Reinforcement Learning Tutorial

Bilal Piot, Corentin Tallec

July 2022

What is Reinforcement Learning?

What is RL?

A general paradigm for **learning** via **trial and error**, when an agent **interacts** with an environment and seeks to **maximize a reward signal over time**.

When should I use RL?

Extremely general framework in its formulation.

Pros:

- Most problems can be formalized as RL problems.
- Interesting paradigm when very little hypothesis can be made about the problems being tackled.

Cons:

- Makes very little hypothesis about the underlying problem.
- Nearly always applicable, rarely the best solution.

Successes of reinforcement learning

Applied to various domains

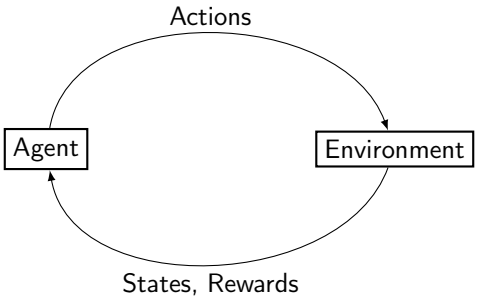
- Board games,
- Video games,
- Robotics, continuous control.

Key ingredient: possibility to simulate the environment at a (relatively) low cost.
So far has shown state of the art results in:

- BackGammon, Atari games, Go, Chess, Shogi, StarCraft II, Dota 2....

Markov Decision Processes and Policies

Reinforcement Learning: High level view



Markov Decision Process (MDP)

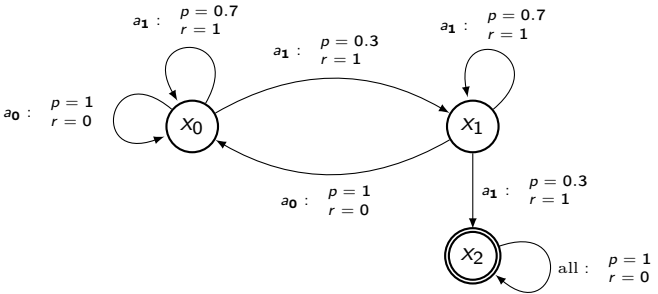
Formally, a finite Markov Decision Process (MDP) is a tuple $(\mathcal{X}, \mathcal{A}, p, \rho, r, \gamma)$ with:

- $\mathcal{X}$: the finite state space of cardinal $|\mathcal{X}|$.
- $\mathcal{A}$: the finite action space of cardinal $|\mathcal{A}|$.
- p: the dynamics is a transition kernel from $\mathcal{X} \times \mathcal{A}$ to $\mathcal{X}$. $p(y|x, a)$ is the probability to transition to state $y \in \mathcal{X}$ by doing action $a \in \mathcal{A}$ in state $x \in \mathcal{X}$.
- $\rho \in \mathbb{R}^{\mathcal{X}}$: the initial state distribution and $\rho(x)$ is the probability to start in state x.
- r: the reward function is a mapping from state-action to real number, $r \in \mathbb{R}^{\mathcal{X} \times \mathcal{A}}$. The reward $r(x, a)$ represents the local benefit of doing action $a \in \mathcal{A}$ in state $x \in \mathcal{X}$.
- $\gamma \in [0, 1)$: the scalar discount factor.

Intuitions on MDPs

- **Markov assumption:** Next state and reward only depend on the current state and action. This implies:
  - States are fully representative of the environment. There are no hidden variables.
  - States provide all the information required to select actions. No need to remember the past.
- **Sequential decision making**: Action not only influence immediate reward, but also future state. Need to *plan* several steps ahead.

- **Actions**: 0. charge battery 1. look for garbage
- **States**: 0. high battery 1. low battery 2. battery depleted



Reinforcement Learning Goal

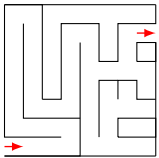
The goal of Reinforcement Learning algorithms is to find a **policy** (a behaviour) that maximises a notion of **cumulative return**.

Goal of the GarbageBot

For the GarbageBot, the goal is to find a **policy** that allows it to look for garbage as long as possible without depleting the battery.

