Reinforcement Learning

Multi-armed bandits

Or an MDP with a unique state

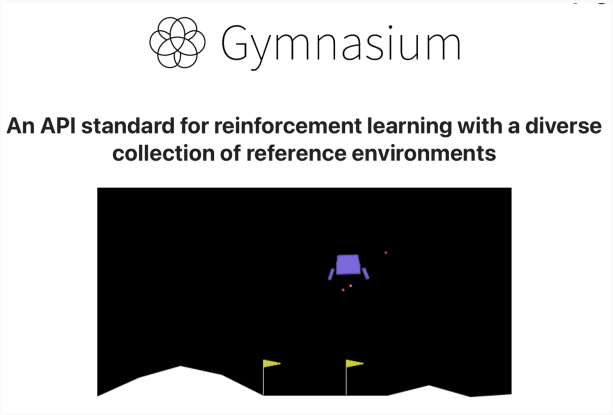
Akka Zemmari

General information

Some references:

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

Gymnasium:



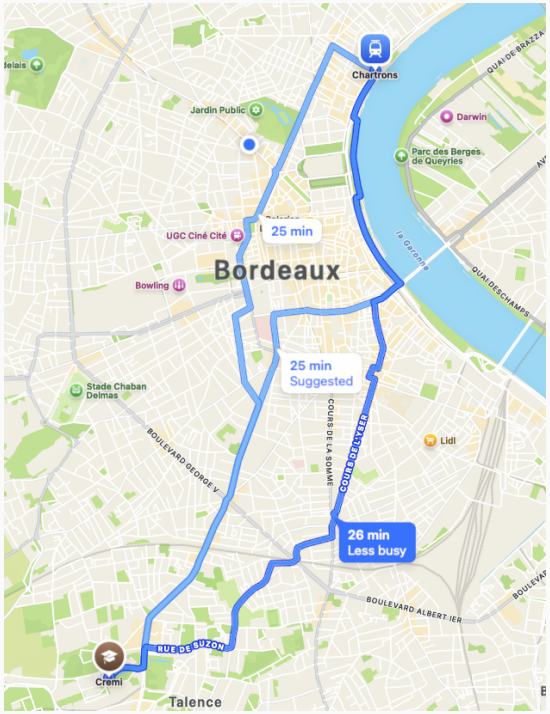
`https://gymnasium.farama.org`

AlphaGo the movie:

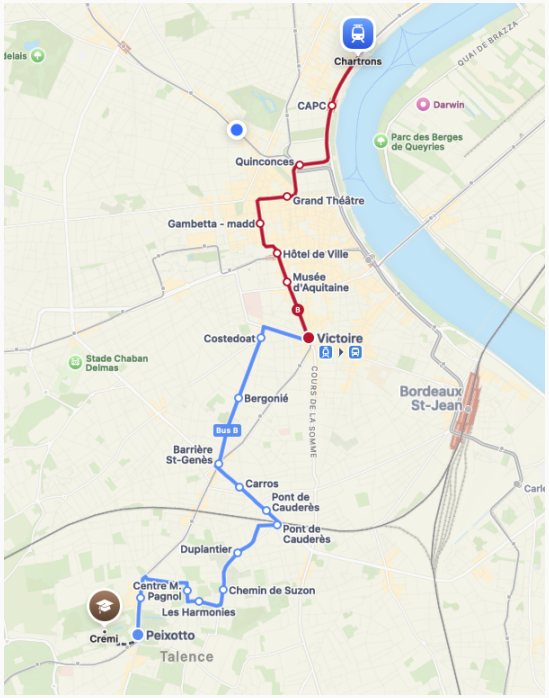
About this lecture

Which route to take?

