Reinforcement Learning

robots, trading, symptom checking & co

Rémi Besson, Frédéric Logé

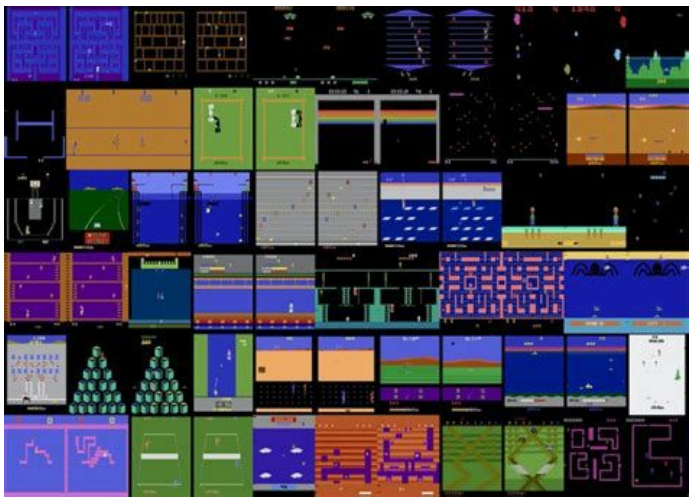
rb.remi.besson@gmail.com, frederic.logemunerel@gmail.com

github.com/FredericLoge/introToRL

ENSAI

January 26^{th}-27^{th}, 2022

News from Reinforcement learning



Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih Koray Kavukcuoglu David Silver Alex Graves Ioannis Antonoglou
Daan Wierstra Martin Riedmiller
DeepMind Technologies
{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com

Abstract

We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. We apply our method to seven Atari 2600 games from the Arcade Learning Environment, with no adjustment of the architecture or learning algorithm. We find that it outperforms all previous approaches on six of the games and surpasses a human expert on three of them.

1 Introduction

Learning to control agents directly from high-dimensional sensory inputs like vision and speech is one of the long-standing challenges of reinforcement learning (RL). Most successful RL applications that operate on these domains have relied on hand-crafted features combined with linear value

What is Machine Learning ?

Definition of ML by Tom Mitchell

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

$\rightarrow$ *Learning with experiences, in opposition to standard algorithms*

Standard categorization of ML

Supervised Learning	**Unsupervised Learning**
With data (X, Y)	With data X
build $\hat{f}$ s.t. $\hat{f}(X) \approx Y$	a. Reduce dimension of X
	b. Find (co)-clusters in X

$\rightarrow$ *What about Reinforcement Learning ?*

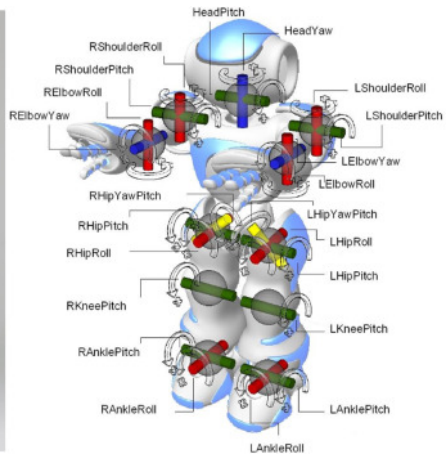
What is Reinforcement Learning ?

Definition

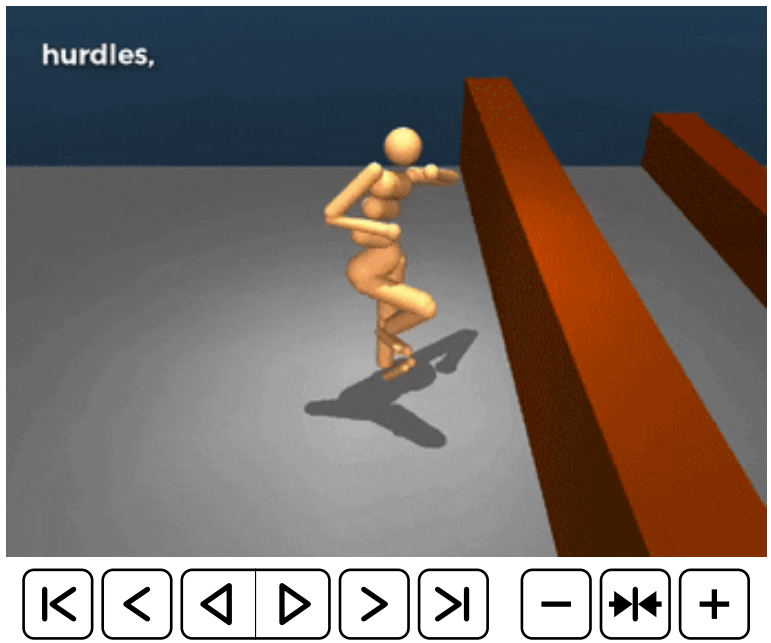
Reinforcement Learning (RL) is solving decision-making problems using **data** and **trial and error** process.

Robot training example

Model



Trial & error



→ ↗

Action recommendation system

↓

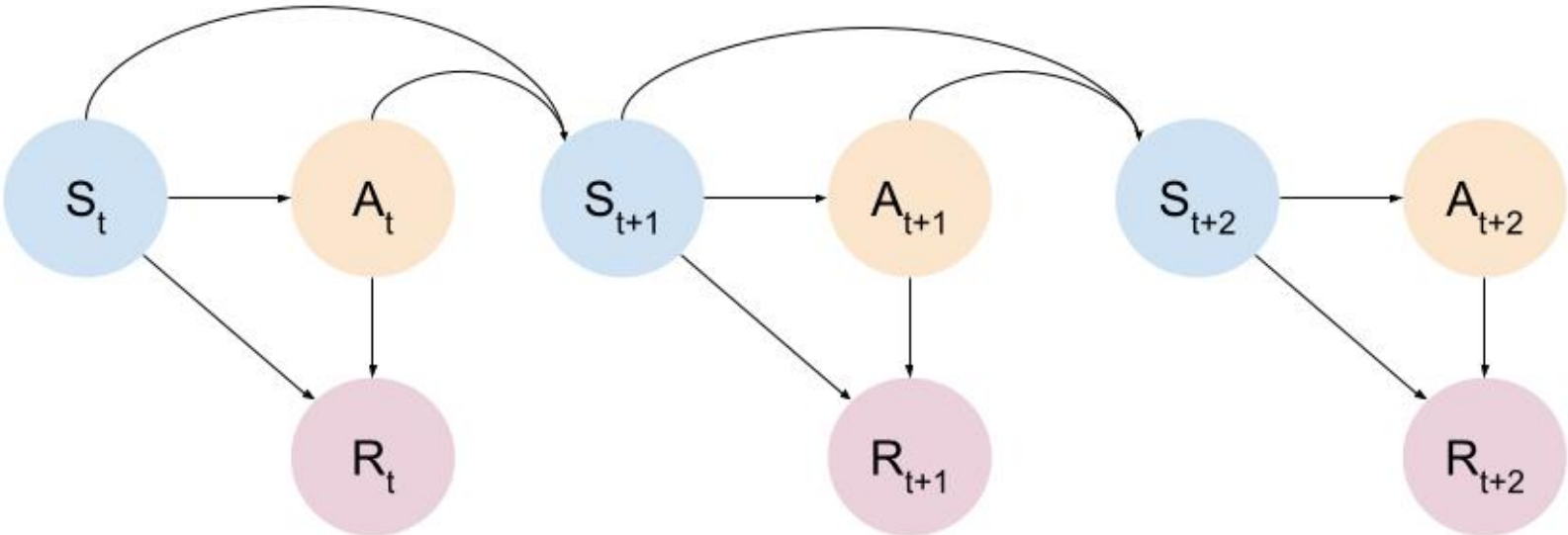
← **Log file** {position, environment, action, progress, is_terminated}

Major notions in this course

Notion	Description
Agent	the element making decisions
	robot
System	(syn. environment) the setup with which the agent interacts
	obstacle course
State	current infos on agent and system (S_t)
	robot parameters & position in course
Action	interaction of the agent with the system (A_t)
	change in model parameters
Reward	quantifies how good a decision was on short-term
	a necessary feedback for the machine (R_t)
	$r_0 \cdot 1\{\texttt{passed obstacle}\} + l_0 \cdot 1\{\texttt{fell}\}$
Policy	action recommendation system
	how to change parameters given course ?
Strategy	reinforcement learning approach to learn best policy
	what part of the system to (not) explore ?

Generic objective

A Markov Decision Process is the go-to model for RL problems



RL aims at finding a good way to take sequential actions.

Mathematically, it tries to solve

$$\arg\max_{(a_{t'})_{t'\geq t}} \mathbb{E}\left[\sum_{t'\geq t} R(S'_t, a'_t)\right]. \tag{1}$$

Note this cannot be solved using standard supervised learning.
RL is a much complex problem than SL.

Major difficulty

Exploration

Dimension Curse

Overview of this 2-day course

<table>
<tr><td>9h45</td><td>Introduction</td><td rowspan="4">MDPs, unknown environment</td></tr>
<tr><td></td><td rowspan="3">Bandits</td></tr>
<tr><td></td></tr>
<tr><td>12h45</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td>14h</td><td rowspan="4">MDPs, known environment</td><td rowspan="4">Projects</td></tr>
<tr><td></td></tr>
<tr><td></td></tr>
<tr><td>17h</td></tr>
</table>

In each section we will see :
a particular problem,
a mathematical framework,
algorithms and principles.

We will also code using `R`, building RL algorithms from scratch.

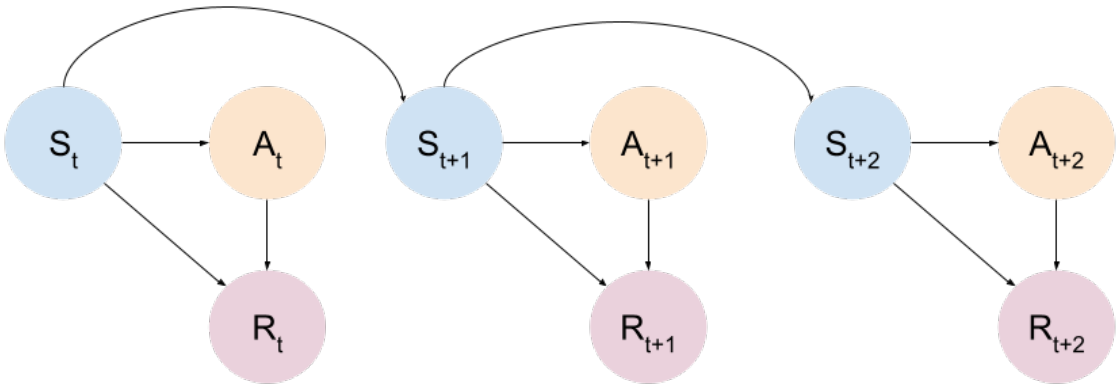
Table of contents

Bandits

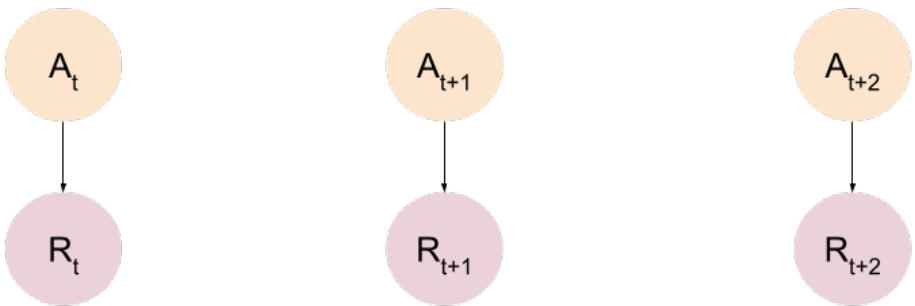
Bandits - Setup

Simpler problems

Problem #1 : actions do not impact system



Problem #2 : no state



$\Rightarrow$ let's start with Problem #2

Bandits - Setup

Definition

Bandit problems are **online decision-making** problems, played over a significant amount of time.

