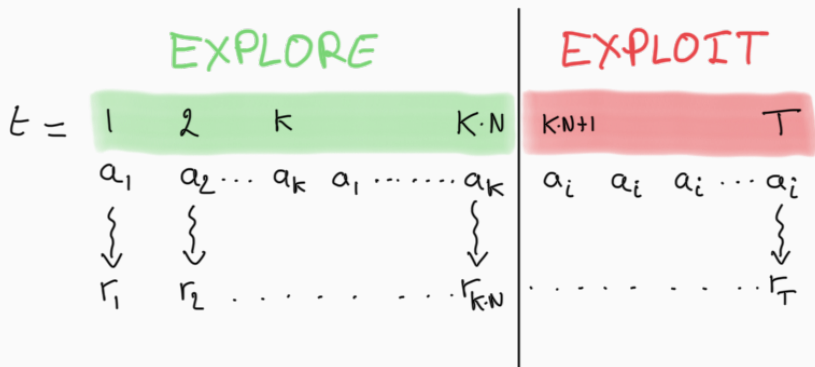
$$r_{i,t} = \begin{cases} r_t & \text{if action } i \text{ was played at time } t \\ 0 & \text{otherwise} \end{cases}$$

- $n_i(T)$: number of times action i was played up to time T
- $\hat{\mu}_i(T)$: empirical mean of rewards for action i up to time T

$$\hat{\mu}_i(T) = \frac{1}{n_i(T)} \sum_{t=1}^T r_{i,t}$$

Explore-then-Exploit

- **Explore:** Try each arm a fixed number of times, N .
- **Compute:** Estimate the mean reward $\hat{\mu}_i$ of each arm based on the collected data.
- **Exploit:** Play the arm with the highest estimated mean reward for the remaining time steps.



What can go wrong?

What can go wrong?

- **Not enough exploration:** We may not have enough data to estimate the mean reward accurately.
- **Too much exploration:** We may waste time playing suboptimal arms.

Problem

We don't know in advance what number of plays are needed so that the estimates are precise enough to separate the best arm from the others.

Never stop exploring!

$\varepsilon \in]0, 1[$.

$$\text{Play} = \begin{cases} \text{Uniformly at random} & \text{with probability } \varepsilon, \\ \arg \max_{i \in [1, K]} \hat{\mu}_i(T) & \text{with probability } 1 - \varepsilon \end{cases}$$

How to update the mean estimates?

We need to maintain the estimates $\hat{\mu}_i(T) = \frac{\sum_{t=1}^T r_{i,t}}{n_i(T)}$.

Compute the sum at each time step \rightarrow update in $O(T)$, this is not efficient

How to update the mean estimates?

We need to maintain the estimates $\hat{\mu}_i(T) = \frac{\sum_{t=1}^T r_{i,t}}{n_i(T)}$.

Compute the sum at each time step \rightarrow update in $O(T)$, this is not efficient

Trick. Express $\hat{\mu}_i(T+1)$ in terms of $\hat{\mu}_i(T)$ and $r_{i,T+1}$.

$$\hat{\mu}_i(T+1) = \hat{\mu}_i(T) + \frac{1}{n_i(T+1)}(r_{i,T+1} - \hat{\mu}_i(T))$$

NEW = OLD + α · [TARGET - OLD]

STEP SIZE ERROR ESTIMATE

Now we can update in $O(1)$.

At the blackboard

Issues with ε -Greedy:

- Exploration never stops
- Exploration does not take into account existing knowledge
- May take a long time to converge (i.e., so that $\arg \max \hat{\mu}_i(T) = a^*$)

Question:

Can we beat a linear regret?

What is the best we can hope for?

Lower bound on the regret

Theorem (Lai and Robbins, 1985)

For any bandit algorithm, the regret satisfies

$$\liminf_{T \rightarrow \infty} L(T) \geq \log(T) \cdot \sum_{i: \mu_i < \mu^*} \frac{\mu^* - \mu_i}{D_{KL}(\mathcal{D}_i \parallel \mathcal{D}_*)} = \Omega(\log(T))$$

where $D_{KL}(p, q)$ is the Kullback-Leibler divergence between the two distributions p and q .

This is **way better** than a linear regret!...

... But can we achieve it in practice?

Upper Confidence Bound (UCB)

Also known as: *Optimism in the Face of Uncertainty*

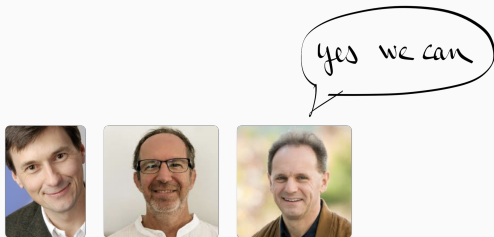


Figure 1: Auer, P., Cesa-Bianchi, N. & Fischer, P

The UCB algorithm balances **optimally** exploration and exploitation by considering the uncertainty in reward estimates.

UCB Algorithm

- At time T :

$$\text{Play} = \arg \max_{i \in [1, K]} \hat{\mu}_i(T) + c(i, T)$$

where:

- $c(i, T) = \sqrt{\frac{2 \log(T)}{n_i(T)}}$.

UCB Algorithm

- **At time T :**

$$\text{Play} = \arg \max_{i \in [1, K]} \hat{\mu}_i(T) + c(i, T)$$

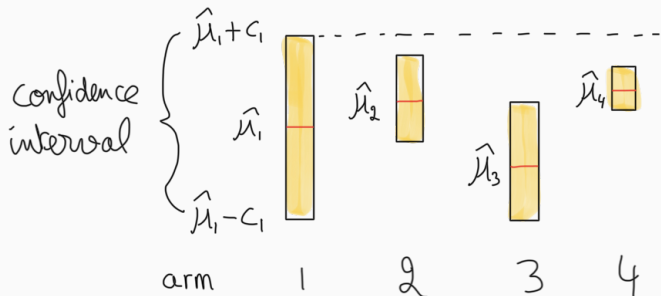
where:

- $c(i, T) = \sqrt{\frac{2 \log(T)}{n_i(T)}}$.

