Deep RL course

Deep RL team

Introduction

This document is not a book nor a survey. It is not a reference nor exhaustive. It is simply a mere presentation of some existing deep Reinforcement Learning (RL) algorithms and the relative underlying concepts. Reinforcement Learning is a general framework for Control Theory where an agent, interacting with the world, try to act optimally with respect to a quantity called the reward function. In Chapter 1, concepts and notations relative to RL are introduced alongside Dynamic Programming schemes that solve the RL problem when the world is fully-known and discrete, in space and time, and its dynamics is fully-known, Markovian and can be represented by a discrete transition kernel (finite-size matrix). The rest of the chapters tackle the general RL problem where only local pieces of information about the world are collected via direct interaction. More specifically, the other chapters focus on deep RL where quantity of interests are stored in neural networks and optimized via stochastic gradient descent (or similar optimization schemes).

Chapter 1

Reminders of Dynamic Programming

1.1 Introduction

This chapter should be seen as a *cookbook.* It gives reminders of the important concepts and theorems that should have been introduced in a Reinforcement Learning course. We refer more advanced readers to the following contributions [Hernández-Lerma and Lasserre, 2012, Puterman, 2014, Geist et al., 2019] that have been our sources of inspiration. After introducing *Markov Decision Processes* (MDP), the usual framework for which reinforcement learning algorithms are designed, we prove the Bellman equations which is a key result towards finding an optimal behaviour. Then, we introduce the necessary background and notations to define *Dynamic Programming* (DP) schemes in MDPs, such as value iteration and policy iteration. Those schemes are at the heart of most Reinforcement Learning algorithms. In addition, we give some elements that highlight the main differences between DP in general, and DP in Reinforcement Learning.

All statements (theorems, corollaries and properties) are proven except statements relative to Markov chains and general stochastic processes. This choice has been made for two reasons: (i) to keep the section compact and readable and (ii) because proofs on Markov chains properties can be easily found in any good book or course on discrete stochastic processes.

1.2 Background and Notations

Reinforcement Learning algorithms are usually framed as algorithms that try to find an optimal solution to a reward maximization problem in a *Markov Decision Process* (MDP). MDPs formalise the concept of *fully observable environments.* Intuitively, a Markov Decision Process is an environment within which an agent can interact through actions, and that provides observations, that are *fully representative of the state of the environment at a given time.* These observations are often referred to as *states.* What this formally implies is that, given a current state x, and an action a to be executed, the new state y that is observed only depends on x and a, and is independent from all past states and actions. In addition to states, MDPs also provide agents with *rewards* which are used to quantify how good an action is in a given state. To simplify things, we will consider, in what follows, only MDPs with finite action and state sets. 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: models the Markov dynamics of the MDP. Formally, it is a transition kernel from $\mathcal{X} \times \mathcal{A}$ to $\mathcal{X}$ which means that $p(y|x, a)$ denotes the probability of transitioning to state $y \in \mathcal{X}$ by choosing action $a \in \mathcal{A}$ in state $x \in \mathcal{X}$.
- ρ: the initial state distribution on $\mathcal{X}$ and $\rho(x)$ is the probability to start in state $x \in \mathcal{X}$.
- 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. The discount factor quantifies a notion of preference for the present; with a discount of γ, obtaining a reward of 1 at time t is as beneficial as obtaining a reward of $1/\gamma$ at time $t + 1$.

An agent interacts with an MDP by choosing an action on a given state. This choice is modelled by a policy. We consider *markovian* stochastic policies π, i.e. state dependent probability distributions over actions. This means that, for a policy π, $\pi(a|x)$ 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. Note that considering only Markovian policies is restrictive: we could consider stochastic policies whose choice of actions depends of several states and actions in the past. In the following, we are going to define the concept of value function of policy π that is the expected and discounted cumulative reward collected by the policy π and show that it verifies the

Bellman equations. We present two different approaches to get this result. An algebraic approach that is easy to understand and requires only some notions of linear algebra and a Markov chain approach that necessitates some notions of measure theory but can be easily generalise to continuous state-action spaces. We let the reader choose one or the other.

1.3 Algebraic approach

In this section, we are going to define value functions and prove the Bellman equations with simple notions of linear algebra. To do so, we are going to first compute what is the expected reward of acting with the policy π and then what is the expected and discounted cumulative return after $k \in \mathbb{N}$ steps. At the limit of $k \to \infty$, we are going to see that this notion of discounted and expected cumulative return still makes sense and verifies the Bellman equations.

First, let define a finite probability distribution $\eta \in \mathbb{R}^{\mathcal{S}}$ as a positive real function that sums to one over a finite set $\mathcal{S}$:

$$\forall s \in \mathcal{S}, \eta(s) \geq 0,$$
$$\sum_{s\in\mathcal{S}} \eta(s) = 1.$$

We note $\Delta_{\mathcal{S}}$ the set of discrete probability distributions over $\mathcal{S}$. Let $f \in \mathbb{R}^{\mathcal{S}}$ a real function, we define the expectation of f with respect to the $\eta \in \Delta_{\mathcal{S}}$ as:

$$\mathbb{E}_\eta[f] = \sum_{s\in\mathcal{S}} f(s)\eta(s).$$

We can remark that if we see f and η are seen as column vectors of dimensions $|\mathcal{S}| \times 1$ then we have:

$$\mathbb{E}_\eta[f] = \eta^\top f,$$

where $\top$ is the transposition operator.

Expected reward associated to a policy $\pi \in \Pi$ and probability distribution $\eta \in \Delta_{\mathcal{X}}$. Let define the function $r^\pi \in \mathbb{R}^{\mathcal{X}}$ as the expected reward in state x following policy π:

$$\forall x \in \mathcal{X}, r^\pi(x) = \sum_{a\in\mathcal{A}} \pi(a|x) r(x,a),$$

then the expected reward associated to a policy π and distribution η is simply:

$$r^\pi_\eta = \mathbb{E}_\eta[r^\pi] = \sum_{x\in\mathcal{X}} r^\pi(x)\eta(x) = \eta^\top r^\pi.$$

The quantity r^π_η represents the expected immediate reward an agent collects starting from a distribution of states η and then acting with policy π.