Quizz time: Is this an MDP? (1)

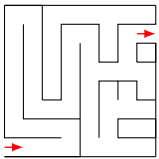
Maze environment
Fixed maze template, agent starts in the bottom left corner and must go to the top right corner.



- **Reward:** $+1$ when getting out of the maze.
- **Actions:** One of the four directions.
- **States:** Position in the maze

Quizz time: Is this an MDP? (1)

Maze environment
Fixed maze template, agent starts in the bottom left corner and must go to the top right corner.



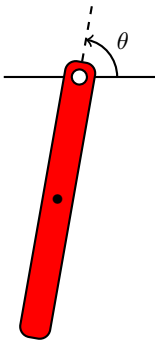
- **Reward:** $+1$ when getting out of the maze.
- **Actions:** One of the four directions.
- **States:** Position in the maze

Yes

Quizz time: Is this an MDP? (2)

Pendulum environment:

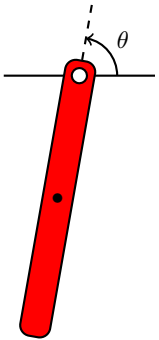
- **Reward:** $+1$ if pendulum is upward, 0 otherwise.
- **Actions:** Torque $+1$ or -1 (i.e. angular acceleration to the left or to the right).
- **States:** Angle θ and angular velocity $\dot{\theta}$.



Quizz time: Is this an MDP? (2)

Pendulum environment:

- **Reward:** $+1$ if pendulum is upward, 0 otherwise.
- **Actions:** Torque $+1$ or -1 (i.e. angular acceleration to the left or to the right).
- **States:** Angle θ and angular velocity $\dot{\theta}$.

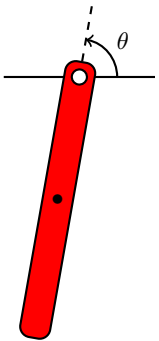


Yes

Quizz time: Is this an MDP? (3)

Pendulum environment:

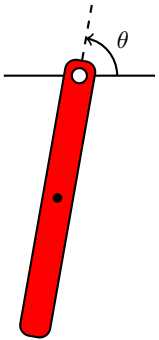
- **Reward:** $+1$ if pendulum is upward, 0 otherwise.
- **Actions:** Torque $+1$ or -1 (i.e. angular acceleration to the left or to the right).
- **States:** Angle θ.



Quizz time: Is this an MDP? (3)

Pendulum environment:

- **Reward:** $+1$ if pendulum is upward, 0 otherwise.
- **Actions:** Torque $+1$ or -1 (i.e. angular acceleration to the left or to the right).
- **States:** Angle θ.



No

Policy

We consider *Markovian* stochastic policies π, i.e. state dependent probability distributions over actions. $\pi(a|x)$ denotes the probability of choosing action $a \in \mathcal{A}$ in state $x \in \mathcal{X}$. The set of markovian stochastic policies is denoted by Π.

Intuitions on the policy

- A policy defines what action a the agent will perform in state x.
- The action selection process can incorporate some randomness: a policy associates a **distribution over actions** $\pi(\cdot|x)$ to each state x.
- Deterministic policies can be recovered when $\forall x, \exists a \text{ s.t. } \pi(a|x) = 1$.

Trajectories generated by a policy π

We define recursively trajectories $\tau^\pi = (X_t, A_t, R_t)_{t\in\mathbb{N}^+}$ as:

$$X_0 \sim \rho$$
$$\forall t \geq 0, \quad A_t \sim \pi(.|X_t), \quad R_t = r(X_t, A_t), \quad X_{t+1} \sim p(.|X_t, A_t)\cdot$$

$\mathcal{T}(\pi)$ is the distribution of such trajectories, and $\mathcal{T}(x, \pi)$ (resp. $\mathcal{T}(x, a, \pi)$) is the distribution of trajectories conditioned on $X_0 = x$ (resp. $X_0 = x$, $A_0 = a$).