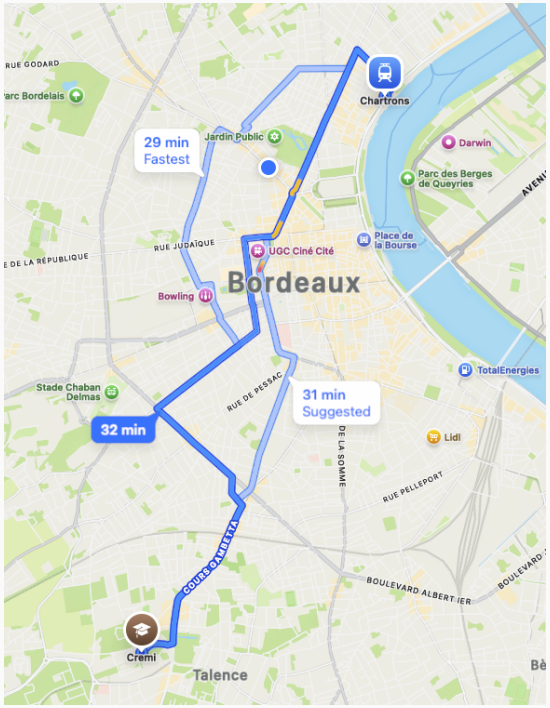
Bike



Tram B



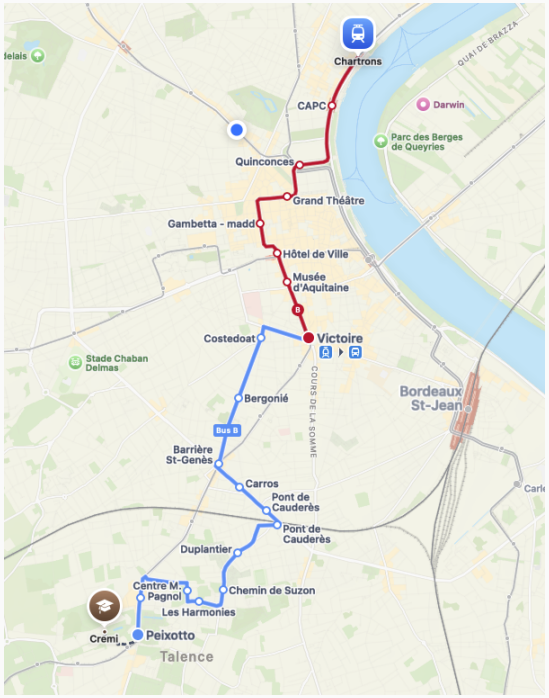
Car



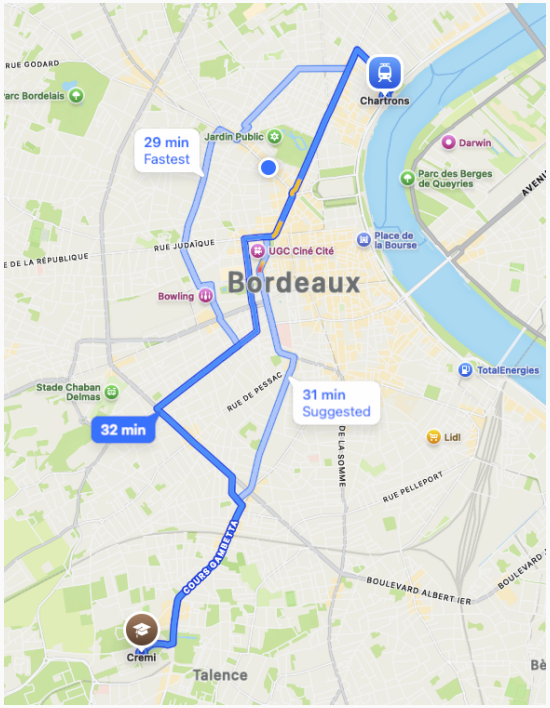
Which route to take?