• **Decision-making** :

(a) in an observable context e.g. day of the week, we take an action within an acceptable set e.g. $\{\texttt{write C++}, \texttt{go out with friend}\}$.

(b) following this decision, we get an immediate feedback, grading whether the action was good or not basically (e.g. ★★★☆☆)

• **Online** : with time, you should get better and better at making decisions At first you don't have any data and progressively you collect experience samples depending on the actions you take !

↪ With enough **diverse** data, we should find the best decision. We must play to learn **and** to win : balancing **exploration-exploitation**.

Bandits - Setup

Many (business) applications

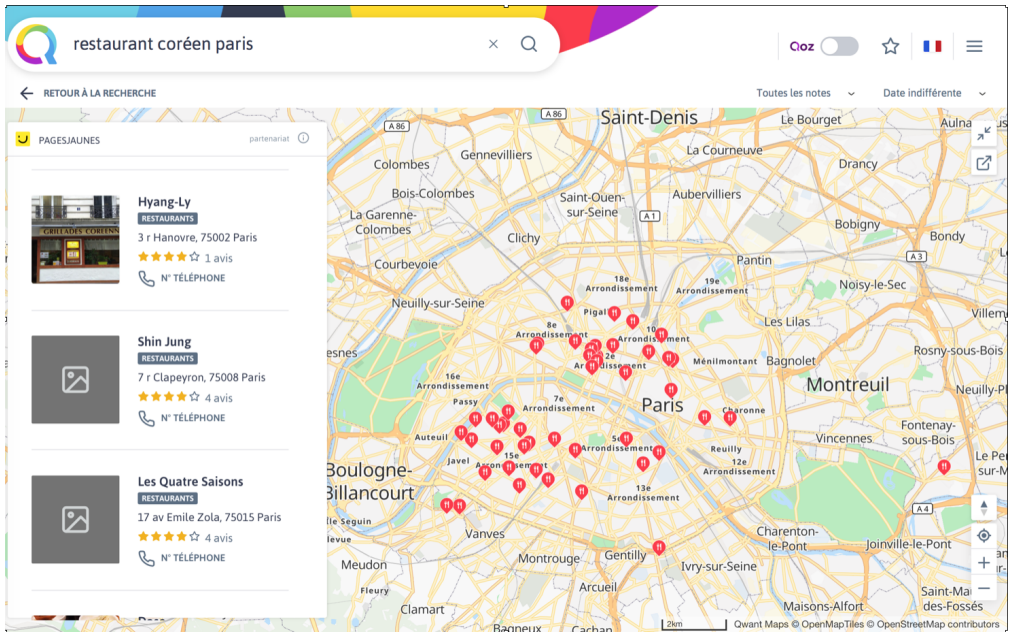
- ad placements
- thumbnail selection for articles/movies
- search engine optimization
- product recommendation
- online educational learning
- smart radio bandwidth
- resource allocation
- expert-based decision-making
- ...
- **choosing your next restaurant**.

Note : A/B testing can be seen as a type of bandit

Bandits - Setup

Choosing a restaurant

You have just arrived in a new neighborhood which has K restaurants.



Of course, you try each of them one time, to be fair, but then, how do you choose where to eat ? And over time ?

$\hookrightarrow$ first off, define the problem mathematically !

Bandits - Setup

Formalization

Let us write $a_t \in \{1, \dots, K\}$ the restaurant chosen at time t and r_t the happiness you got from eating there, called *reward*.

Assumption: $r_t \sim R(a_t)$ and $\mathbb{E}(R(a_t)) = \mu_{a_t}$, average reward.

💬 the best possible action is the one maximizing expected reward

Objective : pick actions which optimize either

$$\text{regret}((a_t)_{t=1}^T) = \sum_{t=1}^{T} \max_a \mu_a - \mu_{a_t} = \sum_{a=1}^{K} \Delta_a n_{a,T} \tag{2}$$

$$\text{pct}((a_t)_{t=1}^T) = \sum_{t=1}^{T} \mathbb{1}\{a_t \neq a^*\} \tag{3}$$

for any large T.

ℹ we choose a_t based on $H_t = (a_{t'}, r_{t'})_{t'<t}$

Code

1. Pull from `github.com/FredericLoge/introToRL/`
2. Open .RProj file in RStudio
3. Run simulations of restaurant.R

Question: is regret/pct deterministic ? Why ?

Note that **pct** should converge towards 1 with time, and **regret** should increase sub-linearly with time.

Bandits - Strategies

Ideas ?

Exploration-exploitation dilemma - How to set a strategy ?

Code, step 1

Implement the following strategies (described in next slides) :

- ε-greedy
- Explore-then-Commit
- EXP3
- Thompson Sampling
- UCB

Code, step 2

Compare the performances of the strategies, play with model parameters

Define the nb of samples and average reward for restaurant a at time t as:

$$n_{a,t} = \sum_{t'<t} \mathbb{1}\{a_{t'} = a\} \qquad \hat{\mu}_{a,t} = \frac{\sum_{t'<t} r_{t'} \mathbb{1}\{a_{t'} = a\}}{\sum_{t'<t} \mathbb{1}\{a_{t'} = a\}} \tag{4}$$

Compute this in R :

```
historic %>%
    group_by(action) %>%
    summarise(
        n_a = n(),
        mu_a = mean(reward)
    )
```

Consider the ε-greedy strategy :

For $t \geq 1$:
  If $t <= m \cdot K$:
    pick $a_t = 1 + t\%\%K$
  Else:
    identify $\hat{a}_t^* = \arg\max_a \hat{\mu}_{a,t}$
    pick $a_t = \hat{a}_t^*$ with probability $1 - \varepsilon$, otherwise random
End For

Consider the Explore-Then-Commit strategy :

For $t \geq 1$:
  If $t <= m \cdot K$:
    pick $a_t = 1 + t\%\%K$
  Else:
    pick $a_t = \arg\max_a \hat{\mu}_{a,m \cdot K}$
End For

EXP3

Consider the EXP3 strategy :

Initialize $w^1 = (0, 0, \ldots, 0)$

For $t \geq 1$:

$p^t = w^t / |w^t|$

$a_t \sim \texttt{Multinomial}(p^t)$

$w^{t+1} = w^t$

$w_{a_t}^{t+1} = w_{a_t}^t \exp\{-\eta r_t / p^t\}$

End For

Consider the UCB strategy :

For $t \geq 1$:
  If $t <= m \cdot K$:
    pick $a_t = 1 + t\%\%K$
  Else:
    pick $a_t = \arg\max_a \hat{\mu}_{a,t} + \text{Regularizer}(n_{a,t})$
End For

❓ how do you build confidence intervals ? why being optimist ?

Consider the TS strategy :

For $t \geq 1$:
  If $t <= m \cdot K$:
    pick $a_t = 1 + t\%\%K$
  Else:
    $\forall a, \tilde{\mu}_{t,a} \sim \texttt{Beta}(1 + \underbrace{n_{a,t}\hat{\mu}_{a,t}}_{\#\text{ successes}}, 1 + \underbrace{n_{a,t}(1 - \hat{\mu}_{a,t})}_{\#\text{ fails}})$
    pick $a_t = \arg\max_a \tilde{\mu}_{t,a}$
End For

❓ why would this work ?

Bandits - Strategies

Key takeaways

- Relatively easy to discard really sub-optimal options
- Aside from Explore-then-Commit, all strategies have good performance without much tuning
- Strong links between this model and hypothesis testing
- You have to learn the reward function whilst playing hence you must learn via playing but you want also to win ! Exploitation-exploration dilemma + regret measure to evaluate performance.

⏩ Check out other bandit models ⏩ Theoretical analysis

Some references

Lattimore and Szepesvári 2020, Bubeck and Cesa-Bianchi 2012 (free)

a lot of great lectures in video.

Csaba Szepesvári, Tor Lattimore, Nicolò Cesa-Bianchi, Sébastien Bubeck, Alessandro Lazaric, Robert Schapire, Aurélien Garivier, Gilles Stoltz, Rémi Munos, Emilie Kaufmann, John Langford, ...

The applications and companies drive the research but other fields of research also drive the research: typically in Reinforcement Learning! see survey Bouneffouf, Rish, and Aggarwal 2020 on applications

Bandits - Strategies

... towards Markov Decision Processes

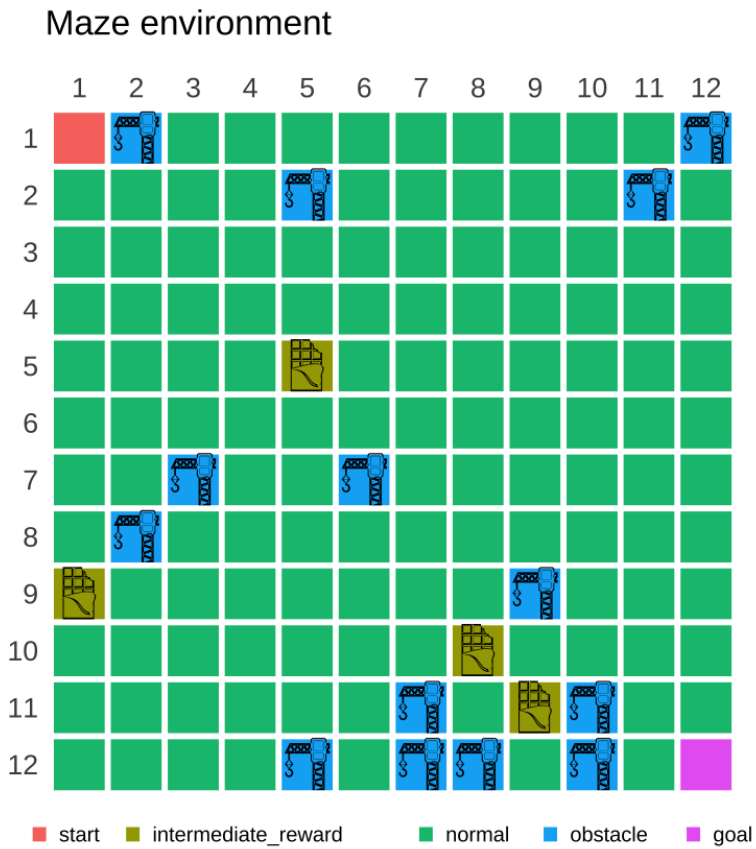
! The action only impacts what you directly observe, it has no impact further in the future. We will see what happens with MDPs

Markov Decision Processes (MDPs), known parameters

MDP-Known - Maze problem

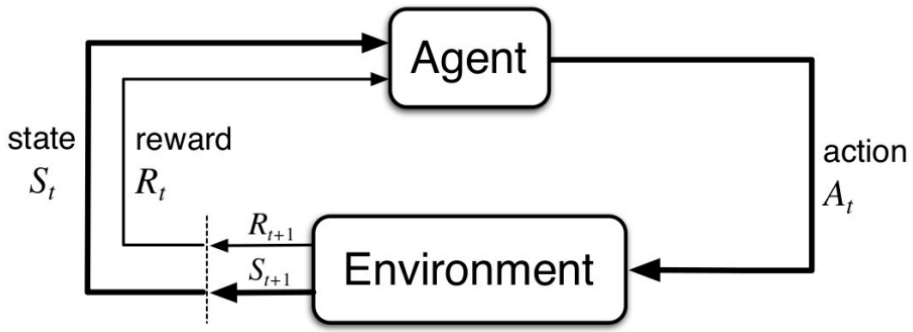
Shortest path problem in a maze

Goal : Control an agent which start in cell S, to find the shortest path toward the cell G in a given maze. At each cell you are allowed to move from one cell toward the left, or the right, up or down.



MDP-Known - Maze problem

Markov Decision Process framework



- At time $t \in \mathbb{N}$
  - Agent is in state $s_t \in \mathcal{S}$
  - Agent takes action $a_t \in \mathcal{A}$
  - Agent receives reward $r_{t+1} \in \mathrm{R}$
  - Agent reaches state $s_{t+1} \in \mathcal{S}$
- Markov property of the environment dynamic (non-dependence on the path) :

$$\mathbb{P}[S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a] = p(s', r \mid s, a)$$

Finite MDP : $\mathcal{S}$, $\mathcal{A}$ and R are finite.