Intuitions on the trajectory

Informally, a trajectory is obtained by:

- Picking an initial state according to distribution ρ.
- Iterating:
  - Picking an action according to the policy.
  - Drawing a new state and a reward, according to the transition kernel of the environment.

A simple custom policy for our GarbageBot

- Charges the battery as soon as they are low: $\pi(a_1|x_0) = 1$
- Otherwise looks for garbage: $\pi(a_0|x_1) = 1$ and $\pi(a_0|x_2) = 1$

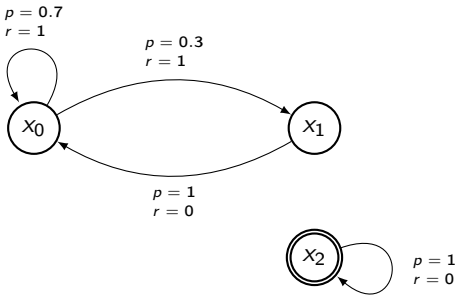


Figure: Markov Reward Process (Markov Chain) associated to policy π on the GarbageBot environment.

Quizz time: Is this an MDP? (4)

StarCraft II against a fixed opponent

- **Reward:** $+1$ for winning, -1 for losing.
- **Actions:** Keyboard and mouse controls.
- **States:** The state of the screen (+ audio).

Quizz time: Is this an MDP? (4)

StarCraft II against a fixed opponent

- **Reward:** $+1$ for winning, -1 for losing.
- **Actions:** Keyboard and mouse controls.
- **States:** The state of the screen (+ audio).

No

Quizz time: Is this an MDP? (5)

Chess against a fixed opponent

- **Reward:** $+1$ for winning, -1 for losing.
- **Actions:** Pieces standard moves.
- **States:** The state of the board.
- **Opponent policy:** $\pi_{\text{fixed}}(\text{move}|\text{board state})$.

Quizz time: Is this an MDP? (5)

Chess against a fixed opponent

- **Reward:** $+1$ for winning, -1 for losing.
- **Actions:** Pieces standard moves.
- **States:** The state of the board.
- **Opponent policy:** $\pi_{\text{fixed}}(\text{move}|\text{board state})$.

Yes

Quizz time: Is this an MDP? (6)

Chess against Magnus Carlsen

- **Reward:** $+1$ for winning, -1 for losing.
- **Actions:** Pieces standard moves.
- **States:** The state of the board.
- **Opponent policy:** What Magnus Carlsen would play.

Quizz time: Is this an MDP? (6)

Chess against Magnus Carlsen

- **Reward:** $+1$ for winning, -1 for losing.
- **Actions:** Pieces standard moves.
- **States:** The state of the board.
- **Opponent policy:** What Magnus Carlsen would play.

Probably not

Returns and goal of Reinforcement Learning

Cumulative Return

We define the cumulative return of a trajectory, $Z(\tau^\pi)$ as the discounted sum of rewards over the trajectory:

$$Z(\tau^\pi) = \sum_{t \geq 0} \gamma^t R_t.$$

Intuitions on Cumulative Return

- The cumulative return takes into account the rewards obtained at **all steps of the trajectory**.
- The discount factor $\gamma < 1$ quantifies a preference for the present:
  - A unit of reward at time 0 is worth $\frac{1}{\gamma}$ units of reward at time 1.
  - With a discount of γ, future rewards will typically become negligible after $O(\frac{1}{1-\gamma})$ timesteps.
  - Undiscounted returns can be defined, but are harder to work with.

Important Remark

Cumulative returns Z^π are stochastic (they are random variables), due to **randomness in both the policy and the environment**. We want to define (and optimize) averages of these quantities over trajectories.

