# Reinforcement Learning (2)

Maxime Berar

25 septembre 2025

# From Bandits to RL

Bandits : Chosen actions do not affect the distribution of arms, so the current action is independent of future actions. However, the policy determined by the bandit algorithm will change.
$\Rightarrow$ There is no "environment" in a bandit problem. The system can be represented by a unique state.

The concepts of exploration and exploitation, as defined in the context of bandits, will remain relevant to reinforcement learning problems.

# Markov Decision Process

tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$ with

- $\mathcal{S}$ set of states
- $\mathcal{A}$ set of actions
- $\mathcal{R}$ set of rewards

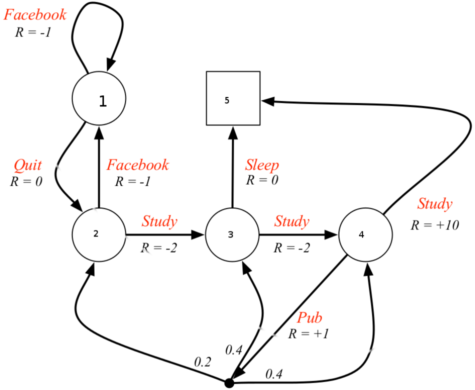
all of finite number of elements.

At time $t-1$ in state $S_{t-1}$ ($\in \mathcal{S}$) after choosing action $A_{t-1}$ ($\in \mathcal{A}$), the agent is in state $S_t$ ($\in \mathcal{S}$) and receives reward $R_t$ ($\in \mathcal{R}$).

### Assumption

$R_t$ and $S_t$ have well defined discrete probability distribution **dependent only on the preceding state and action**

# Example, student dilemna (from D. Silver)




- 5 states $\{S_1, S_2, S_3, S_4, S_5\}$
- 1 final state $(S_5)$
- 5 actions $\{F, Q, St, Sl, P\}$
- a model
  - $p(2|1, Q) = 1$,
  - $p(2|4, P) = 0.2$,

    ...
  - $r(2|1, Q) = 0$,
  - $r(2|4, P) = 1$,

    ...
- Episodes

# Model of the environment

$p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ is an ordinary deterministic function

$$p(s', r|s, a) \doteq \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$$

$p$ specifies a probability distribution for each choice of $s$ and $a$

$$\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} p(s', r|s, a) = 1$$

From it, one can compute anything else one might want to know about the environment.

- state-transition probabilities $p(s'|s, a) \doteq \sum_{r \in \mathcal{R}} p(s', r|s, a)$
- expected rewards for state-action pairs $r(s, a) \doteq \mathbb{E}[R_t|S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a)$
- expected rewards for state-action-next-state triples $r(s, a, s') \doteq \mathbb{E}[R_t|S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in \mathcal{R}} r \frac{p(s', r|s, a)}{p(s'|s, a)}$

# Reward hypothesis

Maximizing the cumulative reward in the long run ; not the immediate reward.

- Return, over an episode

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T$$

- Return, for continued tasks, weighted cumulative reward :

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

  where $\gamma$ is a parameter, $0 \leq \gamma \leq 1$, called the discount rate.

Return recursion property

$$G_t = R_{t+1} + \gamma G_{t+1}$$

If the reward is a constant $+1$, what is the return ?
Suppose $\gamma = 0.5$ and the following sequence of rewards is received $R_1 = -1, R_2 = 2, R_3 = 6, R_4 = 3, R_5 = 2$, with $T = 5$. What are $G_0, G_1, \cdots, G_5$ ?

# Policies and Value Functions

*We have yet to model how to choose an action.*

Formally, a policy is a mapping from states to probabilities of selecting each possible action. If the agent is following policy $\pi$ at time $t$, then $\pi(a|s)$ is the probability that $A_t = a$ if $S_t = s$.