Intuition Behind UCB

- **When $n_i(T)$ is small (little information):**
 - $c(i, T)$ is large \rightarrow we need to **explore**.
- **When $n_i(T)$ is large (much information):**
 - $c(i, T)$ is small \rightarrow play according to $\hat{\mu}_i(T)$. It happens exponentially more often as T grows.

The c_i can be seen as a confidence bounds.



Why $c(i, T) = \sqrt{\frac{2 \log(T)}{n_i(T)}}$?

Hoeffding's inequality

Let X_1, X_2, \dots, X_n be i.i.d. samples of a random variable X such that $X \in [0, 1]$. $\mu = \mathbb{E}[X]$. Then, for any $\varepsilon > 0$,

$$\mathbb{P} \left(\left| \frac{1}{n} \sum_{i=1}^n X_i - \mu \right| \geq \varepsilon \right) \leq 2 \exp(-2n\varepsilon^2)$$

Apply to our case: $\mathbb{P}(|\bar{X} - \mu| \geq c/n) \leq 2 \exp(-2nc^2/n) = 2 \exp(-2c^2)$
 $\leq 2 \exp(-2\varepsilon^2)$
 $\leq \frac{2}{n^2} \exp(0)$
 $\leq \frac{2}{n^2}$
 When n is large the probability to be $\geq \frac{2}{n^2}$ is very small. \Rightarrow high probability.

Intuition on the board.

UCB has a logarithmic regret

Theorem (Auer, Bianchi, and Fischer, 2002)

The UCB algorithm satisfies

$$L(T) \leq 8 \cdot \sum_{i: \mu_i < \mu^*} \frac{\log(T)}{\Delta_i} + O\left(\sum_i \Delta_i\right)$$

where $\Delta_i = \mu^* - \mu_i$.

The regret is logarithmic in T . This is close to the lower bound!

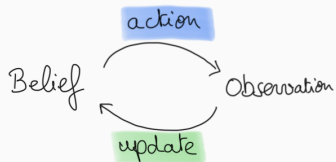
A Bayesian Approach: Thompson Sampling

What is a Bayesian approach?

To simplify, rewards are 0 or 1, given by a Bernoulli distribution.

Key idea.

- Models what you observe as a distribution (**belief**) & update when you observe new observations.
- Take action based on your belief using Bayesian inference.



Bayesian Bandits

We set a belief on the mean reward of each arm, using a beta distribution.

$$P(\mu_i) = \mathcal{B}(1 + \alpha_i, 1 + \beta_i)$$

where

- α_i : number of successes for arm i
- β_i : number of failures for arm i

Beta distribution

- $\mathcal{B}_i(\alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1-x)^{\beta-1} \propto x^{\alpha-1} (1-x)^{\beta-1}$

Thompson Sampling Algorithm

- **Initialize** $\alpha_i = 1$ and $\beta_i = 1$ for all arms. (uniform prior)
- **Repeat.**
 1. **Sample** a mean reward $\hat{\mu}_i$ for each arm i from the beta distribution $P(\mu_i) = \mathcal{B}(1 + \alpha_i, 1 + \beta_i)$.
 2. **Play** the arm with the highest sampled mean reward.
 3. **Update** the parameters α_i and β_i based on the observed reward, by computing the posterior distribution.

Here, the update rule is simple. If we played arm i at time t and received reward $r_{i,t}$, then we update the parameters as follows:

$$\alpha_i \leftarrow \alpha_i + r_{i,t}$$

$$\beta_i \leftarrow \beta_i + 1 - r_{i,t}$$

Proof of the update rule

The posterior distribution is (Baye's rule)

$$P(\mu_i | r_{i,t}) \propto P(r_{i,t} | \mu_i) P(\mu_i).$$

Proof of the update rule

The posterior distribution is (Baye's rule)

$$P(\mu_i | r_{i,t}) \propto P(r_{i,t} | \mu_i) P(\mu_i).$$

- $P(r_{i,t} | \mu_i) = \mu_i^{r_{i,t}} (1 - \mu_i)^{1 - r_{i,t}}$
- $P(\mu_i) = \mathcal{B}(1 + \alpha_i, 1 + \beta_i) \propto \mu_i^{\alpha_i - 1} (1 - \mu_i)^{\beta_i - 1}$

Proof of the update rule

The posterior distribution is (Baye's rule)

$$P(\mu_i | r_{i,t}) \propto P(r_{i,t} | \mu_i) P(\mu_i).$$

- $P(r_{i,t} | \mu_i) = \mu_i^{r_{i,t}} (1 - \mu_i)^{1-r_{i,t}}$
- $P(\mu_i) = \mathcal{B}(1 + \alpha_i, 1 + \beta_i) \propto \mu_i^{\alpha_i-1} (1 - \mu_i)^{\beta_i-1}$

Hence,

$$P(\mu_i | r_{i,t}) \propto \mu_i^{\alpha_i+r_{i,t}-1} (1 - \mu_i)^{\beta_i+1-r_{i,t}-1}$$

or equivalently,

$$P(\mu_i | r_{i,t}) \propto \mathcal{B}(1 + \alpha_i + r_{i,t}, 1 + \beta_i + 1 - r_{i,t})$$

Theorem (Agrawal and Goyal, 2012)

Thompson Sampling achieves a logarithmic regret.