Expected Cumulative Returns

$$
\begin{aligned}
J^\pi &= \mathbb{E}_{\tau^\pi \sim \mathcal{T}(\pi)}\Big[Z(\tau^\pi)\Big], \\
V^\pi(x) &= \mathbb{E}_{\tau^\pi \sim \mathcal{T}(x,\pi)}\Big[Z(\tau^\pi)\Big], \\
Q^\pi(x,a) &= \mathbb{E}_{\tau^\pi \sim \mathcal{T}(x,a,\pi)}\Big[Z(\tau^\pi)\Big], \\
V^\pi(x) &= \mathbb{E}_{a\sim\pi(\cdot|x)}\Big[Q^\pi(x,a)\Big] \text{ and } J^\pi = \mathbb{E}_{x\sim\rho}\Big[V^\pi(x)\Big].
\end{aligned}
$$

More Intuitions on the Expected Returns, Value and Q-functions

- J^π is the average cumulative returns over all trajectories. It provides a single number to attest for **how good a policy performs**.
- $V^\pi(x)$, the value (or V-)function, is the average cumulative returns over all trajectories that start in state x. It provides an indication of **how good one state is**, compared to another, **under the current policy**. It can also be used to **compare the performance of two policies**, starting in the same initial state x.
- $Q^\pi(x, a)$, the Q-function, is the average cumulative returns over all trajectories that start with action a being performed in state x. It provides an indication of **how good one action is in one state**, compared to another, **under the current policy**.

Formal Definition of The RL Goal

The goal of a reinforcement learning algorithm, in a given MDP, is to find an **optimal policy** $\pi^* \in \Pi$, such that for all $x \in \mathcal{X}$, and for any policy π, $V^{\pi^*}(x) \geq V^{\pi}(x)$.

Remark on Optimal Policy.

An optimal policy is thus simply **a policy that performs better than any other policy**, for all initial state x.

Question: How do we compute the optimal policy?

Dynamic programming: Reinforcement learning with p, r and ρ known

Bellman equation on V

The value function verifies the following **Bellman** equation:

$$V^{\pi}(x) = \sum_{a \in \mathcal{A}} \pi(a|x) r(x,a) + \gamma \sum_{y \in \mathcal{X}} \sum_{a \in \mathcal{A}} \pi(a|x) p(y|x,a) V^{\pi}(y)$$

What the Bellman equation means

The expected return can be decomposed into two parts:

- **An instantaneous part**, how much rewards am I going to receive at the next step, following the policy.
- **A future part**. This future part looks at all states that are reachable from x, computes the V function on each of those states (i.e. the expected return one will get from this state), and averages all those values.

Bellman equation on V: Sketch of Proof

$$
\begin{aligned}
V^\pi(x) &= \mathbb{E}[\sum_{t\geq 0}\gamma^t r(X_t, A_t)|X_0 = x] \\
&= \mathbb{E}[r(X_0, A_0) + \sum_{t\geq 1}\gamma^t r(X_t, A_t)|X_0 = x] \\
&= \mathbb{E}_{A_0\sim\pi(\cdot|x)}[r(x, A_0)] + \gamma\mathbb{E}[\sum_{t\geq 0}\gamma^t r(X_{t+1}, A_{t+1})|X_0 = x] \\
&= \mathbb{E}_{a\sim\pi(\cdot|x)}[r(x, a)] + \gamma\mathbb{E}[\mathbb{E}[\sum_{t\geq 0}\gamma^t r(X_{t+1}, A_{t+1})|X_1]|X_0 = x] \\
&= \mathbb{E}_{a\sim\pi(\cdot|x)}[r(x, a)] + \gamma\mathbb{E}[V^\pi(X_1)|X_0 = x] \\
&= \sum_a \pi(a|x)r(x, a) + \gamma\sum_{y\in\mathcal{X}}\sum_{a\in\mathcal{A}}\pi(a|x)p(y|x, a)V^\pi(y).
\end{aligned}
$$

Bellman equation on Q

The Q function similarly verifies an **equivalent Bellman equations**:

$$Q^\pi(x,a) = r(x,a) + \gamma \sum_{y \in \mathcal{X}} \sum_{b \in \mathcal{A}} p(y|x,a)\pi(b|y)Q^\pi(y,b).$$

Affine Property of the Bellman equations

One can remark that $V^\pi(x)$ and $Q^\pi(x,a)$ can be expressed as affine functions of $V^\pi(x)$ and $Q^\pi(x,a)$ respectively. Therefore it is possible to express them with matricial notations when the state and action spaces are finite.