Bike

Tram B



Car

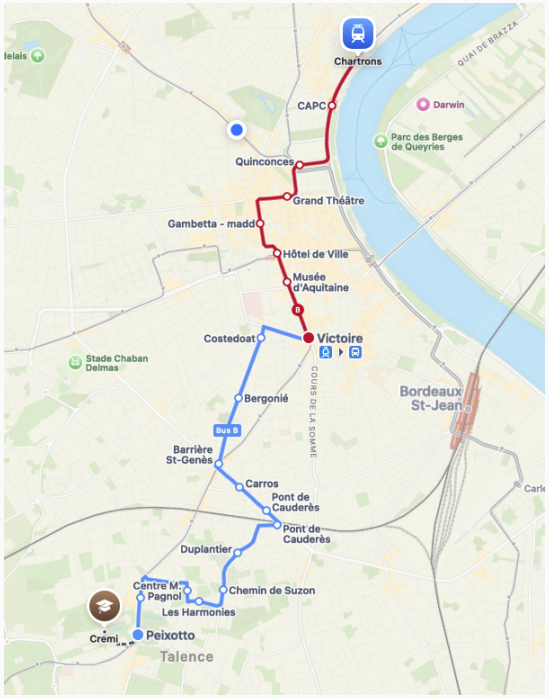


- 7 possible routes to go from Chartrons to the University
- Suppose no app to tell you which route is the fastest
- You can try a route each day and see how long it takes
- You want to converge to the best route as fast as possible

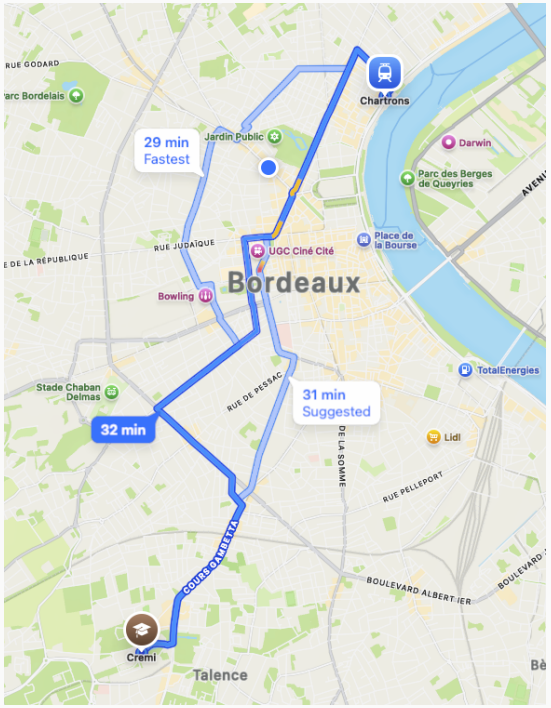
Which route to take?