Next state distribution associated to a policy $\pi \in \Pi$ and probability distribution $\eta \in \Delta_{\mathcal{X}}$. One natural question is what will be the probability $p^\pi_{\mathcal{X}}(y|x)$ to reach a given state y after applying policy π starting from state x. The answer is :

$$\forall \pi \in \Pi, \forall (x,y) \in \mathcal{X}^2, p^\pi_{\mathcal{X}}(y|x) = \sum_{a\in\mathcal{A}} p(y|x,a)\pi(a|x).$$

It is trivial to check that $p^\pi_{\mathcal{X}}(.|x) \in \mathbb{R}^X$ is indeed a probability distribution:

$$\sum_{y\in\mathcal{X}} p^\pi_{\mathcal{X}}(y|x) = \sum_{y\in\mathcal{X}}\sum_{a\in\mathcal{A}} p(y|x,a)\pi(a|x) = \sum_{a\in\mathcal{A}} \pi(a|x) \sum_{y\in\mathcal{X}} p(y|x,a) = \sum_{a\in\mathcal{A}} \pi(a|x) = 1.$$

Now, we define $\eta'(y)$ the probability to reach a given state y after applying policy π starting from state distribution η:

$$\forall \pi \in \Pi, \forall \eta \in \Delta_{\mathcal{X}}, \forall y \in \mathcal{X}, \eta'(y) = \sum_{x\in\mathcal{X}} \eta(x) p^\pi_{\mathcal{X}}(y|x).$$

It is trivial to check that η' is indeed a probability distribution:

$$\sum_{y\in\mathcal{X}} \eta'(y) = \sum_{y\in\mathcal{X}}\sum_{x\in\mathcal{X}} \eta(x) p^\pi_{\mathcal{X}}(y|x) = \sum_{x\in\mathcal{X}} \eta(x) \sum_{y\in\mathcal{X}} p^\pi_{\mathcal{X}}(y|x) = \sum_{x\in\mathcal{X}} \eta(x) = 1.$$

In addition, if we define $P^\pi_{\mathcal{X}}$ the stochastic square matrix of size $|\mathcal{X}|$ such that:

$$\forall (x,y) \in \mathcal{X}^2, \quad P^\pi_{\mathcal{X}}(x,y) = p^\pi_{\mathcal{X}}(y|x) = \sum_{a\in\mathcal{A}} p(y|x,a)\pi(a|x),$$

then, by definition of η' and $P^\pi_{\mathcal{X}}$, we have:

$$\eta'^\top = \eta^\top P^\pi_{\mathcal{X}}$$

State probability distributions associated to a policy $\pi \in \Pi$ starting from $\rho \in \Delta_{\mathcal{X}}$. One natural question is what will be the state probability distribution of states ρ_t starting from the initial ρ and applying the policy for $t \in \mathbb{N}$ steps. From the previous paragraph we already know that:

$$\forall t \in \mathbb{N}, \rho_{t+1}^\top = \rho_t^\top P_{\mathcal{X}}^\pi,$$

where $\rho_0 = \rho$. Therefore by recurrence, we have:

$$\forall t \in \mathbb{N}, \rho_t^\top = \rho^\top (P_{\mathcal{X}}^\pi)^t.$$

Expected and discounted cumulative return of a policy π starting from $\rho \in \Delta_{\mathcal{X}}$. We note $V_\mu^{\pi,k}$ the discounted and expected cumulative rewards that has been collected by the agent after $k \in \mathbb{N}$ steps starting from μ and then following policy π. As seen in the previous paragraph, we have a closed-form formula for the state probability distribution μ_t, therefore the expected reward collected at step t is simply $r_{\mu_t}^\pi$ and we can directly define $V_\mu^{\pi,k}$ as the discounted sum of rewards $r_{\mu_t}^\pi$ for k steps:

$$V_\mu^{\pi,k} = \sum_{t=0}^{k} \gamma^t r_{\mu_t}^\pi = \sum_{t=0}^{k} \gamma^t \mu_t^\top r^\pi = \mu^\top \sum_{k=0}^{t} (\gamma P_{\mathcal{X}}^\pi)^t r^\pi.$$

We define the function $V^{\pi,k} \in \mathbb{R}^{\mathcal{X}}$ such that:

$$V^{\pi,k} = \sum_{t=0}^{k} (\gamma P_{\mathcal{X}}^\pi)^t r^\pi,$$

therefore:

$$V_\mu^{\pi,k} = \mu^\top V^{\pi,k}.$$

One can remark that for $\mu = \delta_x$, where δ_x is the Dirac distribution over $\mathcal{X}$ on state x, we have:

$$V_{\delta_x}^{\pi,k} = \delta_x^\top V^{\pi,k} = V^{\pi,k}(x).$$

Therefore $V^{\pi,k}(x)$ is the expected and discounted cumulative return starting from state x and following policy π for k steps. In RL, the value function $V^\pi(x)$ is defined as the limit of $V^{\pi,k}(x)$, when $k \to \infty$ and represents the expected and discounted cumulative return starting from state x and following policy π infinitely.

Theorem 1. *Definition and Existence of V^π.*
For $\pi \in \Pi$, let define the function $V^\pi \in \mathbb{R}^{\mathcal{X}}$ as:

$$V^\pi = \lim_{k \to \infty} V^{\pi,k}.$$

This quantity exists and equals to:

$$V^\pi = (I_{\mathcal{X}} - \gamma P_{\mathcal{X}}^\pi)^{-1} r^\pi,$$

where $I_{\mathcal{X}}$ is the identity matrix of size $|\mathcal{X}|$.

Proof. First, one can remark that for $k \in \mathbb{N}$:

$$(I_{\mathcal{X}} - \gamma P_{\mathcal{X}}^\pi) \sum_{t=0}^{k} (\gamma P_{\mathcal{X}}^\pi)^t = I_{\mathcal{X}} - (\gamma P_{\mathcal{X}}^\pi)^{k+1}.$$

Second as $P_{\mathcal{X}}^\pi$ is a square stochastic matrix, its eigenvalues are smaller in absolute value than 1. Therefore, as $\gamma \in [0, 1)$, the matrix $\gamma P_{\mathcal{X}}^\pi$ has its eigenvalues smaller than 1 in absolute value. Therefore the matrix $I_{\mathcal{X}} - \gamma P_{\mathcal{X}}^\pi$ have non-zeros eigenvalues and is invertible. This implies that for $k \in \mathbb{N}$:

$$\sum_{t=0}^{k} (\gamma P_{\mathcal{X}}^\pi)^t = (I_{\mathcal{X}} - \gamma P_{\mathcal{X}}^\pi)^{-1} (I_{\mathcal{X}} - (\gamma P_{\mathcal{X}}^\pi)^{k+1}).$$

Third, because the product of two stochastic matrices is still a stochastic matrix, we know that all elements of the matrix $(P_{\mathcal{X}}^\pi)^k$ are smaller in absolute value than one. Thus all elements of the matrix $(\gamma P_{\mathcal{X}}^\pi)^k$ are smaller in absolute value than γ^k. Therefore, as $\gamma \in [0, 1)$, the series $\left((\gamma P_{\mathcal{X}}^\pi)^k\right)_{k \in \mathbb{N}}$ converges to $0_{\mathcal{X}}$ the null square matrix. This implies that:

$$\lim_{k \to \infty} \sum_{t=0}^{k} (\gamma P_{\mathcal{X}}^\pi)^t = (I_{\mathcal{X}} - \gamma P_{\mathcal{X}}^\pi)^{-1} \lim_{k \to \infty} (I_{\mathcal{X}} - (\gamma P_{\mathcal{X}}^\pi)^{k+1}) = (I_{\mathcal{X}} - \gamma P_{\mathcal{X}}^\pi)^{-1} (I_{\mathcal{X}} - 0_{\mathcal{X}}) = (I_{\mathcal{X}} - \gamma P_{\mathcal{X}}^\pi)^{-1}.$$