Exercice on the GarbageBot

Using the custom policy:

- Charges the battery as soon as they are low: $\pi(a_1|x_0) = 1$
- Otherwise looks for garbage: $\pi(a_0|x_1) = 1$ and $\pi(a_0|x_2) = 1$

compute the following quantities by solving Bellman equations:

- Q^π
- V^π

What we know

- V^π verifies a Bellman equation.
- Q^π verifies a Bellman equation.

What we will show

- V^π is the unique function that verifies the Bellman equation.
- We will propose the optimality Bellman equation with unique solution V^*.
- The function V^* dominates all the V^π.
- There is a set of policies such that $V^\pi = V^*$.

What do we need

- Evaluation Operator.
- Optimality Operator.

Evaluation Operator

The evaluation operator $B^\pi \in \mathbb{R}^{\mathcal{X}^{\mathbb{R}^{\mathcal{X}}}}$ is defined as follows:

$$B^\pi[V](x) = \sum_{a \in \mathcal{A}} \pi(a|x) \left[r(x,a) + \gamma \sum_{y \in \mathcal{X}} p(y|x,a) V(y) \right].$$

Optimality Operator

The optimality operator $B^\star \in \mathbb{R}^{\mathcal{X}^{\mathbb{R}^{\mathcal{X}}}}$ is defined as follows:

$$B^\star[V](x) = \max_{a \in \mathcal{A}} \left[r(x,a) + \gamma \sum_{y \in \mathcal{X}} p(y|x,a) V(y) \right].$$

Evaluation Operator

The evaluation operator $T^\pi \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}{}^{\mathbb{R}^{\mathcal{X}\times\mathcal{A}}}$ is defined as follows:

$$T^\pi[Q](x,a) = r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a) \sum_{b\in\mathcal{A}} \pi(b|y) Q(y,b).$$

Optimality Operator

The optimality operator $T^\star \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}{}^{\mathbb{R}^{\mathcal{X}\times\mathcal{A}}}$ is defined as follows:

$$T^\star[Q](x,a) = r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a) \max_{b\in\mathcal{A}} Q(y,b).$$

Monotonicity

Let $V_1 \in \mathbb{R}^X$ and $V_2 \in \mathbb{R}^X$ such that $V_1 \leq V_2$ and let $\pi \in \Pi$, then

$$B^\pi[V_1] \leq B^\pi[V_2],$$
$$B^\star[V_1] \leq B^\star[V_2].$$

The same goes with evaluation and optimality operators for Q-functions.

Domination of the Optimal Operator

Let $V \in \mathbb{R}^X$ and $\pi \in \Pi$, then:

$$B^\pi V \leq B^\star V.$$

The same goes with evaluation and optimality operators for Q-functions.

Exercise

Prove those properties.

Th-1: Fixed-points of the evaluation operators.

B^π is a γ-contraction with unique fixed-point V^π. Similarly, T^π is a γ-contraction with unique fixed-point Q^π:

$$B^\pi[V^\pi] = V^\pi,$$
$$T^\pi[Q^\pi] = Q^\pi.$$

Proof that T^π is a contraction.

Let $Q_1 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ and $Q_2 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$, then $\forall (x,a) \in \mathcal{X}\times\mathcal{A}$ we have:

$$\begin{aligned} T^\pi[Q_1](x,a) - T^\pi[Q_2](x,a) &= \gamma \sum_{y\in\mathcal{X}} \sum_{b\in\mathcal{A}} p(y|x,a)\pi(b|y)[Q_1(y,b) - Q_2(y,b)], \\ &\leq \gamma \max_{b\in\mathcal{A}, y\in\mathcal{X}} [Q_1(y,b) - Q_2(y,b)], \\ &\leq \gamma \|Q_1 - Q_2\|_\infty. \end{aligned}$$

Therefore $\|T^\pi[Q_1] - T^\pi[Q_2]\|_\infty \leq \gamma \|Q_1 - Q_2\|_\infty$

Th-2: Fixed-points of the optimality operators.