MDP-Known - Maze problem

Exercise

Goal : Define $\mathcal{S}$, $\mathcal{A}$ and R for the maze planning problem. Is the Markov property verified ?

Solution :

$(\mathcal{S}, \mathcal{A}, R)$ a MDP.

Policy

A policy π is a mapping from $\mathcal{S}$ to $\mathcal{A}$

Value function

Let us define V_π the value function of the policy π as :

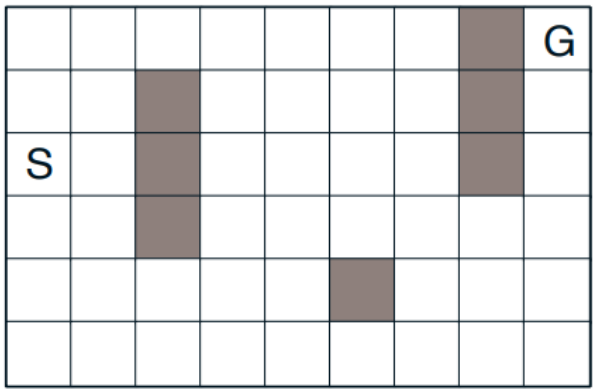
$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid s\right] \tag{5}$$

Amount of reward you can expect when being in state s and following policy π. Two naturals problems :

- Policy evaluation : given a policy π compute $V_\pi(s)$ for each $s \in \mathcal{S}$
- Planning : Our goal is to find the optimal policy $\pi^\star$ such that $\forall s \in \mathcal{S}, V_{\pi^\star}(s) \geq V_\pi(s)$

MDP-Known - Maze problem

Brute force



actions

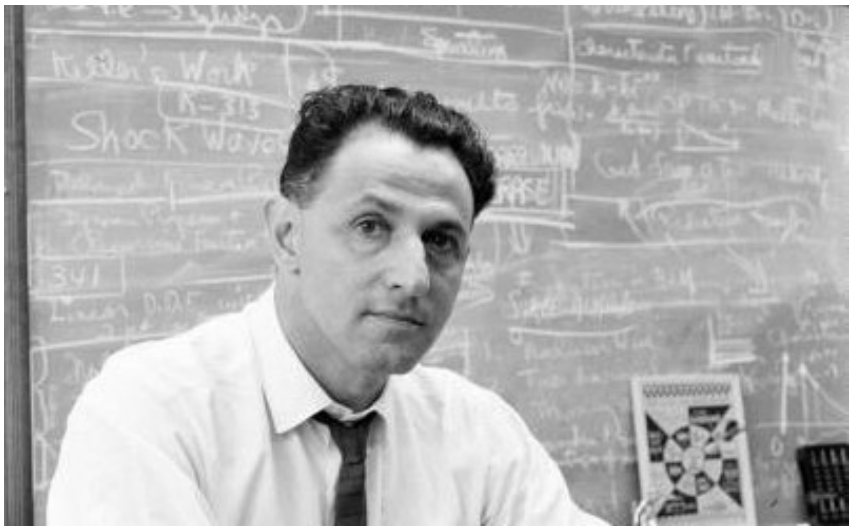
To find the optimal policy let's consider all the possible path and evaluate them ! Curse of the dimension...

A very sub-optimal first brute force proposition : simulate randomly numerous paths, starting from cell S, stopping it when reaching cell G, taking action randomly with uniform probability on each allowed actions, remind the path π and its value $V_\pi(S)$.

Note : you can stop a simulation as soon as the number of step is superior of your current best path.

MDP-Known - Maze problem

Dynamic programming



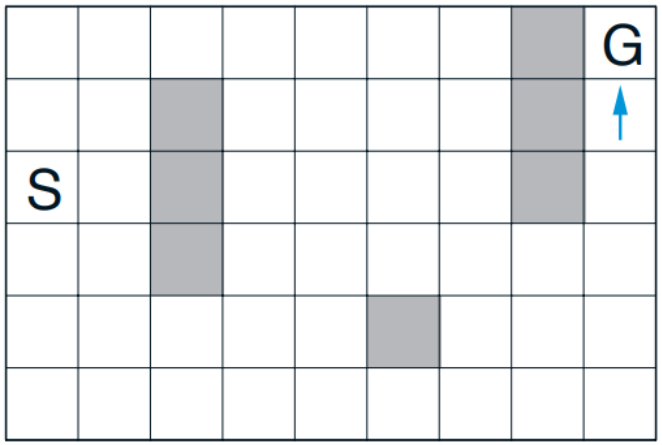
Dynamic programming ($\sim$ 1953) "=" solving an optimisation problem by decomposing it in a recursive manner into interrelated subproblems.

An optimal solution is optimal everywhere. Remind $\pi^\star$ is defined as $\forall s \in \mathcal{S}, V_{\pi^\star}(s) \geq V_\pi(s)$.

By construction $V_{\pi^\star}(s) = \mathbb{E}_\pi[R(s,a)] + \gamma \max_{s' \in \mathcal{S}} V_{\pi^\star}(s')$. To go from A to B is the best option to go from A to C then C to B, hence we can decompose the problem and progressively trim paths.

MDP-Known - Maze problem

Value iteration algorithm



$$V_{\pi^\star}(s) = \max_a \sum_{s'} \sum_r p(s', r \mid s, a)[r + \gamma v_{\pi^\star}(s')] = \mathcal{T}^\star(V_{\pi^\star})(s) \qquad (6)$$

Value iteration algorithm

Iterative algorithm: $V_{k+1}(s) = \mathcal{T}^\star(V_k)(s)$ where $k \geq 0$.

Convergence

$V_k \to V^\star$ if $T < \infty$ or $\gamma < 1$. Banach's fixed-point theorem.

MDP-Known - Maze problem

Value iteration algorithm, pseudocode

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $\quad v \leftarrow V(s)$
| $\quad V(s) \leftarrow \max_a \sum_{s',r} p(s', r \,|\, s, a)\big[r + \gamma V(s')\big]$
| $\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s', r \,|\, s, a)\big[r + \gamma V(s')\big]$

Contraction mapping

Let (X, d) be a complete metric space. A map $\mathcal{T} : X \to X$ is called a contraction mapping on X if there exists $q \in [0, 1[$ such that :

$$d(\mathcal{T}(x), \mathcal{T}(y)) \leq q d(x, y)$$

Theorem

Let (X, d) be a non-empty complete metric space with a contraction mapping $\mathcal{T} : X \to X$. Then $\mathcal{T}$ admits a unique fixed-point $x^\star \in X$ (i.e. $\mathcal{T}(x^\star) = x^\star$).

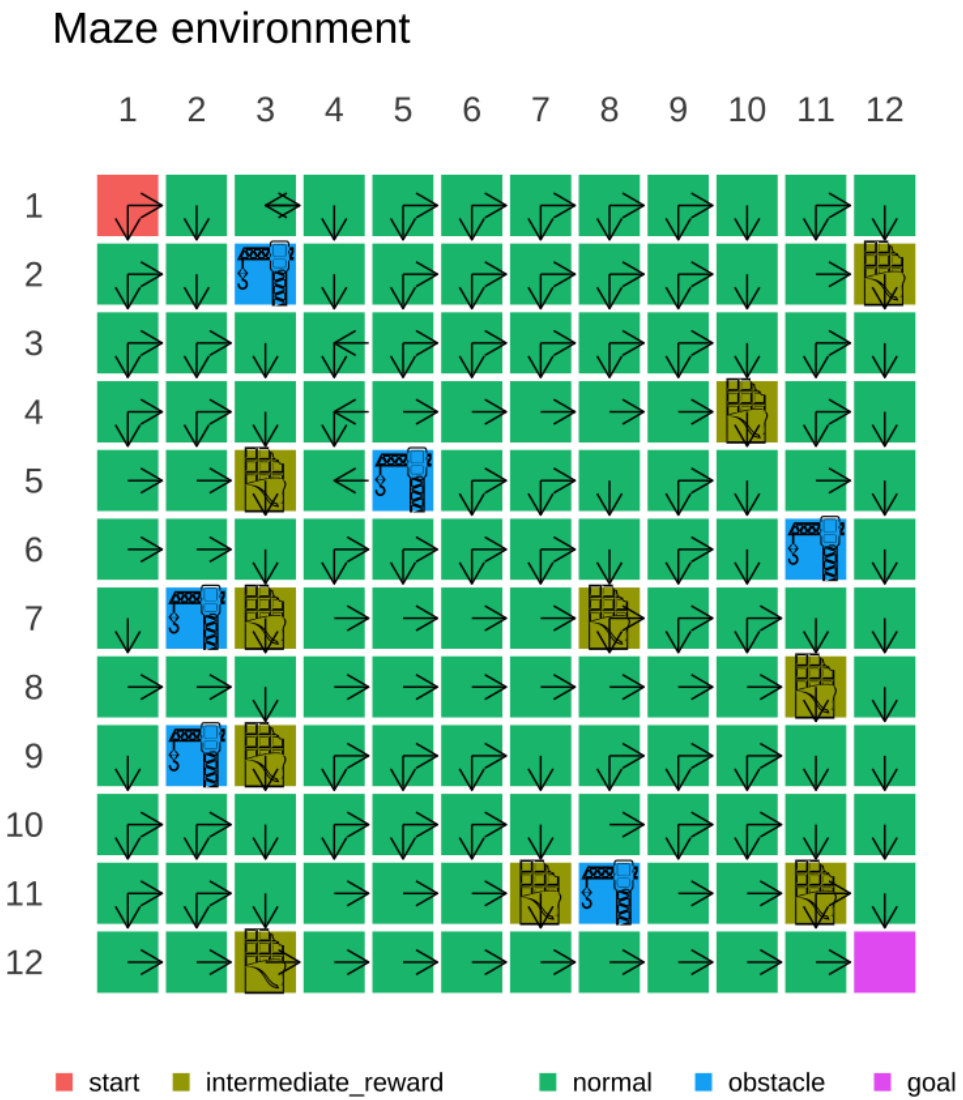
Furthermore, $x^\star$ can be found as follows: start with an arbitrary element $x_0 \in X$ and define a sequence $(x_n)_{n \in \mathbb{N}}$ by $x_n = \mathcal{T}(x_{n-1})$ for $n \geq 1$. Then $\lim_{n \to \infty} x_n = x^\star$.

Exercise

Prove that $\mathcal{T}^\star$ is a contraction mapping. Solution :

MDP-Known - Maze problem

Solving maze problem



Improvement: We may additionally use the information regarding the goal cell and prioritize value iteration by starting from last states and coming back progressively until reaching start state.

MDP-Known - Cliff problem

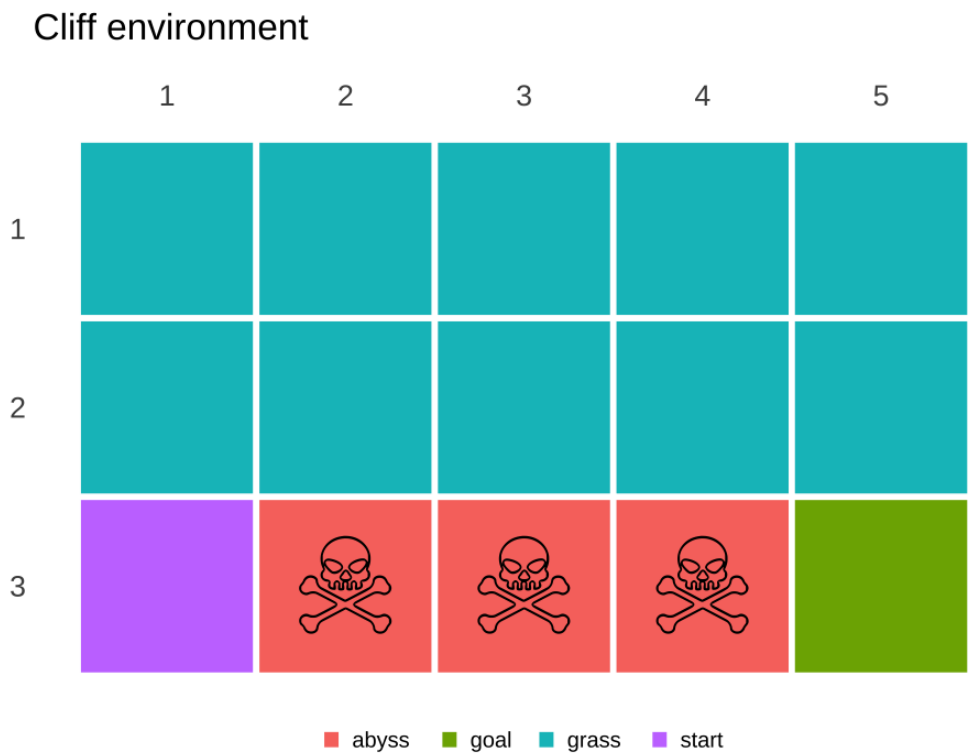
Adding some randomness : cliff problem

Direct extension from Maze : the cliff problem.