Therefore, this allows us to conclude:

$$\lim_{k \to \infty} V^{\pi,k} = \lim_{k \to \infty} \sum_{t=0}^{k} (\gamma P_{\mathcal{X}}^\pi)^t r^\pi = (I_{\mathcal{X}} - \gamma P_{\mathcal{X}}^\pi)^{-1} r^\pi.$$

□

The Bellman equations are at the heart of RL. They express the fact that it is possible to compute the cumulative return by using the cumulative return at the next state. This is a simple bootstrapping trick.

Theorem 2. *For all $\pi \in \Pi$, V^π verifies the following Bellman equation:*

$$\forall \pi \in \Pi, \quad V^\pi = r^\pi + \gamma P^\pi_{\mathcal{X}} V^\pi.$$

Proof.

$$V^\pi = (I_{\mathcal{X}} - \gamma P^\pi_{\mathcal{X}})^{-1} r^\pi \iff (I_{\mathcal{X}} - \gamma P^\pi_{\mathcal{X}}) V^\pi = r^\pi \iff V^\pi = r^\pi + \gamma P^\pi_{\mathcal{X}} V^\pi.$$

□

In addition to the concept of value function, the concept of state-action value function is central in RL. This function corresponds to the expected and discounted cumulative return following a policy π and starting from a state-action couple (x, a) or a distribution of states and actions $\mu \in \Delta_{\mathcal{X}\times\mathcal{A}}$. This is an important concept because the action a can be seen as a degree of freedom and its choice allows us to improve the current policy π. We follow the same method to define the state-action value function as the one we just used to define the value function.

Next state distribution associated to a policy $\pi \in \Pi$ and probability distribution $\mu \in \Delta_{\mathcal{X}\times\mathcal{A}}$. We define the probability $p^\pi_{\mathcal{X}\times\mathcal{A}}((y,b)|(x,a))$ to reach a given state-action couple (y, b) after applying policy π starting from state (x, a):

$$\forall \pi \in \Pi, \forall ((x,a),(y,b)) \in (\mathcal{X}\times\mathcal{A})^2, \quad p^\pi_{\mathcal{X}\times\mathcal{A}}((y,b)|(x,a)) = p(y|x,a)\pi(b|y).$$

It is trivial to check that the function $p^\pi_{\mathcal{X}\times\mathcal{A}}(.|(x,a)) \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ is indeed a probability distribution.

Now, we define $\mu'(y,b)$ the probability to reach a given state-action couple (y,b) after applying policy π starting from state-action distribution μ:

$$\forall \pi \in \Pi, \forall \mu \in \Delta_{\mathcal{X}\times\mathcal{A}}, \forall (y,b) \in \mathcal{X}\times\mathcal{A}, \mu'(y,b) = \sum_{(x,a)\in\mathcal{X}\times\mathcal{A}} \mu(x,a) p^\pi_{\mathcal{X}\times\mathcal{A}}((y,b)|(x,a)).$$

It is trivial to check that the function $\mu' \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ is indeed a probability distribution.

In addition, if we define $P^\pi_{\mathcal{X}\times\mathcal{A}}$ the stochastic square matrix of size $|\mathcal{X}\times\mathcal{A}|$ such that:

$$\forall ((x,a),(y,b)) \in (\mathcal{X}\times\mathcal{A})^2, \quad P^\pi_{\mathcal{X}\times\mathcal{A}}((x,a),(y,b)) = p^\pi_{\mathcal{X}\times\mathcal{A}}((y,b)|(x,a)) = p(y|x,a)\pi(b|y),$$

then, by definition of μ' and $P^\pi_{\mathcal{X}\times\mathcal{A}}$, we have:

$$\mu'^\top = \mu^\top P^\pi_{\mathcal{X}\times\mathcal{A}}$$

State-action probability distributions associated to a policy $\pi \in \Pi$ starting from $\mu \in \Delta_{\mathcal{X}\times\mathcal{A}}$. The state probability distribution of states μ_t starting from the initial μ and then following π for $t \in \mathbb{N}$ steps verifies:

$$\forall t \in \mathbb{N}, \mu_{t+1}^\top = \mu_t^\top P^\pi_{\mathcal{X}\times\mathcal{A}},$$

where $\mu_0 = \mu$. Therefore by recurrence, we have:

$$\forall t \in \mathbb{N}, \mu_t^\top = \mu^\top (P^\pi_{\mathcal{X}\times\mathcal{A}})^t.$$

Expected and discounted cumulative return of a policy π starting from $\mu \in \Delta_{\mathcal{X}\times\mathcal{A}}$. We note $Q^{\pi,k}_\mu$ the discounted and expected cumulative rewards that has been collected by the agent after $k \in \mathbb{N}$ steps starting from μ and then following policy π. As seen in the previous paragraph, we have a closed-form formula for the state probability distribution μ_t, therefore the expected reward collected at step t is simply $r_{\mu_t} = \mu_t^\top r$ and we can directly define $Q^{\pi,k}_\mu$ as the discounted sum of rewards r_{μ_t} for k steps:

$$Q^{\pi,k}_\mu = \sum_{t=0}^{k} \gamma^t r_{\mu_t} = \sum_{t=0}^{k} \gamma^t \mu_t^\top r = \mu^\top \sum_{k=0}^{t} (\gamma P^\pi_{\mathcal{X}\times\mathcal{A}})^t r.$$

We define the function $Q^{\pi,k} \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ such that:

$$Q^{\pi,k} = \sum_{t=0}^{k} (\gamma P^\pi_{\mathcal{X}\times\mathcal{A}})^t r,$$

therefore:

$$Q^{\pi,k}_\mu = \mu^\top Q^{\pi,k}.$$

One can remark that for $\mu = \delta_{(x,a)}$, where $\delta_{(x,a)}$ is the Dirac distribution over $\mathcal{X} \times \mathcal{A}$ on state-action (x, a), we have:

$$Q^{\pi,k}_{\delta_{(x,a)}} = \delta^\top_{(x,a)} Q^{\pi,k} = Q^{\pi,k}(x, a).$$

Therefore $Q^{\pi,k}(x, a)$ is the expected and discounted cumulative return starting from state-action (x, a) and following policy π for k steps. In RL, the state-action value function $Q^\pi(x, a)$ is defined as the limit of $Q^{\pi,k}(x, a)$, when $k \to \infty$ and represents the expected and discounted cumulative return starting from state-action (x, a) and following policy π infinitely.