$B^\star$ is a γ-contraction, its unique fixed-point is noted $V^\star$ and called the optimal value function. Similarly, $T^\star$ is a γ-contraction, its unique fixed-point is noted $Q^\star$ and called the optimal action-value function:

$$B^\star[V^\star] = V^\star,$$
$$T^\star[Q^\star] = Q^\star.$$

Proof

Prove the theorem as an exercise. This consists in showing that $T^\star$ and $B^\star$ are contractions.

Th-3:Majorants of the Value functions.

$V^\star$ is a majorant of $\{V^\pi\}_{\pi\in\Pi}$ and $Q^\star$ is a majorant of $\{Q^\pi\}_{\pi\in\Pi}$:

$$\forall \pi \in \Pi, V^\pi \leq V^\star,$$
$$\forall \pi \in \Pi, Q^\pi \leq Q^\star.$$

Proof for Q functions.

By domination of the optimality operator, we have that:

$$Q^\pi = T^\pi Q^\pi \leq T^\star Q^\pi.$$

By monotonicity of the optimality operator, we have that:

$$T^\star Q^\pi \leq (T^\star)^2 Q^\pi \leq (T^\star)^3 Q^\pi \cdots \leq (T^\star)^n Q^\pi.$$

Therefore at the limit $T^\star Q^\pi \leq (T^\star)^\infty Q^\pi = Q^\star$ which concludes the proof.

Th-4: Existence and construction of optimal policies.

A policy $\pi \in \Pi$ is optimal if and only if $V^\pi = V^\star$. Optimal stochastic policies exist and can be constructed from $Q^\star$ or Q^π:

$$\begin{aligned} V^\pi = V^\star &\iff \forall x \in \mathcal{X}, \text{Supp}[\pi(.|x)] \subset \arg\max_{a \in \mathcal{A}}[Q^\star(x,a)], \\ &\iff \forall x \in \mathcal{X}, \text{Supp}[\pi(.|x)] \subset \arg\max_{a \in \mathcal{A}}[Q^\pi(x,a)], \end{aligned}$$

$\text{Supp}[\pi(.|x)] = \{a \in \mathcal{A} | \pi(a|x) > 0\}$ is the support of $\pi(.|x)$.

Proof: Exercise

To do so remark that for a function $Q \in \mathbb{R}^{\mathcal{X} \times \mathcal{A}}$, for a policy π and for $x \in \mathcal{X}$:

$$\text{Supp}[\pi(.|x)] \subset \arg\max_{a \in \mathcal{A}}[Q(x,a)] \iff \sum_{a \in \mathcal{A}} \pi(a|x) Q(x,a) = \max_{a \in \mathcal{A}} Q(x,a).$$

This means that expectation and maximum operators are equivalent.

Remark on Greediness

For a given function $Q \in \mathbb{R}^{\mathcal{X} \times \mathcal{A}}$, the set of policies:

$$\mathcal{G}(Q) = \{\pi \in \Pi | \forall x \in \mathcal{X}, \text{Supp}[\pi(.|x)] \subset \arg\max_{a \in \mathcal{A}}[Q(x,a)]\},$$

is called the greedy set with respect to Q. If $\pi \in \mathcal{G}(Q)$, we say that π is a greedy policy with respect to Q. Therefore, a policy π is optimal if and only if it is greedy with respect to $Q^\star$ or Q^π:

$$\begin{aligned} V^\pi = V^\star &\iff \pi \in \mathcal{G}(Q^\pi), \\ &\iff \pi \in \mathcal{G}(Q^\star). \end{aligned}$$

Th-5: The Greedy Policy Improvement.

Let $\pi \in \Pi$ be a stochastic policy, then the greedy policy with respect to Q^π noted $\pi_\mathcal{G} \in \mathcal{G}(Q^\pi)$ is such that:

$$\forall \pi \in \Pi, \quad Q^\pi \leq Q^{\pi_\mathcal{G}}.$$

In addition:

$$Q^\pi = Q^{\pi_\mathcal{G}} \iff Q^\pi = Q^\star \implies \pi_\mathcal{G} \text{ is optimal.}$$

Lessons of the 5 Fundamental Theorems:

- They show how we can explicitly construct an optimal policy from $Q^\star$.
- $Q^\star$ is the fixed point of a γ-contraction operator.