- value function,
  expected return from state $s$ following policy $\pi$ :

$$v_\pi(s) = \mathbb{E}_\pi \left[ G_t | S_t = s \right]$$

- value function of an action
  expected return from state $s$, choosing action $a$, following policy $\pi$ :

$$q_\pi(s, a) = \mathbb{E}_\pi \left[ G_t | S_t = s, A_t = a \right]$$

# Toward Bellman Equation

$$v_\pi(s) = \mathbb{E}_\pi\left[G_t | S_t = s\right]$$

$$\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi\left[G_t | S_t = s\right] \\
&= \mathbb{E}_\pi\left[R_{t+1} + \gamma G_{t+1} | S_t = s\right] \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | a, s) \left(r + \gamma \mathbb{E}_\pi\left[G_{t+1} | S_{t+1} = s'\right]\right) \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | a, s) \left(r + \gamma v_\pi(s')\right)
\end{aligned}$$

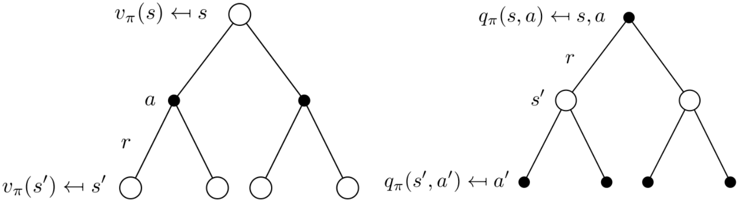
idem for $q_\pi(s, a)$

$$q_\pi(s, a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a\right]$$

# Bellman equations in graph form

- Recursive decomposition of the value functions

$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s\right]$$




$$q_\pi(s,a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a\right]$$

# Solutions of Bellman equations ?

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{r,s'} p(s', r|s, a) \left[ r(s, a) + \gamma v_\pi(s') \right]$$

$$v_\pi(s) = \sum_{r,s'} \sum_a \pi(a|s) p(s', r|s, a) r(s, a) + \gamma \sum_{r,s'} \sum_a \pi(a|s) p(s', r|s, a) v_\pi(s')$$

Known elements : Environment model $p(s', r|s, a)$ , policy $\pi(a|s)$

- state transition according to a policy

$$p_\pi(s, s') = \sum_{a \in \mathcal{A}} \pi(a|s) p(s'|a, s) = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{r \in \mathcal{R}} p(s', r|a, s)$$

- reward $r(s, a) = \sum_{r \in \mathcal{R}} \sum_{s' \in \mathcal{S}} r(s, a) p(s', r|s, a) \pi(a|s)$
- reward according to a policy $r_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) r(s, a)$

Bellman equation for the value-function

$$v_\pi(s) = r_\pi(s) + \gamma \sum_{s' \in \mathcal{S}} p_\pi(s, s') v_\pi(s')$$

# Solutions of Bellman equations according to a policy

$$v_\pi(s) = r_\pi(s) + \gamma \sum_{s' \in \mathcal{S}} p_\pi(s, s') v_\pi(s')$$

in matrix-vector form

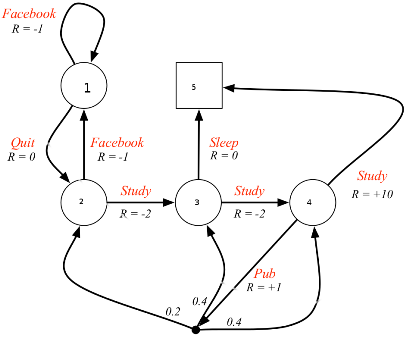
$$\mathbf{v}_\pi = \mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v}_\pi$$

linear system

$$(\mathbf{I}_d - \gamma \mathbf{P}_\pi)\, \mathbf{v}_\pi = \mathbf{r}_\pi$$

Problem when there is a big number of states (e.g. bachgammon ($10^{20}$))

# Student Example with Uniform policy




```
>> gamma=1;
>> P=[.5 .5 0 0 0 ;
   .5  0 .5 0 0 ; 0 0 0 .5 .5 ; 0 .1 .2 .2 .5 ; 0 0 0 0 0]

P =

    0.5000    0.5000         0         0         0
    0.5000         0    0.5000         0         0
         0         0         0    0.5000    0.5000
         0    0.1000    0.2000    0.2000    0.5000
         0         0         0         0         0

>> R=[-0.5 -1.5 -1 5.5 0]'

R =

   -0.5000
   -1.5000
   -1.0000
    5.5000
         0

>> v=(eye(5)-gamma*P)\R;
>> v=(eye(5)-gamma*P)\R

v =

   -2.3077
   -1.3077
    2.6923
    7.3846
         0
```

# Optimal value function & policy

- $v_*(s) = \max_\pi v_\pi(s)$
- $q_*(s,a) = \max_\pi q_\pi(s,a)$

best performance of a MDP. The problem is solved if the functions are known.