Theorem 3. *Definition and Existence of Q^π.*
For $\pi \in \Pi$, let define the function $Q^\pi \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ as:

$$Q^\pi = \lim_{k\to\infty} Q^{\pi,k}.$$

This quantity exists and equals to:

$$Q^\pi = (I_{\mathcal{X}} - \gamma P^\pi_{\mathcal{X}\times\mathcal{A}})^{-1} r,$$

where $I_{\mathcal{X}\times\mathcal{A}}$ is the identity matrix of size $|\mathcal{X} \times \mathcal{A}|$.

Proof. As shown in the proof of Theorem1, we have:

$$\lim_{k\to\infty} \sum_{t=0}^{k} (\gamma P^\pi_{\mathcal{X}\times\mathcal{A}})^t = (I_{\mathcal{X}\times\mathcal{A}} - \gamma P^\pi_{\mathcal{X}\times\mathcal{A}})^{-1}.$$

Therefore, this allows us to conclude:

$$\lim_{k\to\infty} Q^{\pi,k} = \lim_{k\to\infty} \sum_{t=0}^{k} (\gamma P^\pi_{\mathcal{X}\times\mathcal{A}})^t r = (I_{\mathcal{X}\times\mathcal{A}} - \gamma P^\pi_{\mathcal{X}\times\mathcal{A}})^{-1} r.$$

□

The Bellman equations are at the heart of RL. They express the fact that it is possible to compute the cumulative return by using the cumulative return at the next state. This is a simple bootstrapping trick.

Theorem 4. *For all $\pi \in \Pi$, Q^π verifies the following Bellman equation:*

$$\forall \pi \in \Pi, \quad Q^\pi = r + \gamma P^\pi_{\mathcal{X}\times\mathcal{A}} Q^\pi.$$

Proof.

$$Q^\pi = (I_{\mathcal{X}\times\mathcal{A}} - \gamma P^\pi_{\mathcal{X}\times\mathcal{A}})^{-1} r \iff (I_{\mathcal{X}\times\mathcal{A}} - \gamma P^\pi_{\mathcal{X}\times\mathcal{A}}) Q^\pi = r \iff Q^\pi = r + \gamma P^\pi_{\mathcal{X}\times\mathcal{A}} Q^\pi.$$

□

1.4 Markov chain approach

Remark 1. *Link with Markov Chains.*
A finite MDP is a generalization of the concept of finite Markov Chain. More precisely it is a ***valued*** *and* ***controllable*** *Markov chain. Indeed, it is characterised as* ***valued*** *because depending on the chosen state-action couple a different reward is received and* ***controllable*** *because the transition kernel is action dependent. For completeness, we recall that a finite Markov Chain $(\mathcal{X}, p, \rho)$ can be defined as follows:*

- *$\mathcal{X}$: the finite state space of cardinal $|\mathcal{X}|$.*
- *p: a transition kernel from $\mathcal{X}$ to $\mathcal{X}$ which means that $p(y|x)$ denotes the probability of transitioning to state $y \in \mathcal{X}$ in state $x \in \mathcal{X}$.*
- *$\rho \in \Delta_{\mathcal{X}}$: the initial state distribution and $\rho(x)$ is the probability to start in state x.*

A discrete stochastic process $((X_t)_{t\in\mathbb{N}}, \mathbb{P}_\rho)$ taking its values on $\mathcal{X}$ with probability measure $\mathbb{P}_\rho$ is said to be associated to the Markov-chain $(\mathcal{X}, p, \rho)$ if for all $n \in \mathbb{N}^*$ and for all $(x_k)_{k=0}^n \in \mathcal{X}^{n+1}$:

$$\mathbb{P}_\rho((X_k)_{k=0}^n = (x_k)_{k=0}^n) = \rho(x_0) \prod_{k=1}^{n} p(x_k|x_{k-1}).$$

In particular, if $1 \le k \le n$ and $\mathbb{P}_\rho(X_{k-1} = x_{k-1}) > 0$, then:

$$\mathbb{P}_\rho(X_k = x_k | X_{k-1} = x_{k-1}) = p(x_k|x_{k-1}).$$

Remark 2. *Markov chains for advanced readers.*
This remark is not essential for the comprehension of the remaining material but can be helpful for advanced readers with notions of measure theory. A finite Markov Chain $(\mathcal{X}, p, \rho)$ can be associated to a canonical discrete stochastic process $((X_t)_{t\in\mathbb{N}}, \Omega, \mathcal{F}, \mathbb{P}_\rho)$. More precisely, this discrete stochastic process can be explicitly constructed and is called the canonical process associated to the Markov chain $(\mathcal{X}, p, \rho)$. The following theorem (not proven) describes this important result.

Theorem 5. *Let consider the measurable space $(\Omega, \mathcal{F})$ where $\Omega = \mathcal{X}^{\mathbb{N}}$ is the space of possible trajectories and where $\mathcal{F}$ is a σ-algebra that we will define later. Let $t \in \mathbb{N}$, and let define the random variable $X_t \in \mathcal{X}^{\Omega}$:*

$$\forall \omega = (x_n)_{n\in\mathbb{N}} \in \Omega, \quad X_t(w) = x_t.$$

$(X_t)_{t\in\mathbb{N}}$ is called the set of coordinates random variables. The σ-algebra $\mathcal{F}$ is then defined as the smallest σ-algebra that makes all the random variables $(X_t)_{t\in\mathbb{N}}$ measurable. Then, there exists a unique probability measure $\mathbb{P}_\rho \in [0,1]^{\mathcal{F}}$ such that $((X_t)_{t\in\mathbb{N}}, \Omega, \mathcal{F}, \mathbb{P}_\rho)$ can be associated to the Markov-chain $(\mathcal{X}, p, \rho)$. The probability measure $\mathbb{P}_\rho$ verifies for all $n \in \mathbb{N}^*$ and for all $(x_k)_{k=0}^n$:

$$\mathbb{P}_\rho((X_k)_{k=0}^n = (x_k)_{k=0}^n) = \rho(x_0) \prod_{k=1}^{n} p(x_k|x_{k-1}).$$

With this theorem, we are assured of the existence of a canonical discrete stochastic process that can be associated to a given Markov chain. In the remaining, we will always refer to the canonical stochastic process $((X_t)_{t\in\mathbb{N}}, \Omega, \mathcal{F}, \mathbb{P}_\rho)$ when talking of the stochastic process associated to the Markov chain $(\mathcal{X}, p, \rho)$. In addition, when no confusion is possible, we will drop the notations of the measurable space $(\Omega, \mathcal{F})$ and just note $((X_t)_{t\in\mathbb{N}}, \mathbb{P}_\rho)$ the canonical process associated to the Markov chain $(\mathcal{X}, p, \rho)$. The canonical stochastic process allows us to assign probabilities over the possible trajectories in Ω.