Actions : {`north`, `south`, `east`, `west`}

Objective : find path from start to goal

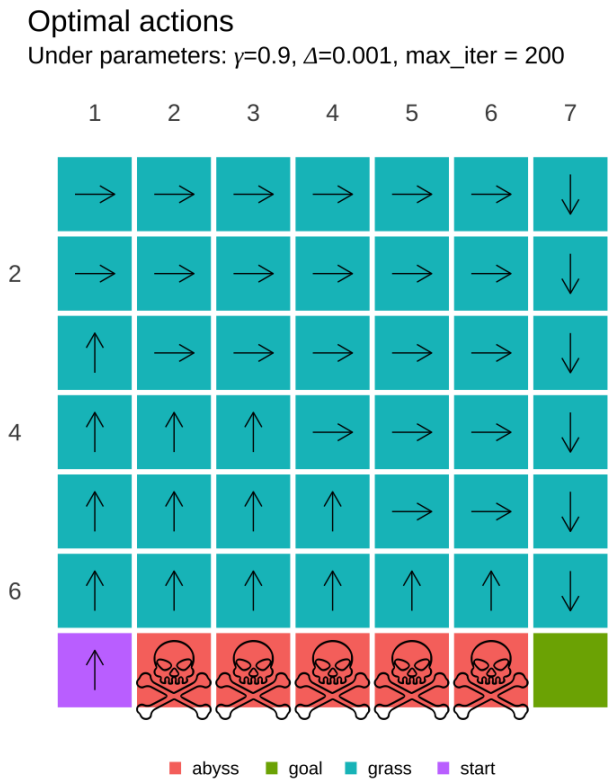
Reward : -1 for each transition, +20 reaching goal, -20 if falling in abyss.

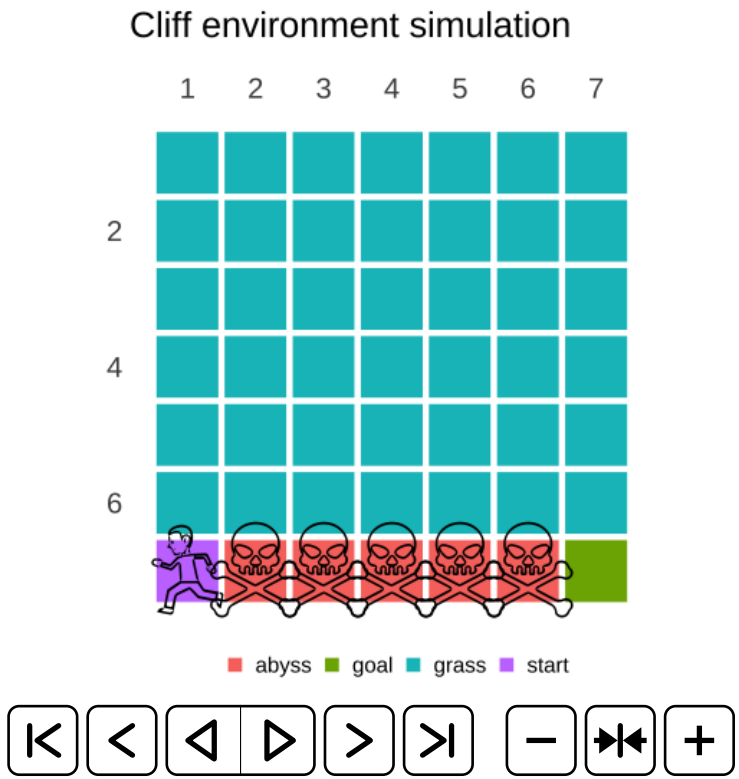


MDP-Known - Cliff problem

Twist

Wind is pushing the person trying to cross the cliff towards the abyss. Use Value Iteration to find optimal trajectory and run simulations. Under strong wind, optimality dictates the following behavior:





Markov Decision Processes (MDPs), unknown parameters

Consider $(S_t, A_t, R_t)_{t\in\{1,\dots,T\}}$ the sequence generated from the interaction of an agent and Markov Decision Process Puterman 2014 $\mathcal{M} \doteq (\mathcal{S}, \mathcal{A}, \mathsf{T}, \mathsf{R})$ where

- $\mathcal{S}$ is the state space
- $\mathcal{A}$ is the action space
- T is the transition function:

$$\forall (s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S} \quad \mathsf{T}(s, a, s') = \mathbb{P}(S_1 = s' | S_0 = s, A_0 = a)$$

- R denotes the reward function:

$$\forall (s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S} \quad \mathsf{R}(s, a, s') = \mathbb{E}(R_0 | S_1 = s', S_0 = s, A_0 = a).$$

Markov property: $(S_{t+1}, R_t) | (S_t, A_t) \perp\!\!\!\perp (S_{t-}, A_{t-})$

Assumption: transition and reward functions are constant throughout the observation period

Objective function

Let Π denote the set of policies assigning to each state $s \in \mathcal{S}$ an action $a \in \mathcal{A}$. Our objective will be to find :

$$\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E}_\pi \left[\sum_{t \geq 0} \gamma^t R_t \right]$$

How to find π^* in an MDP with unknown reward and transition function ?

MDP-Unknown - Setup

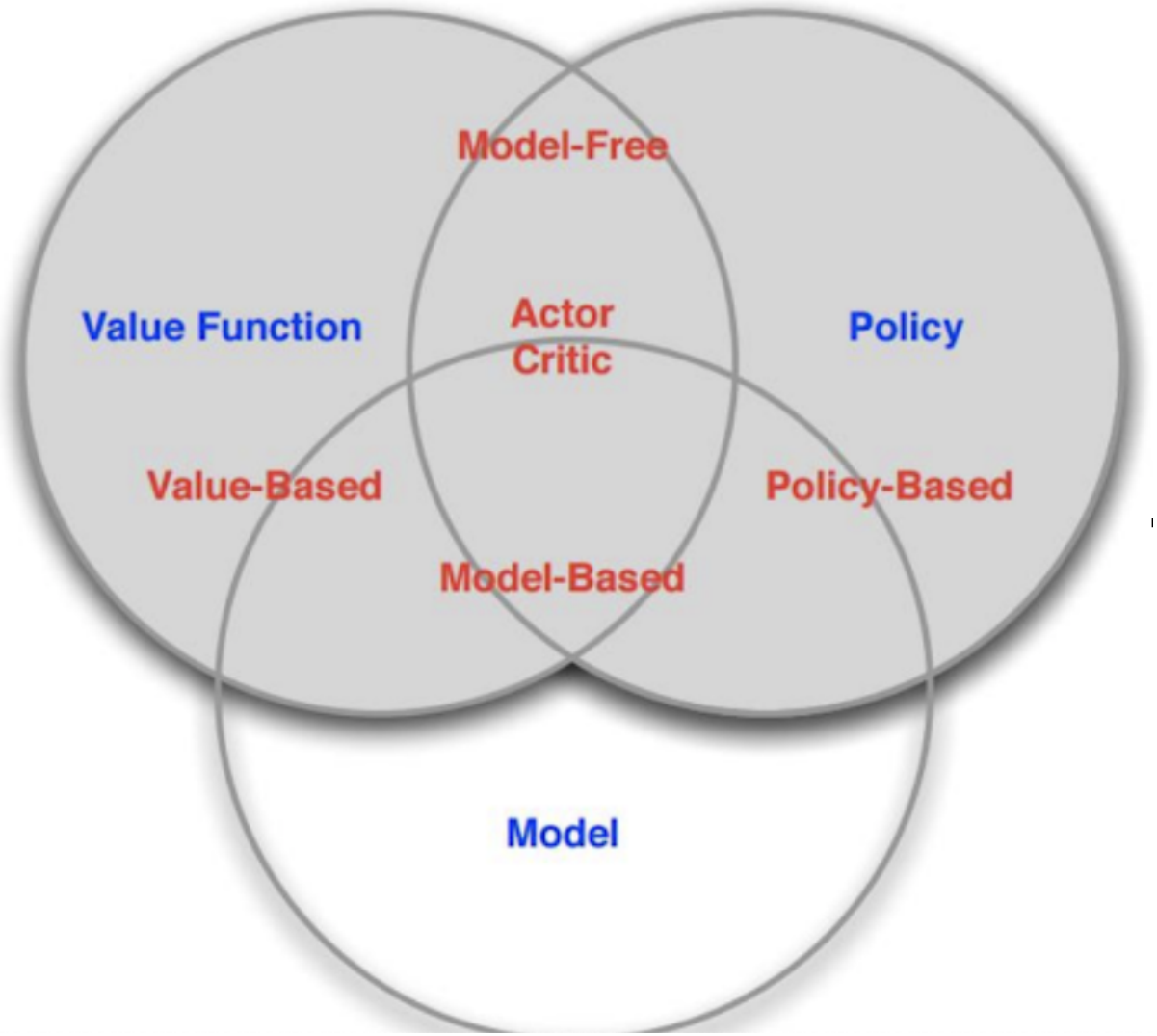
Options

No reward model, no transition model
Either :

- we try to learn the MDP (Plug-in)
- or we ignore it
  - Policy-based
  - Value-based
  - Combination ...

MDP-Unknown - Algorithms & Strategies

Two classes of algorithms



For any policy π and state s, we have

$$V_\pi(s) \doteq \mathbb{E}_\pi\left[\sum_{t\geq 0}\gamma^t R_t \middle| S_0 = s\right] = \underbrace{\mathbb{E}_\pi\left[R_0 \middle| S_0 = s\right]}_{\text{reward at } t=0} + \gamma \underbrace{\mathbb{E}_\pi\left[V_\pi(S_1) \middle| S_0 = s\right]}_{\text{future rewards}}.$$

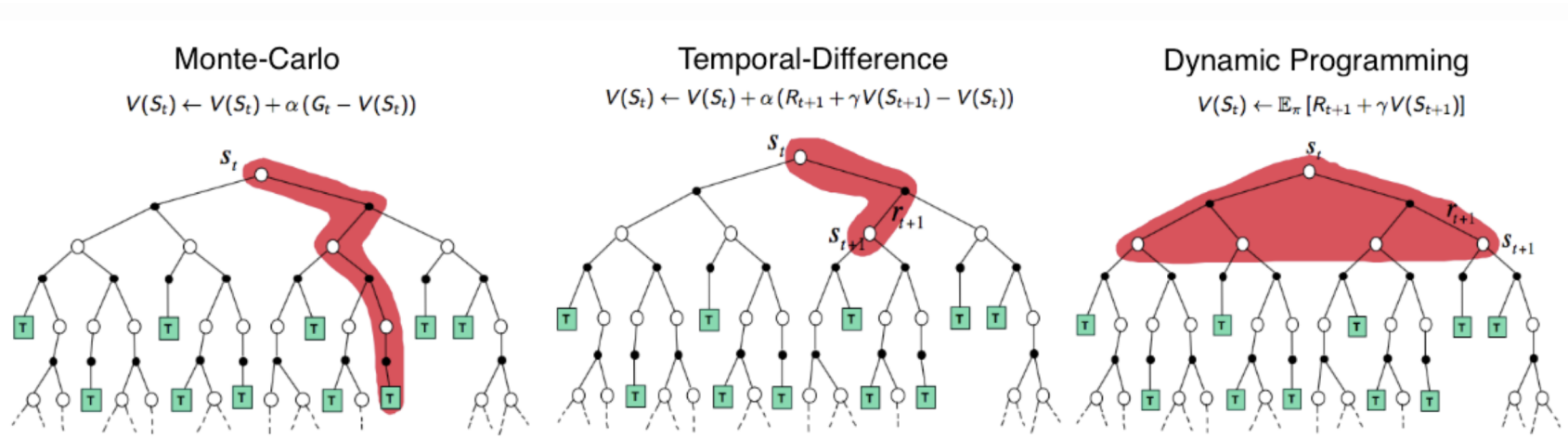
From this equation are derived several algorithms, see Sutton and Barto 2018, most of which consist in estimating V_π or

$$Q_\pi(s,a) \doteq \mathbb{E}_\pi\left[\sum_{t\geq 0}\gamma^t R_t \middle| S_0 = s, A_0 = a\right] \tag{7}$$

either via Temporal-Difference (one-step update), Monte-Carlo evaluation or Dynamic Programming.