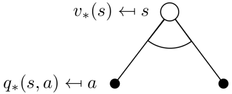
- It is possible to define a partial order of the policies : $\pi \geq \pi'$ if $v_\pi(s) \geq v_{\pi'}(s)\ \forall s$
- Theorem : for all MDP
  - there exists an optimal policy $\pi_*$ better or equal of all others
  - All optimal policies enable optimal function values, i.e. $v_{\pi_*}(s) = v_*(s)$
  - i.e. $q_{\pi_*}(s,a) = q_*(s,a)$
- An optimal policy can be obtained after finding $q_*(s,a)$ with :

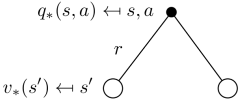
$$\pi_*(a,s) = 1 \text{ if } a = \arg\max_{a \in \mathcal{A}} q_*(s,a)\ ,\ 0 \text{ else}$$

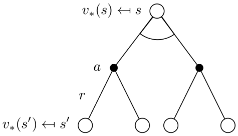
# Optimality equations of Bellman

The optimal policy should always consider the best action

$$
\begin{aligned}
v_*(s) = \max_\pi v_\pi(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi^*}(s, a) \\
&= \max_{a \in \mathcal{A}(s)} \mathbb{E}_{\pi^*} \left[ G_t | S_t = s, A_t = a \right] \\
&= \max_{a \in \mathcal{A}(s)} \mathbb{E}_{\pi^*} \left[ R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a \right] \\
&= \max_{a \in \mathcal{A}(s)} \sum_{s', r} p(s', r | s, a) \left[ R_{t+1} + \gamma v_*(S_{t+1}) \right]
\end{aligned}
$$




no simple solutions $\rightarrow$ iteratives methods.

# Dynamic Programming

How to compute the optimal value functions $v_*$, $q_*$ ?

$$v_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_*(S_{t+1})\right]$$

$$q_*(s, a) = \sum_{s',r} p(s', r|s, a) \left[r + \gamma \max_{a'} q_*(s', a')\right]$$

in order to find the optimal policy ?

# Policy evaluation

Remember that

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|a, s) \left(r + \gamma v_\pi(s')\right)$$

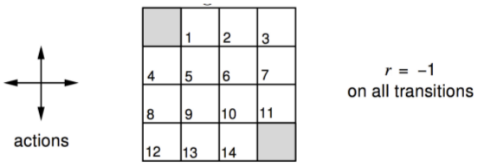
iterative view of the problem : searching for a fixed point in the update rule

$$v_\pi^{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|a, s) \left(r + \gamma v_\pi^k(s')\right)$$

## Iterative policy evaluation

input: $\pi$
$\mathbf{v} = 0_{|\mathcal{S}|}$
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$ :
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

# Example




- $\gamma = 1$
- 14 non-terminal states
- 1 terminal state (2 boxes)
- Actions that leave the grid $\rightarrow$ same state
- Reward is -1 for all non-terminal state, 0 else
- uniform policy (up,down,left,right)

# Example (cont.)

$k = 0$

| | | | |
|---|---|---|---|
| 0.0 | 0.0 | 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 |

$k = 1$

| | | | |
|---|---|---|---|
| 0.0 | -1.0 | -1.0 | -1.0 |
| -1.0 | -1.0 | -1.0 | -1.0 |
| -1.0 | -1.0 | -1.0 | -1.0 |
| -1.0 | -1.0 | -1.0 | 0.0 |

$k = 2$

| | | | |
|---|---|---|---|
| 0.0 | -1.7 | -2.0 | -2.0 |
| -1.7 | -2.0 | -2.0 | -2.0 |
| -2.0 | -2.0 | -2.0 | -1.7 |
| -2.0 | -2.0 | -1.7 | 0.0 |

$k = 3$

| | | | |
|---|---|---|---|
| 0.0 | -2.4 | -2.9 | -3.0 |
| -2.4 | -2.9 | -3.0 | -2.9 |
| -2.9 | -3.0 | -2.9 | -2.4 |
| -3.0 | -2.9 | -2.4 | 0.0 |

$k = 10$

| | | | |
|---|---|---|---|
| 0.0 | -6.1 | -8.4 | -9.0 |
| -6.1 | -7.7 | -8.4 | -8.4 |
| -8.4 | -8.4 | -7.7 | -6.1 |
| -9.0 | -8.4 | -6.1 | 0.0 |