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

- **Trajectories**: By choosing to follow a stochastic policy π, the agent places itself in a Markov chain. Indeed, by fixing the policy π, the probability to get from a state $y \in \mathcal{X}$ starting from $x \in \mathcal{X}$ and then following policy π becomes $\sum_{a\in\mathcal{A}} p(y|x,a)\pi(a|x)$. This naturally defines a transition kernel from $\mathcal{X}$ to $\mathcal{X}$. More precisely, by fixing the policy π, the agent places itself in the Markov chain $(\mathcal{X}, p_{\mathcal{X}}^{\pi}, \rho)$ where:

$$\forall (x,y) \in \mathcal{X}^2, \quad p_{\mathcal{X}}^{\pi}(y|x) = \sum_{a\in\mathcal{A}} p(y|x,a)\pi(a|x).$$

  This Markov chain is called the state Markov chain and the associated discrete stochastic process $((X_t)_{t\in\mathbb{N}}, \mathbb{P}_\rho^\pi)$ is the state-trajectory process.

  The same remark can be done for state-action couples. Indeed, by fixing the policy π, the probability to reach the state-action $(y,b) \in \mathcal{X}\times\mathcal{A}$ starting from $(x,a) \in \mathcal{X}\times\mathcal{A}$ and then following policy π is $p(y|x,a)\pi(b|y)$. This naturally defines a transition kernel from $\mathcal{X}\times\mathcal{A}$ to $\mathcal{X}\times\mathcal{A}$. More precisely, by fixing the policy π, the agent places itself in the Markov chain $(\mathcal{X}\times\mathcal{A}, p_{\mathcal{X}\times\mathcal{A}}^{\pi}, \mu)$ where:

$$\forall ((x,a),(y,b)) \in (\mathcal{X}\times\mathcal{A})^2, \quad p_{\mathcal{X}\times\mathcal{A}}^{\pi}((y,b)|(x,a)) = p(y|x,a)\pi(b|y),$$

  and μ a probability distribution over state-action couples. This Markov chain is called the state-action Markov chain and the associated discrete stochastic process $((X_t, A_t)_{t\in\mathbb{N}}, \mathbb{P}_\mu^\pi)$ is the state-action-trajectory process.

- **Discounted Cumulative returns**: We define the state discounted cumulative return as the discounted sum of rewards over the state-trajectory process:

$$Z_{\mathcal{X}}((X_t)_{t\in\mathbb{N}}) = \sum_{t\geq 0} \gamma^t r^\pi(X_t),$$

  where $r^\pi \in \mathbb{R}^{\mathcal{X}}$ is defined by $\forall x \in \mathcal{X}, r^\pi(x) = \sum_{a\in\mathcal{A}} \pi(a|x) r(x,a)$. By definition, $r^\pi(x)$ is the average reward collected on x by following policy π. We also define the state-action discounted cumulative return as the discounted sum of rewards over the state-action-trajectory process:

$$Z_{\mathcal{X}\times\mathcal{A}}((X_t, A_t)_{t\in\mathbb{N}}) = \sum_{t\geq 0} \gamma^t r(X_t, A_t).$$

  These discounted cumulative returns are the quantities we would like to optimize. However, since trajectories are random processes, so is their returns, and to properly optimize returns, we first need to define a notion of expected cumulative return. In this course, we will focus on *infinite horizon discounted return.* Note that other notions of returns such as undiscounted average return or finite horizon cumulative return exist.

- **Expected cumulative returns**: We define the value (or V-)function and state-action value (or Q-)function as

$$\forall x \in \mathcal{X}, \quad V^\pi(x) = \mathbb{E}_x^\pi\Big[Z_\mathcal{X}((X_t)_{t\in\mathbb{N}})\Big] = \mathbb{E}_x^\pi\Big[\sum_{t\geq 0}\gamma^t r^\pi(X_t)\Big],$$
$$\forall (x,a) \in \mathcal{X}\times\mathcal{A}, \quad Q^\pi(x,a) = \mathbb{E}_{x,a}^\pi\Big[Z_{\mathcal{X}\times\mathcal{A}}((X_t,A_t)_{t\in\mathbb{N}})\Big] = \mathbb{E}_{x,a}^\pi\Big[\sum_{t\geq 0}\gamma^t r(X_t,A_t)\Big],$$

where $((X_t)_{t\in\mathbb{N}}, \mathbb{P}_x^\pi)$ is the state-trajectory process associated with the state Markov chain $(\mathcal{X}, p_\mathcal{X}^\pi, \delta_x)$ and where $((X_t,A_t)_{t\in\mathbb{N}}, \mathbb{P}_{x,a}^\pi)$ is the state-action-trajectory process associated with the state-action Markov chain $(\mathcal{X}\times\mathcal{A}, p_{\mathcal{X}\times\mathcal{A}}^\pi, \delta_{x,a})$. The distributions δ_x and $\delta_{x,a}$ are the Dirac distributions over $\mathcal{X}$ on state x and over $\mathcal{X}\times\mathcal{A}$ on state-action (x,a) respectively. In addition, $\mathbb{E}_x^\pi$ and $\mathbb{E}_{x,a}^\pi$ are the expectation related to the probability measure $\mathbb{P}_x^\pi$ and $\mathbb{P}_{x,a}^\pi$ respectively. Intuitively, $V^\pi(x)$ is the discounted and expected cumulative return starting from state x and then following policy π. Similarly, $Q^\pi(x,a)$ is the discounted and expected cumulative return starting from state-action couple (x,a) and then following policy π. The quantities $V^\pi(x)$ and $Q^\pi(x,a)$ are well defined. Indeed $Z_\mathcal{X}$ can be seen a function from $\mathcal{X}^\mathbb{N}$ to $\mathbb{R}$:

$$\forall \omega = (x_n)_{n\in\mathbb{N}} \in \mathcal{X}^\mathbb{N}, Z_\mathcal{X}(\omega) = \sum_{t\geq 0}\gamma^t r^\pi(x_t).$$

Therefore:

$$\begin{aligned}\forall \omega = (x_n)_{n\in\mathbb{N}} \in \mathcal{X}^\mathbb{N}, |Z_\mathcal{X}(\omega)| &= \sum_{t\geq 0}\gamma^t|r^\pi(x_t)|,\\ &= \sum_{t\geq 0}\gamma^t\sum_{a\in\mathcal{A}}\pi(a|x)|r(x_t,a)|,\\ &\leq \max_{(x,a)\in\mathcal{X}\times\mathcal{A}}|r(x,a)|\sum_{t\geq 0}\gamma^t,\\ &= \frac{\|r\|_\infty}{1-\gamma},\end{aligned}$$

where $\|r\|_\infty = \max_{(x,a)\in\mathcal{X}\times\mathcal{A}}|r(x,a)|$ is the infinite norm of r. Thus, $Z_\mathcal{X}$ is a bounded function which makes $V^\pi(x)$ the expectation of a bounded random variable and therefore exists. The same reasoning can be applied to $Q^\pi(x,a)$.