MDP-Unknown - Algorithms & Strategies

Value-based algorithms



MDP-Unknown - Algorithms & Strategies

Q-Learning pseudo-code

Most of the value-based algorithms share this structure:

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

possible to learn a parametric version of $Q(S, A)$, Mnih et al. 2013.

? In bandits, we spoke of different strategies to balance exploitation-exploration (ϵ-greedy, EXP3, Follow-The-Leader, UCB, ThompsonSampling, ...) where are they now ?

Consider $\theta \in \Theta$ and let $\pi_\theta = \mathbb{P}_\theta(A|S)$ some parameterized policy.

One could try to find π^* by solving

$$\arg \max_{\theta \in \Theta} \mathbb{E}_{\pi_\theta} \left[\sum_{t \geq 0} \gamma^t R_t \right] \tag{8}$$

and directly derive the policy w.r.t θ.

REINFORCE, a baseline

REINFORCE, A Monte-Carlo Policy-Gradient Method (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, \ldots, T-1$:
    $G \leftarrow$ return from step t (G_t)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t, \boldsymbol{\theta})$

Policy-based algorithms

`Actor-Critic`, current standard

Actor : Related to policy parameterization

Critic : Evaluate state-value function for the policy proposed by actor, in order to update the policy

One-step Actor–Critic (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)

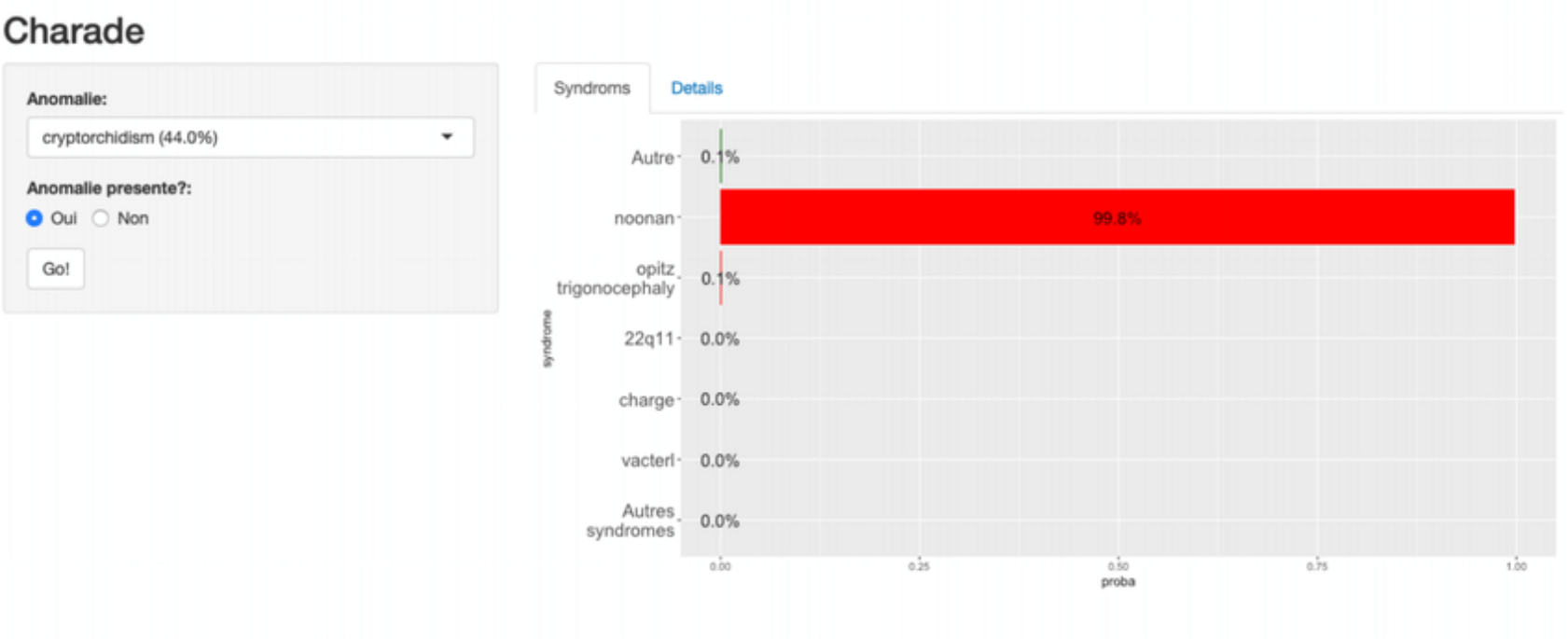
Loop forever (for each episode):
  Initialize S (first state of episode)
  $I \leftarrow 1$
  Loop while S is not terminal (for each time step):
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} I \delta \nabla_{\mathbf{w}} \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Adversarial training

Real-World applications

Real-World applications - Expert recommendation system

Charade



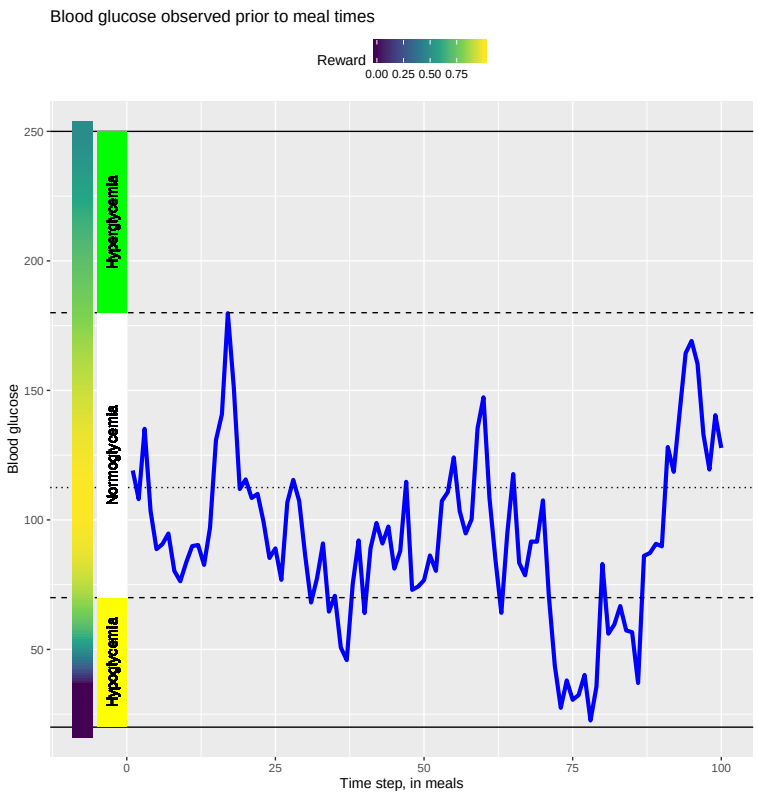
Real-World applications - Expert recommendation system

Now, Sonio

Real-World applications - Insulin management for diabetes

Problem

Consider people with type-I diabetes who self-inject insulin to keep their blood glucose within the range [70 mg/dL, 180 mg/dL]



Based on their meal plan and current blood glucose level, how much bolus insulin should a patient take ?

Literature focuses almost exclusively on continuous injection: insulin pump Daskalaki, Diem, and Mougiakakou 2013; Bothe et al. 2013; Ngo et al. 2018, or long-term prescription levels Javad et al. 2019.

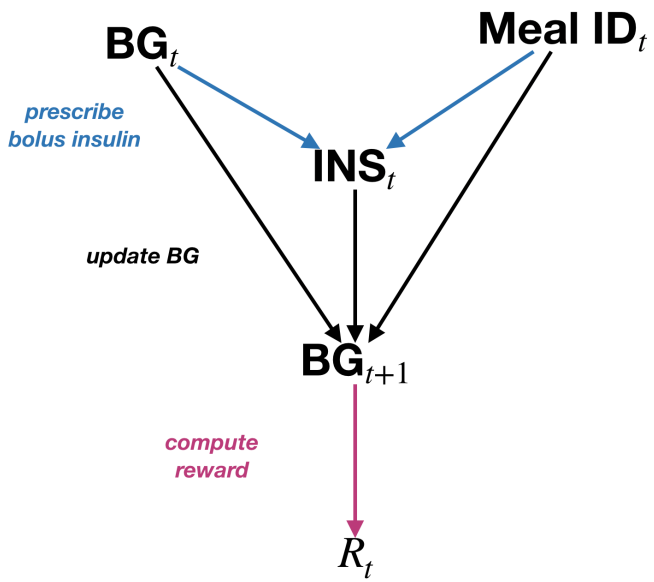
A baseline policy exists.

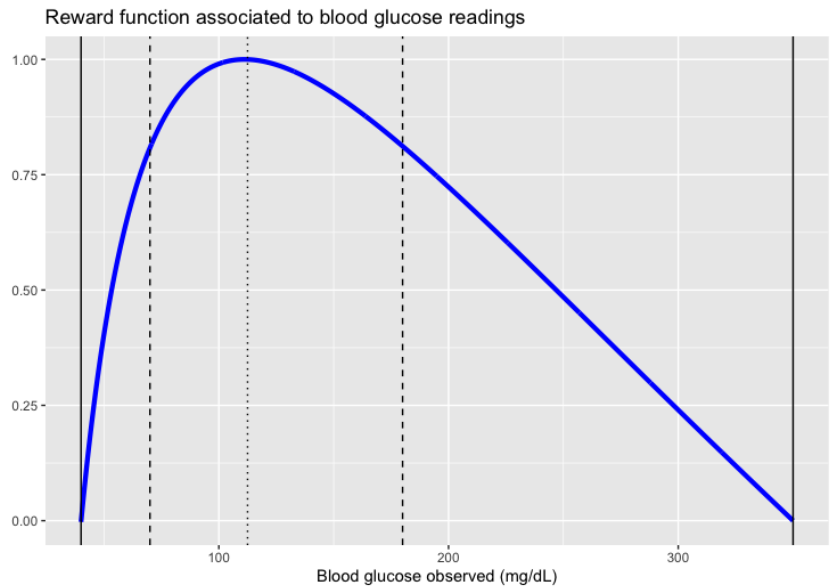
Real-World applications - Insulin management for diabetes

Model

We model this system as a Markov Decision Process, where

- time steps are meal times
- the state (S_t) is the couple (Meal ID$_t$, BG$_t$)
- the action (A_t) is the insulin prescribed
- the reward (R_t) depends on BG_{t+1}, see Kovatchev et al. 2000





Working hypothesis : fixed meal times and quantities.

Note that state information isn't natively discrete. We could make classes, but we chose to adapt our optimization method to the continuous nature of the state.