Bike

Tram B



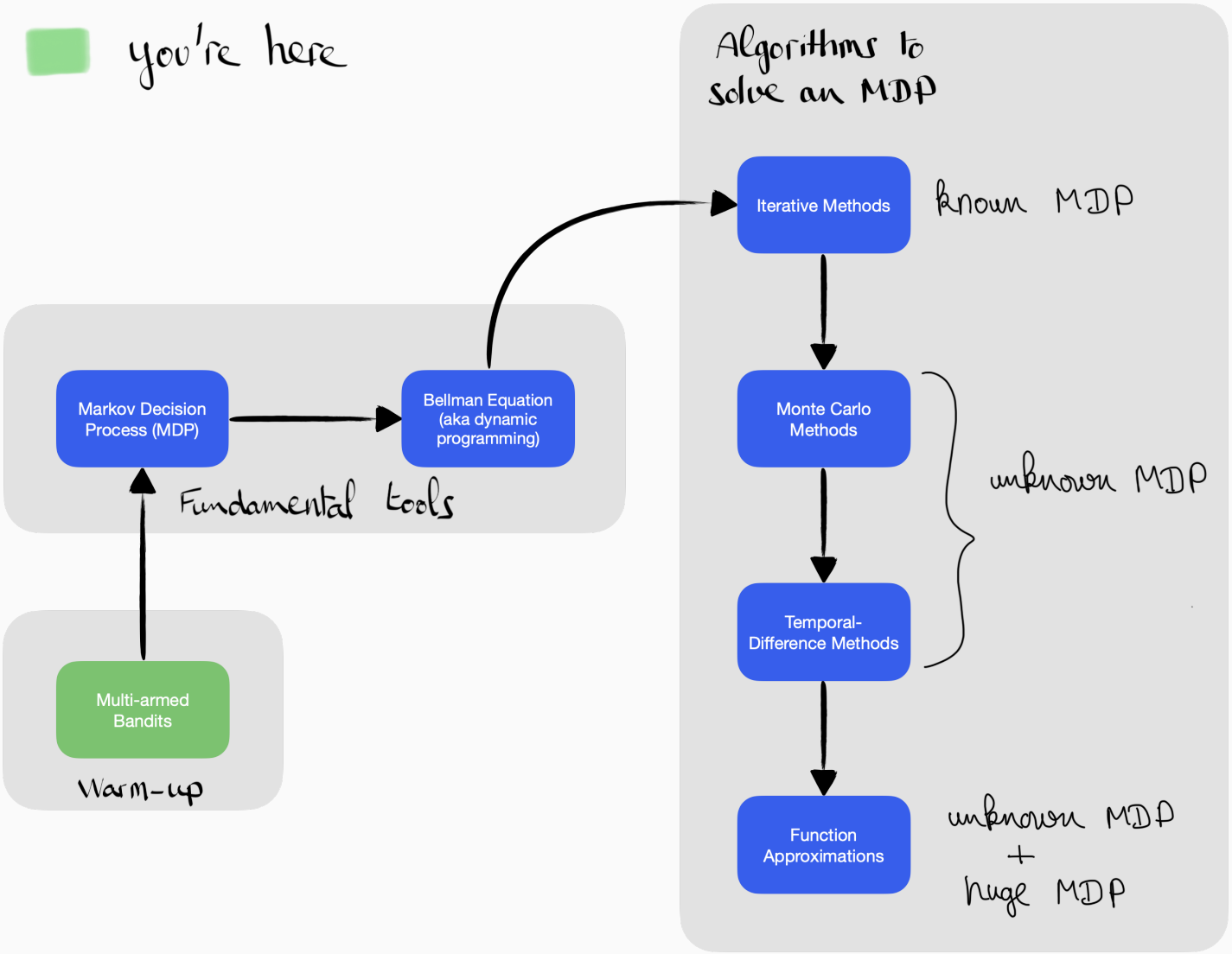
Car



- 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.

Multi-armed bandits


you're here
Algorithms to solve an MDP
Iterative Methods
known MDP
Monte Carlo Methods
Temporal-Difference Methods
unknown MDP
Function Approximations
unknown MDP + huge MDP
Markov Decision Process (MDP)
Bellman Equation (aka dynamic programming)
Fundamental tools
Multi-armed Bandits
Warm-up

- arm = machine = action
- K arms: $1, 2, \ldots, K$ 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 $t = 1, 2, \ldots$
  - choose an arm $a_t \in \{1, 2, \ldots, K\}$
  - 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: $1, 2, \ldots, K$
- 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: $1, 2, \ldots, K$
- 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

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

Random Bandit

Showtime!

Wanted

Find a strategy σ that minimizes the regret

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

Wanted

Find a strategy σ that minimizes the regret

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

Algorithms

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

Challenges

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.

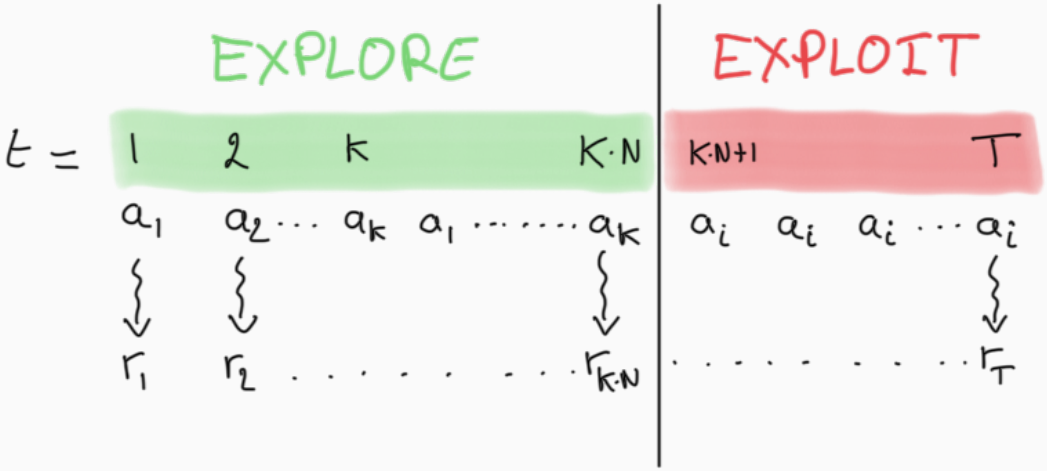
Some notation

- 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.