1.4.1 The Bellman Equations.

The Bellman equations are at the heart of RL. They express the fact that it is possible to compute the cumulative return by using the cumulative return at the next state. This is a simple bootstrapping trick. Such a result is possible because of the time-homogeneous Markov property of Markov chains.

Remark 3. *Time-homogeneous Markov Property*
Let us consider the Markov chain $(\mathcal{X}, p, \rho)$ and its associated process $((X_t)_{t\in\mathbb{N}}, \mathbb{P}_\rho)$. Then, the time-homogeneous Markov property is: let $k\in\mathbb{N}^*$ and $x\in\mathcal{X}$ such that $\mathbb{P}_\rho(X_k = x) > 0$, then:

$$\forall t\in\mathbb{N}, \forall y\in\mathcal{X}, \quad \mathbb{P}_\rho(X_{t+k} = y|X_k = x) = \mathbb{P}_x(X_t = y),$$

where $((X_t)_{t\in\mathbb{N}}, \mathbb{P}_x)$ is the process associated with the Markov chain $(\mathcal{X}, p, \delta_x)$ and δ_x is the Dirac distribution over $\mathcal{X}$ on state x. Basically, this property is just saying that conditioning on an event at time k is equivalent at starting the Markov from scratch at this particular event.

Theorem 6. *For a given policy $\pi\in\Pi$, the value function V^π verifies:*

$$\begin{aligned}\forall x\in\mathcal{X}, \quad V^\pi(x) &= \sum_{a\in\mathcal{A}}\pi(a|x)r(x,a) + \gamma\sum_{y\in\mathcal{X}}p_\mathcal{X}^\pi(y|x)V^\pi(y),\\ &= \sum_{a\in\mathcal{A}}\pi(a|x)r(x,a) + \gamma\sum_{y\in\mathcal{X}}\sum_{a\in\mathcal{A}}\pi(a|x)p(y|x,a)V^\pi(y),\end{aligned}$$

and the action value function Q^π verifies:

$$\begin{aligned}\forall (x,a)\in\mathcal{X}\times\mathcal{A}, \quad Q^\pi(x,a) &= r(x,a) + \gamma\sum_{y\in\mathcal{X}}\sum_{b\in\mathcal{A}}p_{\mathcal{X}\times\mathcal{A}}^\pi((y,b)|(x,a))Q^\pi(y,b),\\ &= r(x,a) + \gamma\sum_{y\in\mathcal{X}}\sum_{b\in\mathcal{A}}p(y|x,a)\pi(b|y)Q^\pi(y,b),\end{aligned}$$

Proof. Let $x \in \mathcal{X}$, by definition of $V^\pi(x)$:

$$\begin{aligned}
V^\pi(x) &= \mathbb{E}^\pi_x[\sum_{t\geq 0} \gamma^t r^\pi(X_t)],\\
&\underset{(a)}{=} \mathbb{E}^\pi_x[r^\pi(X_0)] + \mathbb{E}^\pi_x[\sum_{t\geq 1} \gamma^t r^\pi(X_t)],\\
&\underset{(b)}{=} \mathbb{E}^\pi_x[r^\pi(X_0)] + \gamma\mathbb{E}^\pi_x[\sum_{t\geq 0} \gamma^t r^\pi(X_{t+1})],\\
&\underset{(c)}{=} \sum_{y\in\mathcal{X}} \mathbb{P}^\pi_x(X_0 = y) r^\pi(y) + \gamma \sum_{t\geq 0}\gamma^t \sum_{z\in\mathcal{X}} \mathbb{P}^\pi_x(X_{t+1} = z) r^\pi(z),\\
&\underset{(d)}{=} r^\pi(x) + \gamma \sum_{t\geq 0}\gamma^t \sum_{z\in\mathcal{X}} \mathbb{P}^\pi_x(X_{t+1} = z) r^\pi(z),\\
&\underset{(e)}{=} r^\pi(x) + \gamma \sum_{t\geq 0}\gamma^t \sum_{z\in\mathcal{X}} \sum_{y\in\mathcal{X},\mathbb{P}^\pi_x(X_1=y)>0} \mathbb{P}^\pi_x(X_{t+1} = z|X_1 = y)\mathbb{P}^\pi_x(X_1 = y) r^\pi(z),\\
&\underset{(f)}{=} r^\pi(x) + \gamma \sum_{t\geq 0}\gamma^t \sum_{z\in\mathcal{X}} \sum_{y\in\mathcal{X},\mathbb{P}^\pi_x(X_1=y)>0} \mathbb{P}^\pi_x(X_{t+1} = z|X_1 = y)p^\pi_{\mathcal{X}}(y|x) r^\pi(z),\\
&\underset{(g)}{=} r^\pi(x) + \gamma \sum_{t\geq 0}\gamma^t \sum_{z\in\mathcal{X}} \sum_{y\in\mathcal{X},p^\pi_{\mathcal{X}}(y|x)>0} \mathbb{P}^\pi_y(X_t = z)p^\pi_{\mathcal{X}}(y|x) r^\pi(z),\\
&\underset{(h)}{=} r^\pi(x) + \gamma \sum_{t\geq 0}\gamma^t \sum_{z\in\mathcal{X}} \sum_{y\in\mathcal{X}} \mathbb{P}^\pi_y(X_t = z)p^\pi_{\mathcal{X}}(y|x) r^\pi(z),\\
&\underset{(i)}{=} r^\pi(x) + \gamma \sum_{y\in\mathcal{X}} p^\pi_{\mathcal{X}}(y|x) \sum_{t\geq 0}\gamma^t \sum_{z\in\mathcal{X}} \mathbb{P}^\pi_y(X_t = z) r^\pi(z),\\
&\underset{(j)}{=} r^\pi(x) + \gamma \sum_{y\in\mathcal{X}} p^\pi_{\mathcal{X}}(y|x) \mathbb{E}^\pi_y[\sum_{t\geq 0}\gamma^t r^\pi(X_t)],\\
&\underset{(k)}{=} r^\pi(x) + \gamma \sum_{y\in\mathcal{X}} p^\pi_{\mathcal{X}}(y|x)) V^\pi(y),\\
&\underset{(l)}{=} \sum_{a\in\mathcal{A}} \pi(a|x) r(x,a) + \gamma \sum_{y\in\mathcal{X}}\sum_{a\in\mathcal{A}} p(y|x,a)\pi(a|x) V^\pi(y).
\end{aligned}$$