$k = \infty$

| | | | |
|---|---|---|---|
| 0.0 | -14. | -20. | -22. |
| -14. | -18. | -20. | -20. |
| -20. | -20. | -18. | -14. |
| -22. | -20. | -14. | 0.0 |

# Policy improvement

How to improve a policy ?
If $v_\pi$ has been evaluated, then $q_\pi$ is known :

$$q_\pi(s, a) = \sum_{s', r} p(s', r | s, a) \left[ r + \gamma v_\pi(s') \right]$$

Question is any $q_\pi(s, a)$ greater than $v_\pi(s)$ ? In this case, a good policy would be to choose the corresponding action.

$$\pi'(s) \doteq \arg\max_a q_\pi(s, a)$$

The greedy policy takes the action that looks best in the short term after one step of lookahead.

$$\pi'(s) \doteq \arg\max_a \sum_{s', r} p(s', r | s, a) \left[ r + \gamma v_\pi(s') \right]$$

# Policy iteration (alg.)

`1.Initialisation`

$V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrary

`2. Policy evaluation`

`3. Policy improvement`

*policy-stable* $\leftarrow$ `true`

`for each` $s \in \mathcal{S}$ : *old-action* $\leftarrow \pi(s)$

$\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\,[r + \gamma v_\pi(s')]$

`If` *old-action* $\neq \pi(s)$, `then` *policy-stable* $\leftarrow$ `false`

`If` *policy-stable*, `then stop and return` $V \approx v_*$ and $\pi \approx \pi_*$ ;

`else go to 2.`

# Value iteration

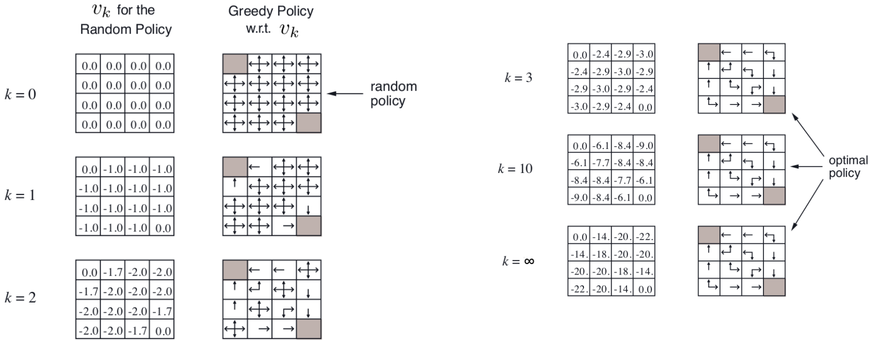
Drawback of policy iteration is that at each iteration, the policy must be evaluated.
However, as $v* \Leftrightarrow \pi^*$ can we work only with the value function ?

## Value iteration

initilalize $V$
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$ :
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output a deterministic policy

$$\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$$

# Back to example




Remark 1 : after 3 iterations, the policy found by greedy amelioration is optimal

Remark 2 : It always converges to $\pi_*$

Remark 3 : if $\pi' = \pi$ then $\pi = \pi*$

# Starting from the end time ?

If this is a finite horizon problem with known rewards for the final set of actions and known transition probability matrices at all times, we can learn the policy backward.

**Backward recursion**
From the final states to the actions

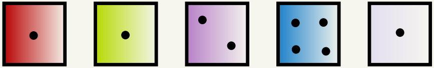
$$q(s,a)_{T-1} = \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | a, s) r$$

From $q(s,a)_{T-1}$ to $v(s)_{T-1}$

$$v(s)_{T-1} = \max_{a \in \mathcal{A}} q(s,a)_{T-1}$$

# Yahtzee example 1/3

naive view



equiprobables states : 6**5=7776

action : keep or reroll each dice 2**5=32

non-equiprobable states : 252

| | | | | | |
|---|---|---|---|---|---|
| 3 | 1 | 0 | 1 | 0 | 0 |

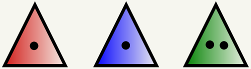
action between "keep" and "re-roll".