ε-greedy

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

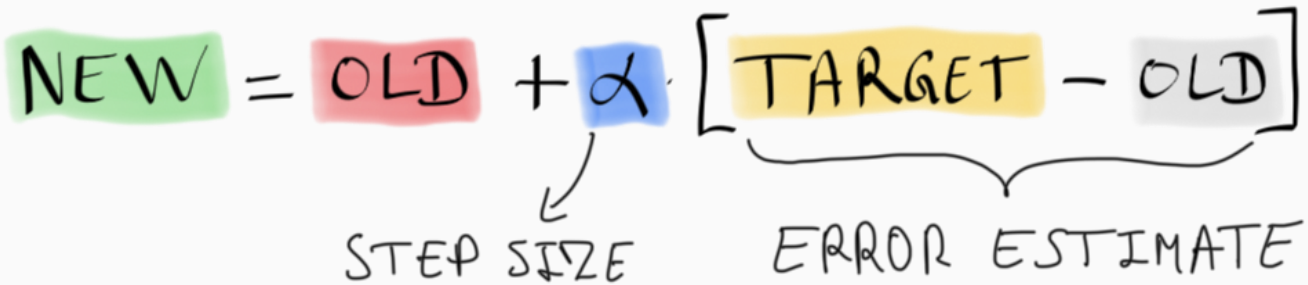
How to update the mean estimates?

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 $r_{i,T+1}$.