(a) is true because of the linearity of the expectation, (b) is a change of variable $t+1 \leftarrow t$ in the summation, (c) is by definition of expectation, (d) is true because $\mathbb{P}^\pi_x(X_0 = y) = 1$ if and only if $y = x$ and 0 otherwise, (e) is true by the law of total probability $\mathbb{P}^\pi_x(X_{t+1} = z) = \sum_{y\in\mathcal{X},\mathbb{P}^\pi_x(X_1=y)>0} \mathbb{P}^\pi_x(X_{t+1} = z|X_1 = y)\mathbb{P}^\pi_x(X_1 = y)$, (f) is true because by the law of total probability $\mathbb{P}^\pi_x(X_1 = y) = \sum_{z\in\mathcal{X},\mathbb{P}^\pi_x(X_0=z)>0} \mathbb{P}^\pi_x(X_1 = y|X_0 = z)\mathbb{P}^\pi_x(X_0 = z) = \mathbb{P}^\pi_x(X_1 = y|X_0 = x)$ and $\mathbb{P}^\pi_x(X_1 = y|X_0 = x) = p^\pi_{\mathcal{X}}(y|x)$, (g) is true because of the Markov property (see Remark 3) applied on the state Markov chain $(X_t)_{t\in\mathbb{N}}$ we have $\mathbb{P}^\pi_x(X_{t+1} = z|X_1 = y) = \mathbb{P}^\pi_y(X_t = z)$ and $p^\pi_{\mathcal{X}}(y|x) = \mathbb{P}^\pi_x(X_1 = y)$, (h) we can change the summation $\sum_{y\in\mathcal{X},p^\pi_{\mathcal{X}}(y|x)>0} \mathbb{P}^\pi_y(X_t = z)p^\pi_{\mathcal{X}}(y|x)r^\pi(z)$ to $\sum_{y\in\mathcal{X}} \mathbb{P}^\pi_y(X_t = z)p^\pi_{\mathcal{X}}(y|x)r^\pi(z)$ because we add only null terms, (i) is just a rearrangement of the sums, (j) is true by definition of the expectation, (k) is true by definition of the value function, (l) is true by definitions of $r^\pi(x) = \sum_{a\in\mathcal{A}} \pi(a|x)r(x,a)$ and of $p^\pi_{\mathcal{X}}(y|x)) = \sum_{a\in\mathcal{A}} p(y|x,a)\pi(a|x)$. This concludes the proof for V^π.

Now let us prove the result for Q^π. Let $(x,a) \in \mathcal{X} \times \mathcal{A}$, by definition of $Q^\pi(x,a)$:

$$\begin{aligned}
Q^\pi(x,a) &= \mathbb{E}^\pi_{x,a}[\sum_{t\geq 0} \gamma^t r(X_t, A_t)],\\
&\underset{(a)}{=} \mathbb{E}^\pi_{x,a}[r(X_0,A_0)] + \mathbb{E}^\pi_{x,a}[\sum_{t\geq 1} \gamma^t r(X_t, A_t)],\\
&\underset{(b)}{=} \mathbb{E}^\pi_{x,a}[r(X_0,A_0)] + \gamma\mathbb{E}^\pi_{x,a}[\sum_{t\geq 0} \gamma^t r(X_{t+1}, A_{t+1})],\\
&\underset{(c)}{=} \sum_{(y,b)\in\mathcal{X}\times\mathcal{A}} \mathbb{P}^\pi_{x,a}((X_0,A_0)=(y,b))r(y,b) + \gamma\sum_{t\geq 0}\gamma^t \sum_{(z,c)\in\mathcal{X}\times\mathcal{A}} \mathbb{P}^\pi_{x,a}((X_{t+1},A_{t+1})=(z,c))r(z,c),\\
&\underset{(d)}{=} r(x,a) + \gamma\sum_{t\geq 0}\gamma^t \sum_{(z,c)\in\mathcal{X}\times\mathcal{A}} \mathbb{P}^\pi_{x,a}((X_{t+1}),A_{t+1}=(z,c))r(z,c),\\
&\underset{(e)}{=} r(x,a) + \gamma\sum_{t\geq 0}\gamma^t \sum_{(z,c)\in\mathcal{X}\times\mathcal{A}} \sum_{\substack{(y,b)\in\mathcal{X}\times\mathcal{A}\\ \mathbb{P}^\pi_{x,a}((X_1,A_1)=(y,b))>0}} \mathbb{P}^\pi_{x,a}((X_{t+1},A_{t+1})=(z,c)|(X_1,A_1)=(y,b))\mathbb{P}^\pi_{x,a}((X_1,A_1)=(y,b))r(z,c),\\
&\underset{(f)}{=} r(x,a) + \gamma\sum_{t\geq 0}\gamma^t \sum_{(z,c)\in\mathcal{X}\times\mathcal{A}} \sum_{\substack{(y,b)\in\mathcal{X}\times\mathcal{A}\\ \mathbb{P}^\pi_{x,a}((X_1,A_1)=(y,b))>0}} \mathbb{P}^\pi_{x,a}((X_{t+1},A_{t+1})=(z,c)|(X_1,A_1)=(y,b))p^\pi_{\mathcal{X}\times\mathcal{A}}((y,b)|(x,a))r(z,c),\\
&\underset{(g)}{=} r(x,a) + \gamma\sum_{t\geq 0}\gamma^t \sum_{(z,c)\in\mathcal{X}\times\mathcal{A}} \sum_{\substack{(y,b)\in\mathcal{X}\times\mathcal{A}\\ p^\pi_{\mathcal{X}\times\mathcal{A}}((y,b)|(x,a))>0}} \mathbb{P}^\pi_{y,b}((X_t,A_t)=(z,c))p^\pi_{\mathcal{X}\times\mathcal{A}}((y,b)|(x,a))r(z,c),\\
&\underset{(h)}{=} r(x,a) + \gamma\sum_{t\geq 0}\gamma^t \sum_{(z,c)\in\mathcal{X}\times\mathcal{A}} \sum_{(y,b)\in\mathcal{X}\times\mathcal{A}} \mathbb{P}^\pi_{y,b}((X_t,A_t)=(z,c))p^\pi_{\mathcal{X}\times\mathcal{A}}((y,b)|(x,a))r(z,c),\\
&\underset{(i)}{=} r(x,a) + \gamma \sum_{(y,b)\in\mathcal{X}\times\mathcal{A}} p^\pi_{\mathcal{X}\times\mathcal{A}}((y,b)|(x,a))\sum_{t\geq 0}\gamma^t \sum_{(z,c)\in\mathcal{X}\times\mathcal{A}} \mathbb{P}^\pi_{y,b}(X_t=z,A_t=c)r(z,c),\\
&\underset{(j)}{=} r(x,a) + \gamma \sum_{(y,b)\in\mathcal{X}\times\mathcal{A}} p^\pi_{\mathcal{X}\times\mathcal{A}}((y,b)|(x,a))\mathbb{E}^\pi_{y,b}[\sum_{t\geq 0}\gamma^t r(X_t,A_t)],\\
&\underset{(k)}{=} r(x,a) + \gamma \sum_{(y,b)\in\mathcal{X}\times\mathcal{A}} p^\pi_{\mathcal{X}\times\mathcal{A}}((y,b)|(x,a)))Q^\pi(y,b),\\
&\underset{(l)}{=} r(x,a) + \gamma \sum_{(y,b)\in\mathcal{X}\times\mathcal{A}} p(y|x,a)\pi(b|y)Q^\pi(y,b).
\end{aligned}$$