Value Iteration

$$\textbf{Value Iteration:} \begin{cases} & Q_0 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}} & \textbf{Initialisation} \\ \forall k \in \mathbb{N}, & Q_{k+1} = T^\star Q_k & \textbf{Recurrence} \end{cases}$$

As $T^\star$ is a γ-contraction operator, we have $\lim_{k\to\infty} Q_k = Q^\star$.

Remark On Greediness

With the precedent remark on greediness, it is trivial to show that :

$$\pi \in \mathcal{G}(Q) \implies T^\star Q = T^\pi Q.$$

Therefore, one can rewrite Value Iteration:

$$\textbf{Value Iteration:} \begin{cases} & Q_0 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}} & \textbf{Initialisation} \\ \forall k \in \mathbb{N}, & \pi_k \in \mathcal{G}(Q_k) & \textbf{Greedy-step} \\ \forall k \in \mathbb{N}, & Q_{k+1} = T^{\pi_k} Q_k & \textbf{Evaluation} \end{cases}$$

Value Iteration Performance Bound

$$\|Q^\star - Q^{\pi_k}\|_\infty \leq \frac{2\gamma^k \|r\|_\infty}{(1-\gamma)^2},$$

where $\|.\|_\infty$ is the infinite norm.

Policy Iteration

It relies mainly on the greedy policy improvement theorem.

$$\textbf{Policy Iteration:} \begin{cases} & Q_0 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}} & \textbf{Initialisation} \\ \forall n \in \mathbb{N}, & \pi_k \in \mathcal{G}(Q_k) & \textbf{Greedy-step} \\ \forall n \in \mathbb{N}, & Q_{k+1} = (T^{\pi_k})^{\infty} Q_k & \textbf{Evaluation} \end{cases}$$

Full Evaluation

In PI, the evaluation is a full evaluation $Q_{k+1} = (T^{\pi_k})^{\infty} Q_k$ which means that $Q_{k+1} = Q^{\pi_k}$.

Improvement Property

Because of the greedy policy improvement theorem, we have:

$$Q_1 = Q^{\pi_0} \leq Q_2 = Q^{\pi_1} \leq Q_3 = Q^{\pi_2} \dots$$

where each step is strictly improving unless optimality is reached.

Convergence Property

The PI scheme finishes in a finite number of steps:

$$\exists K \in \mathbb{N}, \text{ such that } \pi_K \text{ is optimal.}$$

Modified Policy Iteration

It generalizes VI and PI in one common framework.

$$\textbf{MPI:} \begin{cases} & Q_0 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}} & \textbf{Initialisation} \\ \forall k \in \mathbb{N}, & \pi_k \in \mathcal{G}(Q_k) & \textbf{Greedy-step} \\ \forall n \in \mathbb{N}, & Q_{k+1} = (T^{\pi_k})^m Q_k & \textbf{Evaluation} \end{cases}$$

Links between PI, VI and MPI

Here the evaluation is $Q_{k+1} = (T^{\pi_k})^m Q_k$ with $m \in \overline{\mathbb{N}^*}$. This trivially generalizes to VI with $m = 1$ and to PI with $m = \infty$.

Three Major Difficulties

- Locality problem: States can't be trivially reached. Therefore information about the dynamics and the reward can't be trivially collected. This calls for the need of some underlying process, called exploration, to get this necessary information to solve the task.
- Sampling problem: Dynamics are not fully known. This puts an even bigger burden on the exploration process.
- Representation problem: State and action spaces can be so large that associated information can't be stored (nor updated) but need to be learned through functional approximations. Objects of interest such as the Q-values will be parameterised by $\theta \in \mathbb{R}^N$ where $N \in \mathbb{N}$ is the number of parameters. When functional approximation is done with deep neural networks, we are in the territory of Deep Reinforcement Learning.