Approach 1: Q-Learning with basis function approximation

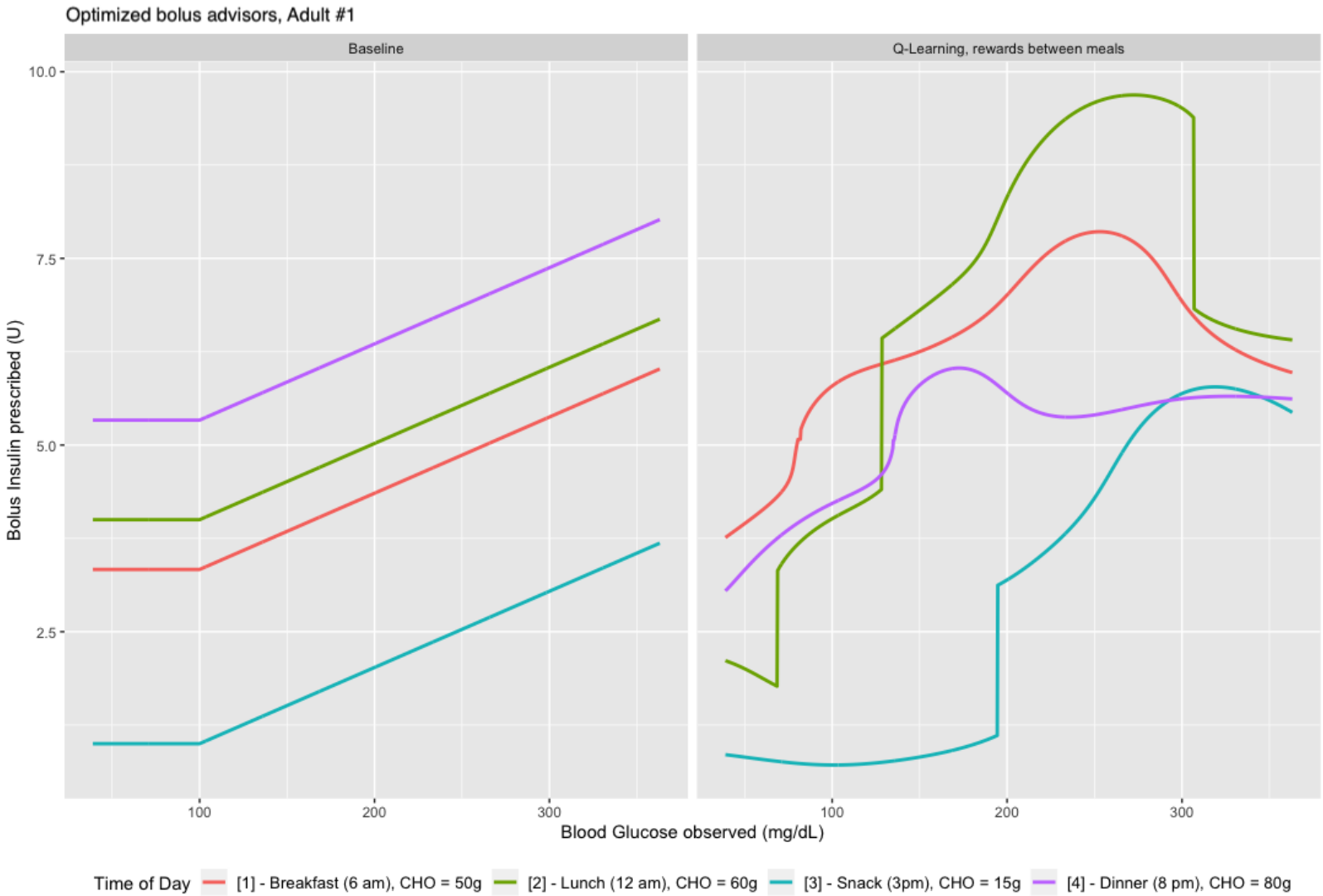
$$Q_\pi(s,a) \approx Q_{\bar{\alpha}}(s,a) = \sum_{b=1}^{B} \alpha_b \phi_b(s,a)$$

where $\bar{\phi} = (\phi_b)_{b=1}^{B}$ is a set of functions.

Approach 2: Deep Q-Network, approximating the Q function with a neural network

Real-World applications - Insulin management for diabetes

Comparing policies



Real-World applications - Insulin management for diabetes

Comparing performances

Indicator	Adult #001		Adult #002		Adult #003	
	π^{base}	$\pi_{\bar{\alpha}^*}$	π^{base}	$\pi_{\bar{\alpha}^*}$	π^{base}	$\pi_{\bar{\alpha}^*}$
Glycemia						
$[40, 70)$	.07	.00	.00	.00	.08	.00
$[70, 112.5)$	.35	.37	.53	.58	.18	.20
$[112, 180)$	.39	.47	.47	.42	.35	.35
$[180, 350)$	.19	.16	.00	.00	.35	.39
$[350, 600]$	.00	.00	.00	.00	.04	.06
Reward						
Q1	.81	.88	.93	.95	.52	.48
Q2	.92	.96	.99	.98	.84	.88
Q3	.97	.99	1.00	.99	.97	.95
Mean	.88	.91	.96	.97	.72	.70

Table 1: Reward and glycemia levels comparison for 3 out of 20 adults in db.

- Optimize jointly meal times and consumption $\Rightarrow$ tested, works !! radical change on the perspective, instead of just trying to correct something, we can pose recommendations on when it would be good to eat or space out meals and everything.
- Integrate stochasticity and more learning in the model $\Rightarrow$ learn next meal times, include probability of skipping next meal time, learn next meal quantity
- Rely on a more sophisticated model of body functions
- Test-out on real humans

Major takeaway : always have a baseline to compare quantitatively and/or qualitatively

Projects

Tackle a decision-making problem with Reinforcement Learning
Group Project (4 people), due date : March 1st 2022

Steps

- Modeling the problem as Bandit / Markov Decision Process
- Data collection and/or simulator
- System visualization
- Learning strategies to define
- Results: hyperparameters that work, optimal policy found, quali/quanti comparison with a baseline policy

Expected output

Report of 2-6 pages (sent by mail to frederic.logemunerel@gmail.com or git) and code used (either shared via email, Colab or git repository). Preferred option is a git repo.

Project suggestions

Games

TicTacToe, Puissance4

Autonomous Driving

Follow-the-line Car; Maze, with traffic

Healthcare

Continuous / Episodic drug prescription

Marketing

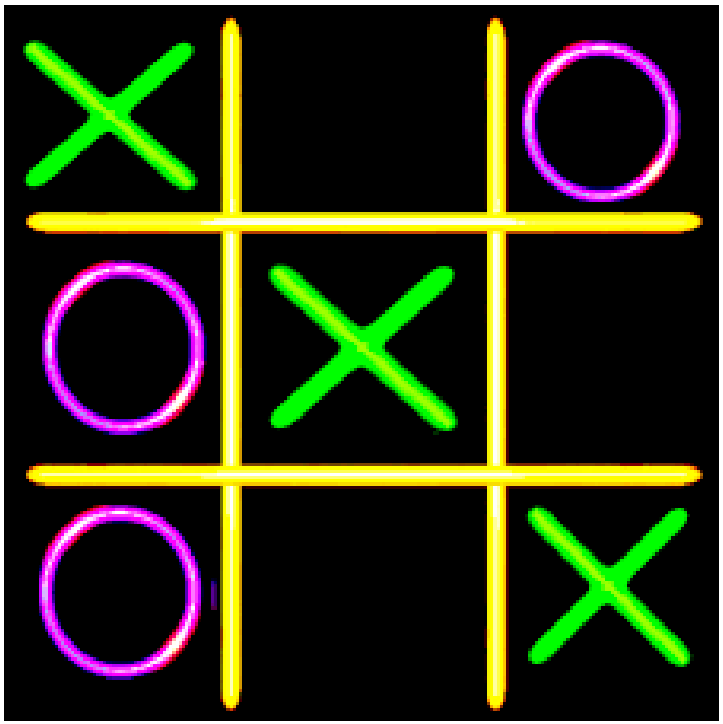
Personalized recommendation, Customer churn

Finance

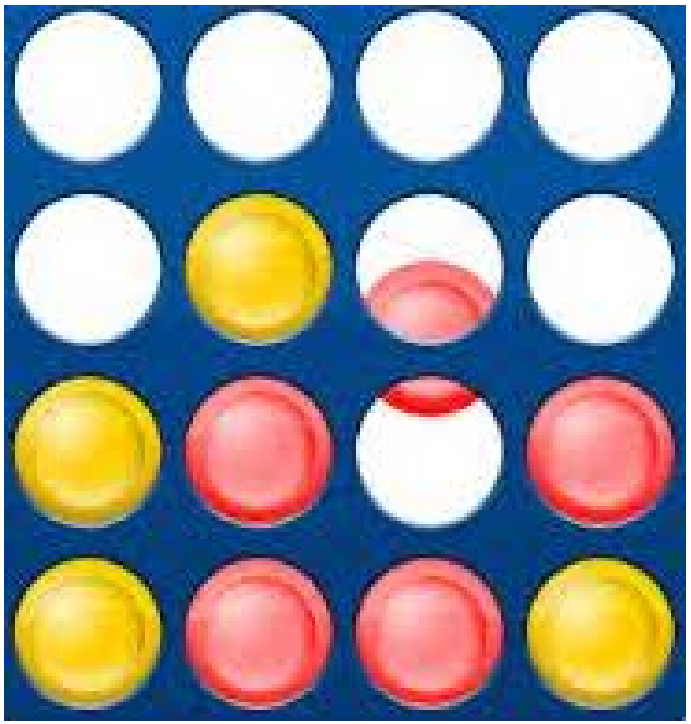
Trading, Dynamic Pricing

... these are only our suggestions, original projects are encouraged !

TicTacToe (a)



Puissance4 (b)

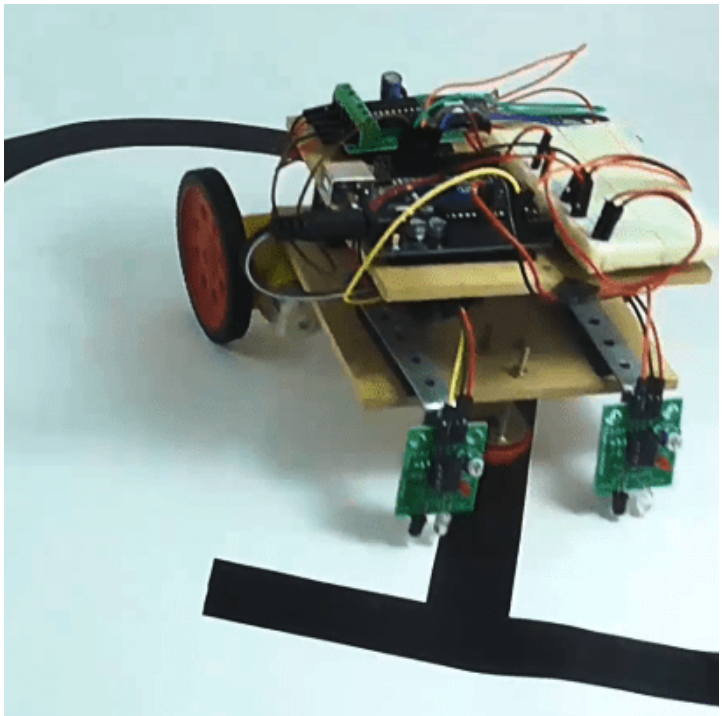


Questions of interest : best first action ? advantage to which player ? when is it close to a tie / not ? Efficient learning via adversarial approach

Projects -

Autonomous Driving

Follow-the-line Car (a)



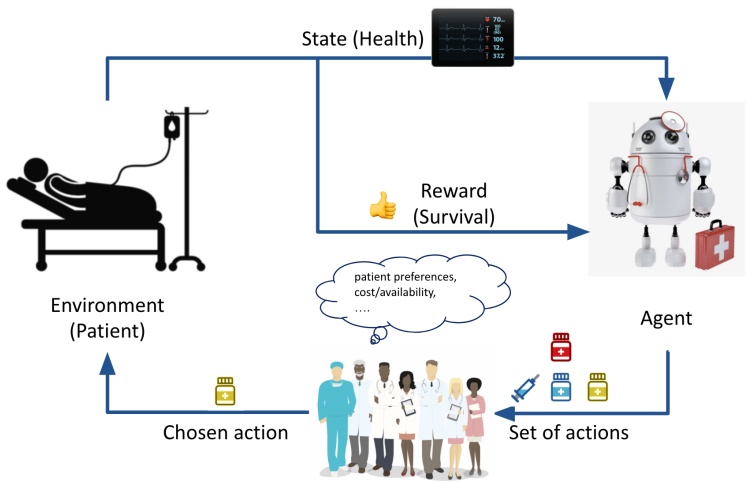
Maze, with traffic issues (b)

Questions of interest : (a) robustness to losing line information ? non-threatening behaviour ? not running over pedestrians ? (b) policy robustness to extreme events ? behaviour under high traffic ?