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



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

Regret Analysis of ε-Greedy

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

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

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**

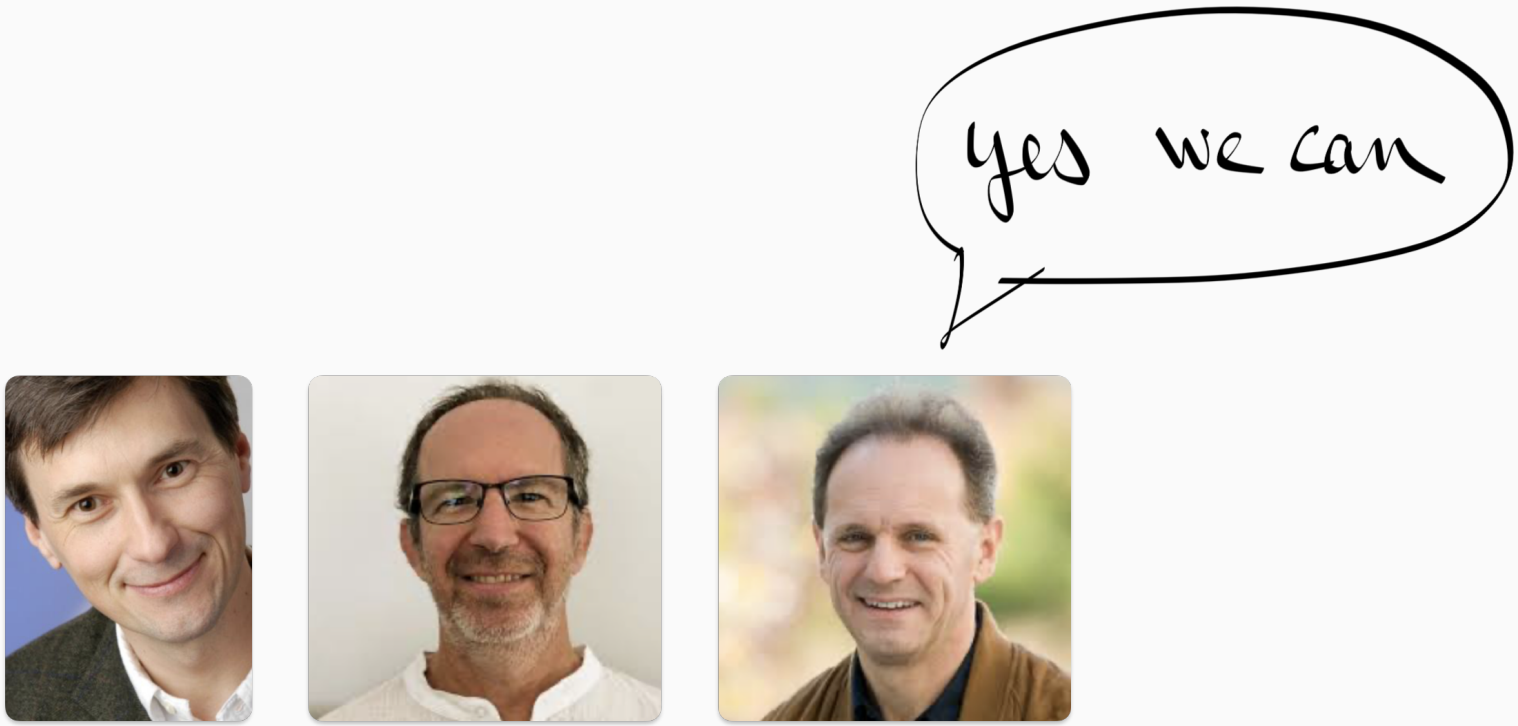


Figure 1: Auer, P., Cesa-Bianchi, N. & Fischer, P

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:

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

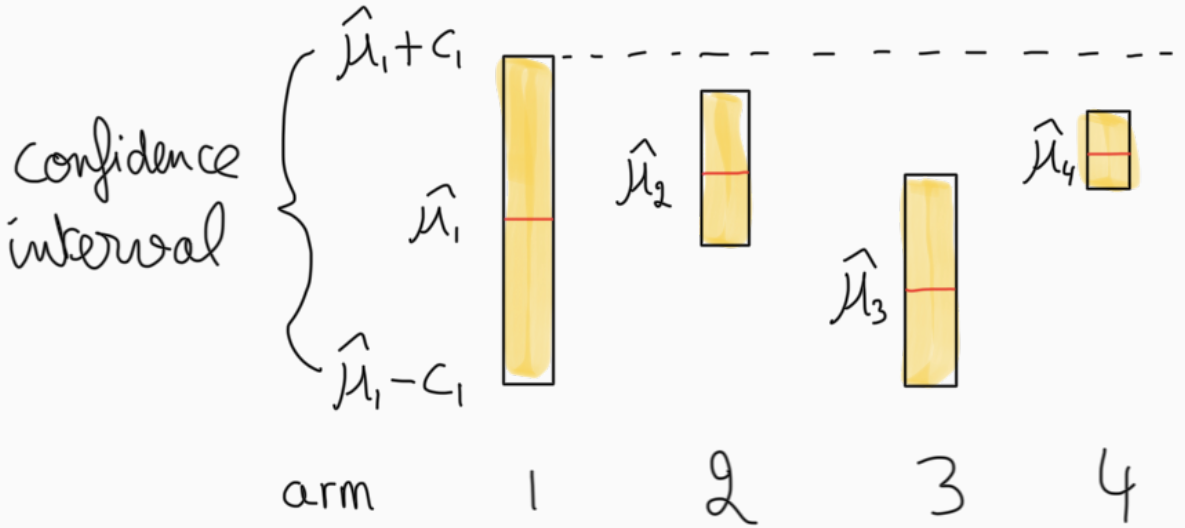
where:

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

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 can be seen as a confidence bounds.