What Information we typically have access to

In most practical applications, an agent observes at time t the state x_t choose action a_t, receives the reward $r_t = r(x_t, a_t)$ and transition to state $x_{t+1} = y_t$ with probability $p(y_t|x_t, a_t)$. The agent has no access to the local probability $p(y_t|x_t, a_t)$ nor the global information of dynamics p, reward r, but only to the transition (x_t, a_t, r_t, y_t).

Reinforcement learning without knowing p, r and ρ

Approximating iterations of the Bellman operator

Idea: We cannot perform an exact application of B^π, as we don't know p, but we can still stochastically approximate it.

Approximate policy evaluation

$$
\begin{array}{lll}
 & V_0 \in \mathbb{R}^{\mathcal{X}}, s_0 \sim \rho_0 & \textbf{Initialisation} \\
t \in \mathbb{N}, & a_t \sim \pi(\cdot|s_t) & \textbf{Action selection} \\
 & s_{t+1} \sim p(\cdot|s_t, a_t), r_t = r(s_t, a_t) & \textbf{Environment step} \\
 & V_{t+1}(s_t) = \alpha_t(r_t + \gamma V_t(s_{t+1})) + (1 - \alpha_t)V_t(s_t). & \textbf{Evaluation}
\end{array}
$$

Remarks

- The α_t scaling acts as a learning rate: when transitions are not deterministic, the evaluation step is itself stochastic, and can permanently oscillate if α_t is not decreased adequately.
- Under good assumptions, i.e. sufficient coverage of the state space by the policy and proper decrease of α_t, V_t converges to V^π.

Approximating iterations of the Bellman optimality operator

Idea: We can do exactly what we did for policy evaluation for policy improvement.

Naive Q-learning

$$
\begin{array}{lll}
 & Q_0 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}, s_0 \sim \rho_0 & \textbf{Initialisation} \\
t \in \mathbb{N}, & a_t = \operatorname{argmax}_a Q_t(s_t, a) & \textbf{Action selection} \\
 & s_{t+1} \sim p(\cdot|s_t, a_t), r_t = r(s_t, a_t) & \textbf{Environment step} \\
 & Q_{t+1}(s_t, a_t) = \alpha_t(r_t + \gamma \operatorname{argmax}_a Q_t(s_{t+1}, a)) + (1-\alpha_t)Q_t(s_t, a_t). & \textbf{Evaluation}
\end{array}
$$

Problem

We are playing a greedy policy, that is very unlikely to cover the whole state space. This breaks the convergence property that we had in the policy evaluation case, under the *good coverage* hypothesis.

ε-greedy Q-learning

	$Q_0 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}, s_0 \sim \rho_0$	**Initialisation**
$t \in \mathbb{N},$	$a_t = \operatorname{argmax}_a Q_t(s_t, a)$ w.p. 1-ε, $a_t \sim U(\lvert\mathcal{A}\rvert)$ w.p. ε	**Action selection**
	$s_{t+1} \sim p(\cdot\lvert s_t, a_t), r_t = r(s_t, a_t)$	**Environment step**
	$Q_{t+1}(s_t, a_t) = \alpha_t(r_t + \gamma \operatorname{argmax}_a Q_t(s_{t+1}, a)) + (1-\alpha_t)Q_t(s_t, a_t).$	**Evaluation**

Remarks

- We are introducing a small probability ε for the agent to play an action that it deems suboptimal, so as to preserve the coverage property that we need for convergence.
- This strategy is a way to tackle the *Exploration - Exploitation* trade-off (albeit a very naive one).
- The policy we are learning, greedy w.r.t. Q, is different from the policy we are using for learning, ε-greedy w.r.t. Q: Q-learning is an **off-policy** algorithm.
- In the practical we are going to see an **on-policy** algorithm that learns a close to optimal policy.

What we did today

- Introduced MDPs, a standard setup for RL.
- Introduced the **goal of reinforcement learning**: to find a policy that **maximizes the discounted cumulated reward** in an MDP.
- Introduced algorithms to achieve this goal, **when we know everything about the MDP**, including the transition kernel.
- Made a excursion into the realm of model-free tabular reinforcement learning, where we don't have access to p, r or ρ.

Readings and courses

- Sutton and Barto.
- David Silver's RL course.