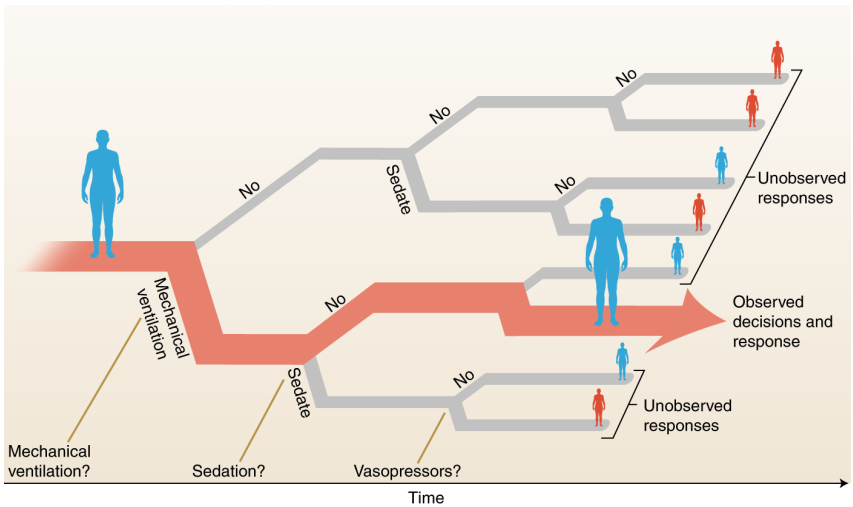
Projects -

Healthcare

Continuous (a)

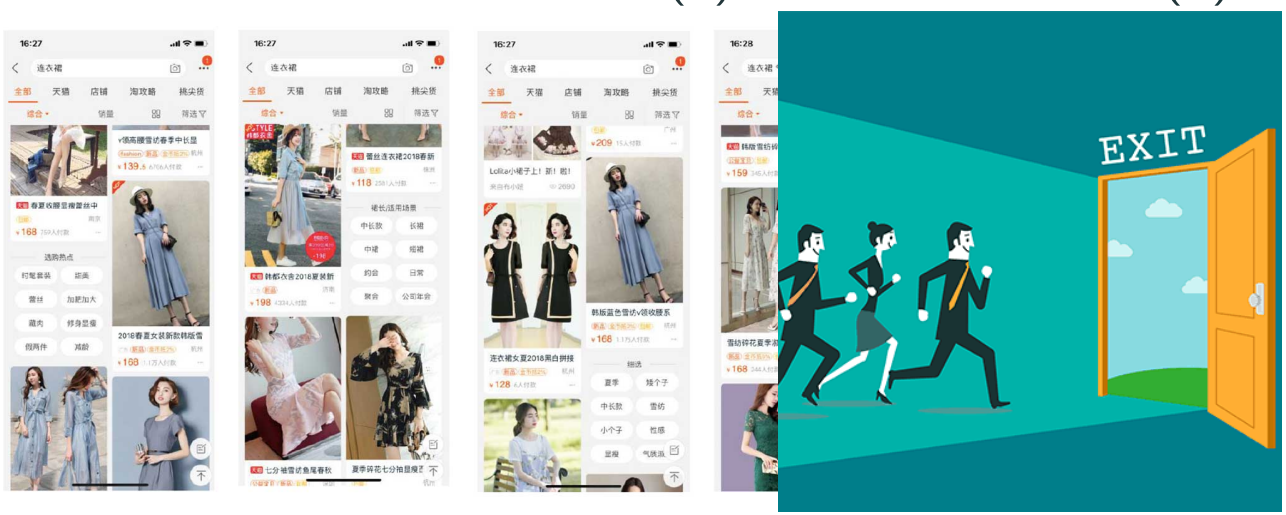


Episodic (b)



Questions of interest : (a) long-term vs mid-term risks ? (b) avoiding all risks, manipulating reward function to prevent bad behaviours

Personalized recommendation (a) Customer churn (b)



Questions of interest : how to simulate accurate interaction with customer ? when is it difficult to propose a pertinent recommendation ?

Personalized recommendation (a)

Customer churn (b)

Dynamic pricing (multiple price point)

Questions of interest : (a) proper simulator of market using real data ? is there any winning strategy ? what timesteps ? (b) parameterized strategy ? beware of customer churn

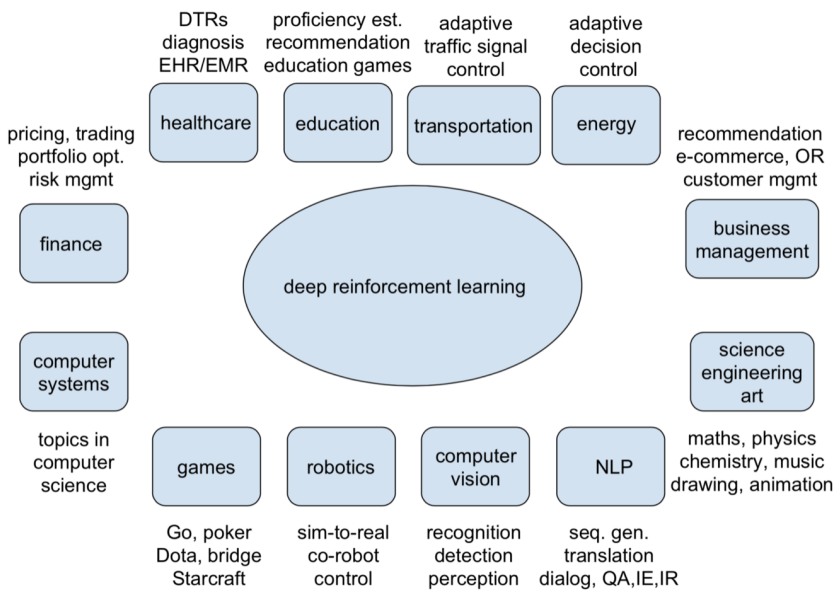
Conclusion

Conclusion -

A focus on applications

Contrary to bandit problems, papers around reinforcement learning focus on application:

- a large set of applications (lot of research communities involved)
- studying such problems is not an easy task
- theoretical study is difficult (Markov chain-like properties)
- hype ...



Yuxi Li, Deep Reinforcement Learning, arXiv, 2018

Conclusion -

Lots of possibilities and limitations

❗ Entry door into decision-making problems for DS

◪ Even if usecase is understood, one still faces:

- Data / sample efficiency issues (exploration / exploitation)
- Optimality guarantees of the solution found ? (deadly triad)
- What does the objective criterion represent, really ?
- What to do when simulations are budgeted ? impossible ?
- Non-stationarity / Adversarial / Unobservable difficult

Conclusion -

Lots of possibilities and limitations

💬 **Best case for RL usecase:**

- free simulations (all possibilities at no cost)
- simple and reliable representation of the world
- deterministic and pure signal
- time to reflect and understand policy behaviour

Sutton and Barto 2018, Lattimore and Szepesvári 2020 (free)

a lot of great lectures in video, slides of David Silver
https://www.davidsilver.uk/teaching/

Richard Sutton, Csaba Szepesvári, David Silvern Tor Lattimore, Alessandro Lazaric, Rémi Munos, ...

The applications and companies drive the research and the applications in this field (DeepMind, Criteo, ...)

Thank you !

Resources & references

References

 Auer, Peter, Nicolo Cesa-Bianchi, and Paul Fischer (2002). "Finite-time analysis of the multiarmed bandit problem". In: *Machine learning* 47.2, pp. 235–256.

 Bothe, Melanie K et al. (2013). "The use of reinforcement learning algorithms to meet the challenges of an artificial pancreas". In: *Expert review of medical devices* 10.5, pp. 661–673.

 Bouneffouf, Djallel, Irina Rish, and Charu Aggarwal (2020). "Survey on Applications of Multi-Armed and Contextual Bandits". In: *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8. DOI: 10.1109/CEC48606.2020.9185782.

 Bubeck, Sébastien and Nicolo Cesa-Bianchi (2012). "Regret analysis of stochastic and nonstochastic multi-armed bandit problems". In: *arXiv preprint arXiv:1204.5721*.

 Daskalaki, Elena, Peter Diem, and Stavroula G Mougiakakou (2013). "An Actor–Critic based controller for glucose regulation in type 1 diabetes". In: *Computer methods and programs in biomedicine* 109.2, pp. 116–125.

 Javad, Mahsa Oroojeni Mohammad et al. (2019). "A Reinforcement Learning–Based Method for Management of Type 1 Diabetes: Exploratory Study". In: *JMIR diabetes* 4.3, e12905.

 Kovatchev, Boris P et al. (2000). "Risk analysis of blood glucose data: a quantitative approach to optimizing the control of insulin dependent diabetes". In: *Computational and Mathematical Methods in Medicine* 3.1, pp. 1–10.

 Lattimore, Tor and Csaba Szepesvári (2020). *Bandit algorithms*. Cambridge University Press.

 Maillard, Odalric-Ambrym, Rémi Munos, and Gilles Stoltz (2011). "A finite-time analysis of multi-armed bandits problems with kullback-leibler divergences". In: *Proceedings of the 24th annual Conference On Learning Theory*. JMLR Workshop and Conference Proceedings, pp. 497–514.

 Mnih, Volodymyr et al. (2013). "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602*.

 Ngo, Phuong D et al. (2018). "Control of blood glucose for type-1 diabetes by using reinforcement learning with feedforward algorithm". In: *Computational and mathematical methods in medicine* 2018.


Puterman, M.L. (2014). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley. ISBN: 9781118625873. URL: https://books.google.fr/books?id=VvBjBAAAQBAJ.


Sutton, Richard S and Andrew G Barto (2018). *Reinforcement learning: An introduction*. MIT press.