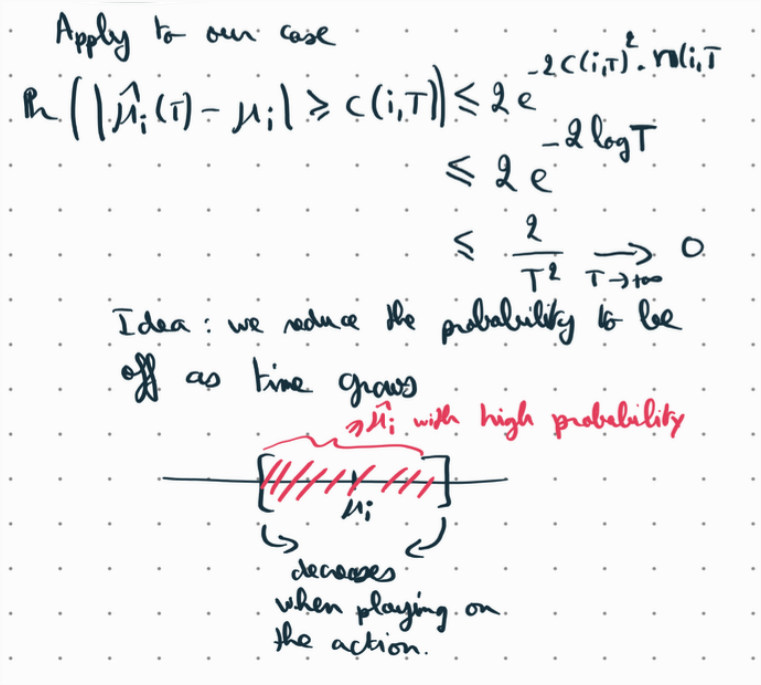


Why $c(i,T) = \sqrt{\frac{2\log(T)}{n_i(T)}}$?

Hoeffding's inequality

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

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



Intuition on the board.

UCB has a logarithmic regret

Theorem (Auer, Bianchi, and Fischer, 2002)

The UCB algorithm satisfies

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

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

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

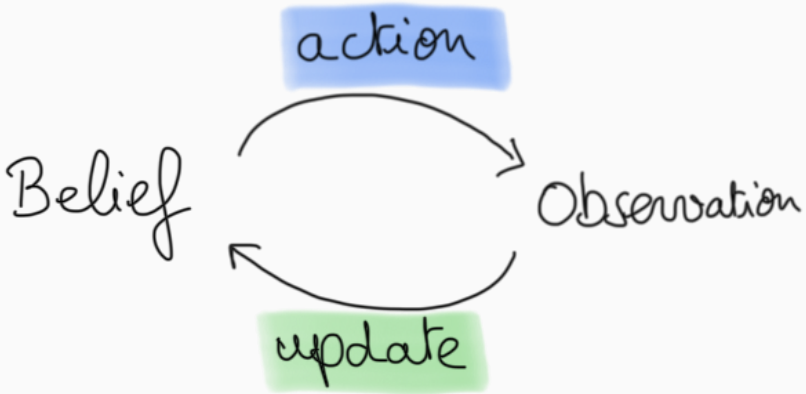
A Bayesian Approach: Thompson Sampling

What is a Bayesian approach?

To simplify, rewards are 0 or 1.

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

Approximate the true reward distribution $\mathcal{D}_i$ of machine i by a Beta distribution $\mathcal{B}_i$ of parameters $\alpha(i,t)$ and $\beta(i,t)$.

$$\mathcal{D}_i \approx \widehat{D}_i(t) = \mathcal{B}_i(\alpha(i,t), \beta(i,t))$$

- $\alpha(i,t)$: number of successes for arm i up to time t.
- $\beta(i,t)$: number of failures for arm i up to time t.
- $\mathcal{B}_i(\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$

Thompson Sampling Algorithm

- **Initialize** $\alpha(i, 0) = 1$ and $\beta(i, 0) = 1$ for all i (uniform prior).
- **Repeat**.
  1. **Sample** a reward $\tilde{r}_i(t)$ from each $\mathcal{B}_i(\alpha(i, t), \beta(i, t))$.
  2. **Play** the arm with the highest sampled reward.
  3. **Update** the parameters $\alpha(i, t)$ and $\beta(i, t)$ based on the observed reward, by maximizing the probability to have seen the observed reward with our model/belief.

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:

$$\alpha(i, t + 1) = \alpha(i, t) + r_{i,t}$$
$$\beta(i, t + 1) = \beta(i, t) + 1 - r_{i,t}$$

Optimality of Thompson Sampling

Theorem (Agrawal and Goyal, 2012)

Thompson Sampling achieves a logarithmic regret.