The number of actions depends on the state

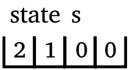
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 4 | 2 | 1 | 2 | 1 | 1 | = 16 |
| keep | 3 | 1 | 0 | 1 | 0 | 0 | all |
| keep | 3 | 1 | 0 | 0 | 0 | 0 | roll 1 |
| keep | 3 | 0 | 0 | 1 | 0 | 0 | roll 1 |
| keep | 2 | 1 | 0 | 1 | 0 | 0 | roll 1 |
| keep | 3 | 0 | 0 | 0 | 0 | 0 | roll 2 |
| | | | ⋮ | | | | |
| reroll | 0 | 0 | 0 | 0 | 0 | 0 | all |

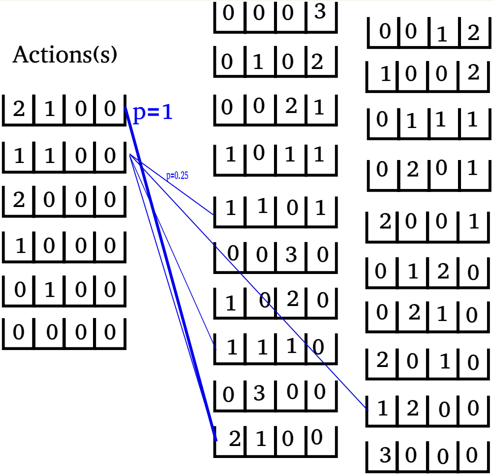
# Yahtzee example 2/3

working view



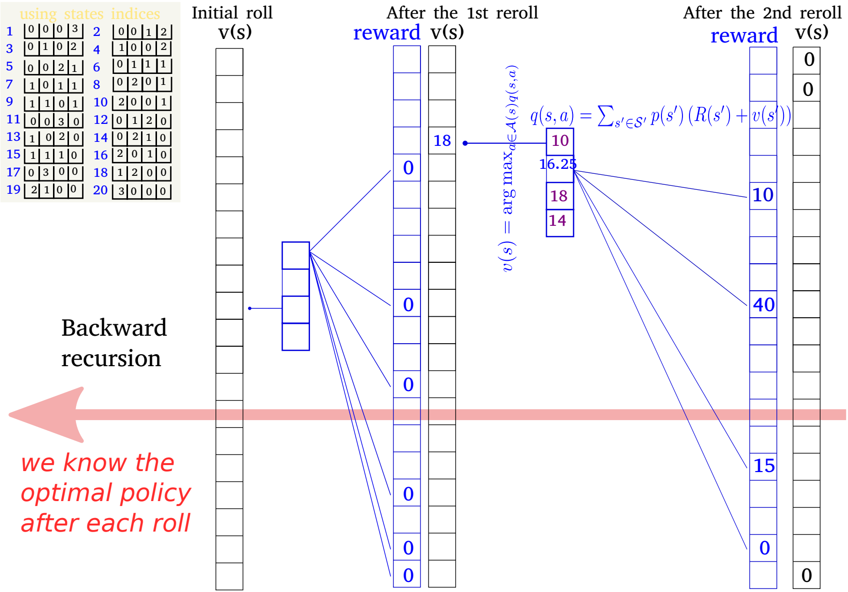
3 dices with 4 faces
20 (histogram) states



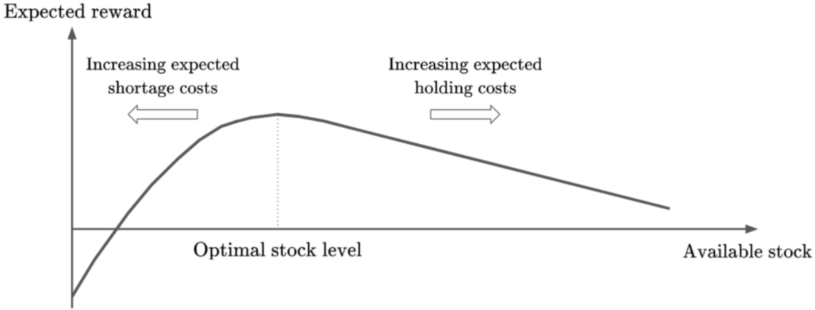

From a given state s, choosing action a among Actions(s) we could attain states s' with probability p, which is known.