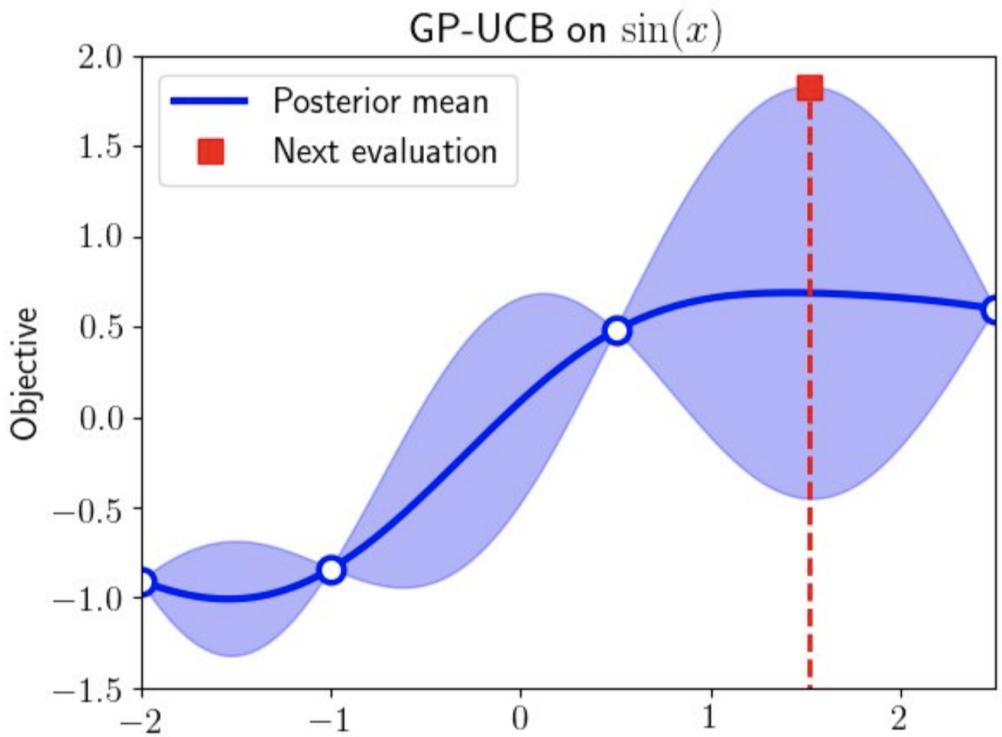
Bonus slides

Bonus slides - Other bandit models

Food for thought

❓ How can we integrate contextual information ? E.g. day of the week

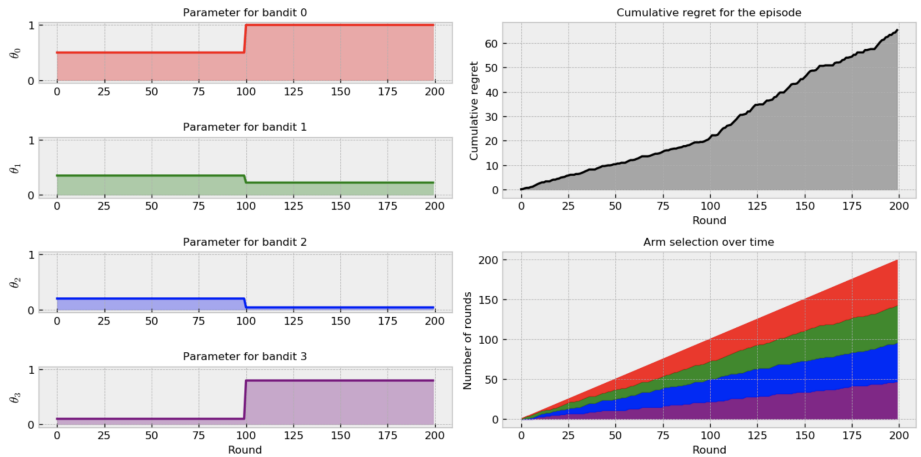
💬 instead of learning $\mu(a)$, we will learn $\mu(a, X_a)$, where X_a is some observed contextual information.



Bonus slides - Other bandit models

Food for thought

❓ What happens if, along the way, $(\mu_a)_a$ changes ? (Non-stationary)



From: https://gdmarmerola.github.io/non-stationary-bandits/

💬 apply changepoint detection, weigh observations by recency, calibrate weigh turnover depending on estimated decay

❓ What happens if the system is controlled by an adversary ?

💬 it depends on the adversary's behaviour – apply EXP3+

Define the nb of samples and average reward for restaurant a at time t as:

$$n_{a,t} = \sum_{t'<t} \mathbb{1}\{a_{t'} = a\} \qquad \hat{\mu}_{a,t} = \frac{\sum_{t'<t} r_{t'} \mathbb{1}\{a_{t'} = a\}}{\sum_{t'<t} \mathbb{1}\{a_{t'} = a\}} \tag{9}$$

Consider the Explore-Then-Commit strategy, extending First-Best :

- sample each restaurant m times (instead of 1) and then
- commit to the best restaurant, $\forall t > m \cdot K$

$$\text{pick } a_t = \arg\max_a \hat{\mu}_{a,m \cdot K}. \tag{10}$$

❓ What is the risk that we will commit to restaurant #1 ?

Bonus slides - Theoretical results on bandit strategies

Explore-Then-Commit - analysis

Assumption : m and $(\mu_a)_{a=1}^K$ allow for Limit Central Theorem application:

$$\frac{\texttt{Binomial}(m, \mu_a) - m\mu_a}{\sqrt{m\mu_a(1-\mu_a)}} \xrightarrow{\mathcal{L}} \texttt{Gaussian}(0,1). \tag{11}$$

$$\begin{aligned}
&\mathbb{P}(\text{"Winner is 1"}) \\
&= \mathbb{P}(\hat{\mu}_{1,2m} > \hat{\mu}_{2,2m}) \\
&= \mathbb{P}(X_1 > X_2) \quad \forall a \;\; X_a \sim \texttt{Binomal}(m, \mu_a), Z_a \sim \texttt{Gaussian}(0,1) \\
&\underset{n,(\mu_a)_a}{\overset{byTCL}{\approx}} \mathbb{P}\left(Z_1\sqrt{m\mu_1(1-\mu_1)} + m\mu_1 > Z_2\sqrt{m\mu_2(1-\mu_2)} + m\mu_2\right) \\
&= \mathbb{P}\left(\texttt{Gaussian}(0,1) > \frac{\sqrt{m}(\mu_2-\mu_1)}{\mu_1(1-\mu_1)+\mu_2(1-\mu_2)}\right)
\end{aligned}$$

❓ What is the best/worst (μ_1, μ_2) ? What is the regret incurred ?

💬 Still linear regret, but risk of occurence is greatly reduced with high m. With an idea of $\mu_2 - \mu_1$ we can make a pertinent stopping rule.

$$\begin{aligned}
&\mathbb{P}(\text{"Winner is 1"}) \\
&\approx \mathbb{P}\left(\texttt{Gaussian}(0,1) > \frac{\sqrt{m}(\mu_2-\mu_1)}{\mu_1(1-\mu_1)+\mu_2(1-\mu_2)}\right) \\
&< \mathbb{P}\left(\texttt{Gaussian}(0,1) > 2\sqrt{m}(\mu_2-\mu_1)\right) \text{ since } \mu_a(1-\mu_a) \leq 1/4 \\
&< C_0 \exp\{-C_1 m(\mu_2-\mu_1)^2\} \text{ by Chernoff bound}
\end{aligned}$$

Hence, in order to control this probability to be of level $\delta \in [0,1]$, arbitrarily small, one needs to take m such that :

$$C_0 \exp\{-C_1 m(\mu_2-\mu_1)^2\} < \delta$$

$$\Rightarrow \quad m > \frac{C_1}{(\mu_2-\mu_1)^2} \log\left(\frac{C_0}{\delta}\right)$$

💬 pick $\delta \approx 1/T$ in order to control regret

Bonus slides - Theoretical results on bandit strategies

Regret higher bounds

See Auer, Cesa-Bianchi, and Fischer 2002, Maillard, Munos, and Stoltz 2011, Lattimore and Szepesvári 2020. In terms of analysis, researchers produce lower and higher bounds for both regret and expected regret:

$$\mathbb{E}(\text{regret}(.,T)) = \sum_{a=1}^{K}(\max_{a'}\mu_{a'} - \mu_a)\mathbb{E}[n_{a,T}] = \sum_{a=1}^{K}\Delta_a\mathbb{E}[n_{a,T}]. \quad (12)$$

One can show that, taking $\text{Regularizer}(n_{a,t}) = 2\sqrt{\log(T)/n_{a,t}}$,

$$\mathbb{E}(\text{regret}(.,T)) \leq C_0\sum_a \Delta_a + \sum_{a\neq a^*}\frac{C_1\log(T)}{\Delta_a}. \quad (13)$$

In the ETC setting, as examined previously, we can show that

$$\mathbb{E}(\text{regret}(.,T)) \leq m\sum_a \Delta_a + (T - mK)\sum_{a\neq a^*}\Delta_a exp\{-\frac{m\Delta_a^2}{4}\} \quad (14)$$

which is hard to balance with m, has we have shown before.

Bonus slides - Theoretical results on bandit strategies

Concentration inequalities

Many different tools for theoretical analysis are available: everything from statistical learning, active learning, probability spaces, etc.

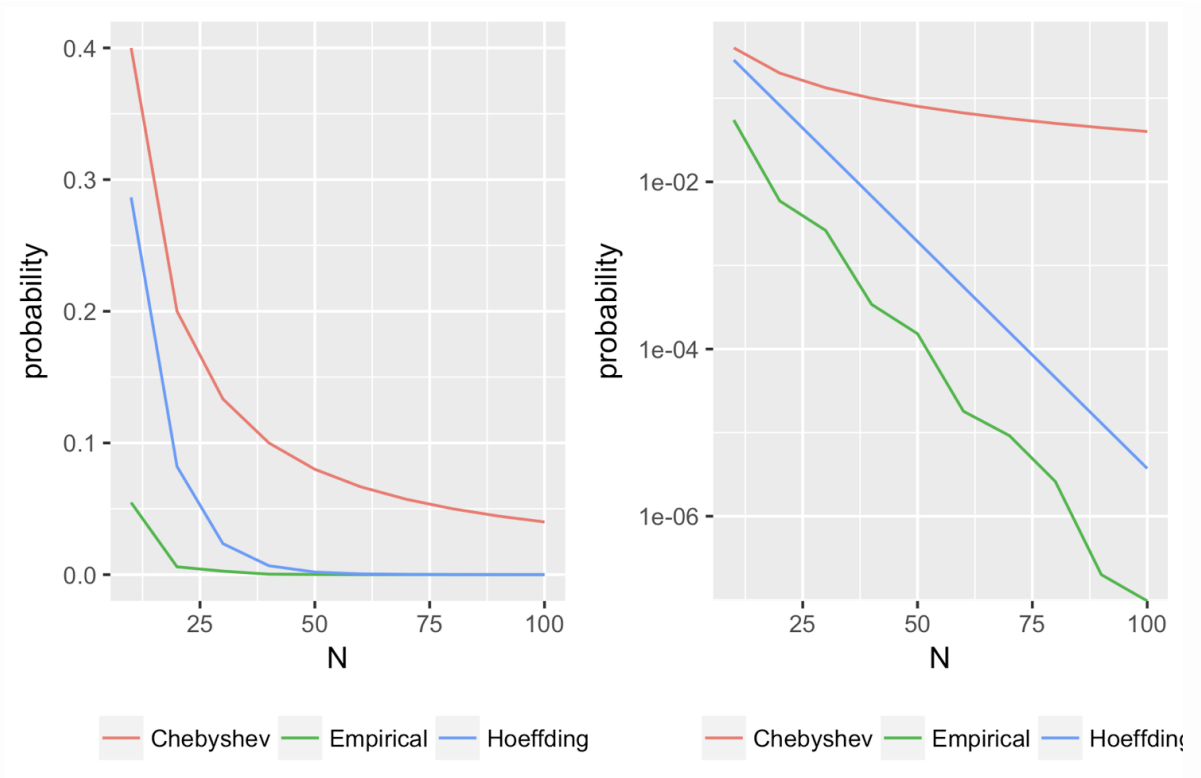
A domain of particular interest, used constantly: concentration inequalities, see chapter 5 of Lattimore and Szepesvári 2020.

- Markov,
- Bienaymé-Tchebychev,
- Hoeffding,
- Cramér-Chernoff,
- Azuma,
- ...

Bonus slides - Theoretical results on bandit strategies

Illustration

Bounding $\mathbb{P}(\texttt{Binomial}(N, p) - Np > N/4)$:



From notebook available at

https://gauss.math.yale.edu/~gcl22/blog/probability/2018/01/07/concentration-inequalities.html

Hoeffding concentration inequality

Consider $(X_t)_{t=1}^T$ a set of independent random variables with support $[a_t, b_t] \in \mathbb{R}^2$ for all t. Writing $S_T = \sum_t X_t$, for any $x > 0$,

$$\mathbb{P}\left(|S_T - \mathbb{E}(S_T)| \geq x\right) \leq 2\exp\left\{-\frac{2x^2}{\sum_t (b_t - a_t)^2}\right\}$$

Reusing our previous notations $\hat{\mu}_{a,t}, n_{a,t}$, from Hoeffding inequality,

$$\mathbb{P}\left(|\hat{\mu}_{a,t} - \mu_a| \geq x\right) \leq 2\exp\left\{-2n_{a,t}x^2\right\} \text{ for any } x > 0. \tag{15}$$

In order to find x such that $\mathbb{P}\left(|\hat{\mu}_{a,t} - \mu_a| \geq x\right) < \delta$, $\delta \in (0,1)$ arbitrarily small, we just need to find x such that $2\exp\left\{-2n_{a,t}x^2\right\} < \delta$ which leads to

$$x > \sqrt{\frac{\log(2/\delta)}{2n_{a,t}}}. \tag{16}$$