# Yahtzee example 3/3



# RL for the MOQ problem 1/3

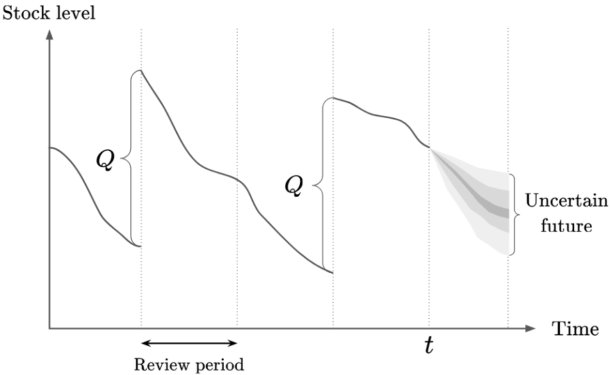
Learning to order quantity / manage stock level for multiple items under uncertainty and minimum-order quantity.




Thèse de G. Deletoile

# RL for the MOQ problem 2/3

Learning to order items quantity / manage stock levels for multiple items under uncertainty and minimum-order quantity.




Thèse de G. Deletoile

# MOQ Problem equations

First, we need to bound the maximum stock and orders.
For a single item,

- $x$ stock level (state)
- $a$ reorder quantity (action)
- $r$ reward equation
  $d$ demand at time $t$ with a $h$ holding penalty, a $m$ shortage penalty and a reodering cost $c(a_t)$.

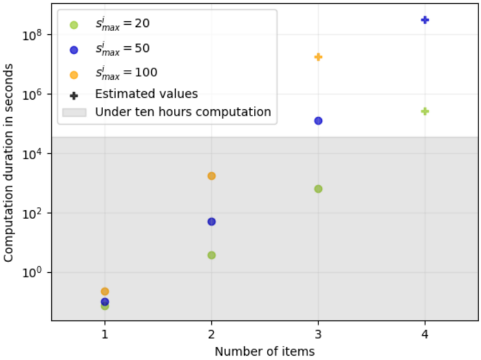
$$x_{t+1} = x_t + a_t - d_t$$

$$r_t = c(a_t) + m[x_t + a_t - d_t]^- - h[x_t + a_t - d_t]^+$$

$$c(a) = \begin{cases} 0 & \text{if } a = 0 \\ K + ca & \text{if } a > 0 \end{cases}$$

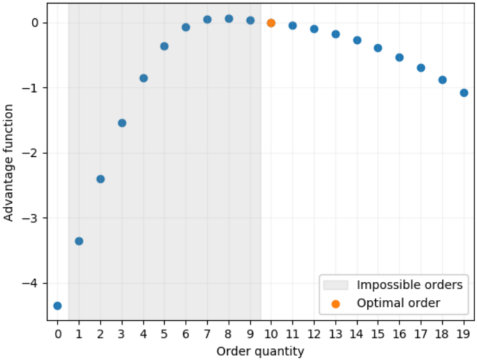
Complexity of multi-item scenari

# Backward recursion ?



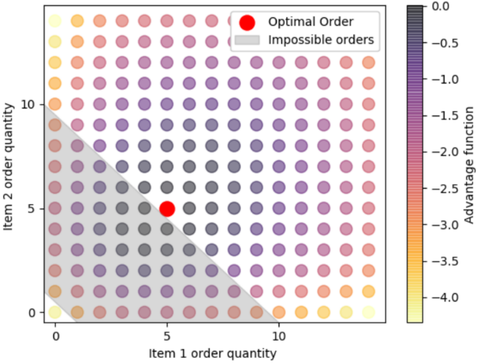

Thèse de G. Deletoile

# Results -Advantage Function



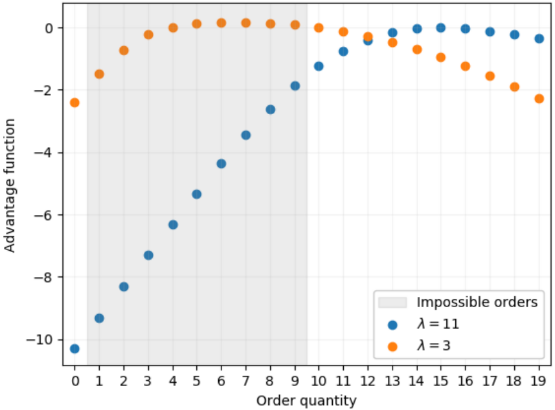

Thèse de G. Deletoile

# Results - Action Space




Thèse de G. Deletoile

# Results - Poisson demand




Thèse de G. Deletoile