(a) is true because of the linearity of the expectation, (b) is a change of variable $t+1 \leftarrow t$ in the summation, (c) is by definition of expectation, (d) is true because $\mathbb{P}^\pi_{x,a}((X_0,A_0)=(y,b))=1$ if and only if $(y,b)=(x,a)$ and 0 otherwise, (e) is true by the law of total probability, (f) by the law of total probability $\mathbb{P}^\pi_{x,a}((X_1,A_1)=(y,b)) = p^\pi_{\mathcal{X}\times\mathcal{A}}((y,b)|(x,a))$, (g) is true because of the Markov property (see Remark 3), (h) we can change the summation over all pairs (y,b) because we add only null terms, (i) is just a rearrangement of the sums, (j) is true by definition of the expectation, (k) is true by definition of the action-value function, (l) is true by definition of $p^\pi_{\mathcal{X}\times\mathcal{A}}((y,b)|(x,a)))$. This concludes the proof for Q^π. □

We can now formally define the goal of Reinforcement Learning algorithms: the goal of a reinforcement learning algorithm in a given MDP is to find an **optimal policy** $\pi^\star \in \Pi$, such that for all $x \in \mathcal{X}$, and for any policy $\pi \in \Pi$, $V^{\pi^\star}(x) \geq V^\pi(x)$.

1.5 Reminders of Dynamic Programming

In this section, we are going to present 5 theorems that allows us from the definitions of Bellman operators to show the existence of an optimal policy. These 5 theorems are also at the core of the classical Dynamic Programming schemes such as Value Iteration and Policy Iteration.

Dynamic Programming is intrinsically linked to the definitions of Bellman operators:

- **Evaluation operator for V-functions**: The evaluation operator $B^\pi \in \mathbb{R}^{\mathcal{X}^{\mathbb{R}^{\mathcal{X}}}}$ is defined as follows:

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

- **Optimality operator for V-functions**: The optimality operator $B^\star \in \mathbb{R}^{\mathcal{X}^{\mathbb{R}^{\mathcal{X}}}}$ is defined as follows:

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

- **Evaluation operator for Q-functions**: The evaluation operator $T^\pi \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}^{\mathbb{R}^{\mathcal{X}\times\mathcal{A}}}}$ is defined as follows:

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

- **Optimality operator for Q-functions**: The optimality operator $T^\star \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}^{\mathbb{R}^{\mathcal{X}\times\mathcal{A}}}}$ is defined as follows:

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

Remark 4. *Remark on matricial notations.*
Evaluation operators are affine operators and they can be written with matricial notations. Indeed, if we define the stochastic square matrix $P^\pi_{\mathcal{X}\times\mathcal{A}}$ of size $|\mathcal{X}|\times|\mathcal{A}|$ such that:

$$\forall ((x,a),(y,b)) \in (\mathcal{X}\times\mathcal{A})^2, \quad P^\pi_{\mathcal{X}\times\mathcal{A}}((x,a),(y,b)) = p^\pi_{\mathcal{X}\times\mathcal{A}}((y,b)|(x,a))) = p(y|x,a)\pi(b|y),$$

then:

$$\forall Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}, \quad T^\pi[Q] = r + \gamma P^\pi_{\mathcal{X}\times\mathcal{A}} Q,$$

where r and Q are seen as column vectors. The same can be done with B^π. Let define $P^\pi_{\mathcal{X}}$ the stochastic square matrix of size $|\mathcal{X}|$ such that:

$$\forall (x,y) \in \mathcal{X}^2, \quad P^\pi_{\mathcal{X}}(x,y) = p^\pi_{\mathcal{X}}(y|x) = \sum_{a\in\mathcal{A}} p(y|x,a)\pi(a|x),$$

then:

$$\forall V \in \mathbb{R}^{\mathcal{X}}, \quad B^\pi[V] = r^\pi + \gamma P^\pi_{\mathcal{X}} V,$$

where r^π and V are seen as column vectors.

The Bellman operators have the following basic properties.

Property 1. *Monotonocity.*
The evaluation and optimality operators for V-functions are monotonous. Let $V_1 \in \mathbb{R}^X$ and $V_2 \in \mathbb{R}^X$ such that $V_1 \le V_2$ and let $\pi \in \Pi$, then

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

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

Proof.

$$\begin{aligned} V_1 \le V_2 &\underset{(a)}{\Longrightarrow} P^\pi_{\mathcal{X}} V_1 \le P^\pi_{\mathcal{X}} V_2, \\ &\underset{(b)}{\Longrightarrow} \gamma P^\pi_{\mathcal{X}} V_1 \le \gamma P^\pi_{\mathcal{X}} V_2, \\ &\underset{(c)}{\Longrightarrow} r^\pi + \gamma P^\pi_{\mathcal{X}} V_1 \le r^\pi + \gamma P^\pi_{\mathcal{X}} V_2, \\ &\underset{(d)}{\Longrightarrow} B^\pi[V_1] \le B^\pi[V_2], \end{aligned}$$

where (a) is true because $P^\pi_{\mathcal{X}}$ is non-negative, (b) is true because $\gamma \in [0,1)$, c is true because we add r^π on both sides of the inequality and d is true by definition of B^π.

$$\begin{aligned}
\forall x \in \mathcal{X}, V_1(x) \leq V_2(x) &\underset{(a)}{\Longrightarrow} \forall (x,a) \in \mathcal{X}\times\mathcal{A}, \sum_{y\in\mathcal{X}} p(y|x,a)V_1(y) \leq \sum_{y\in\mathcal{X}} p(y|x,a)V_2(y),\\
&\underset{(b)}{\Longrightarrow} \forall (x,a) \in \mathcal{X}\times\mathcal{A}, \gamma\sum_{y\in\mathcal{X}} p(y|x,a)V_1(y) \leq \gamma\sum_{y\in\mathcal{X}} p(y|x,a)V_2(y),\\
&\underset{(c)}{\Longrightarrow} \forall (x,a) \in \mathcal{X}\times\mathcal{A}, r(x,a) + \gamma\sum_{y\in\mathcal{X}} p(y|x,a)V_1(y) \leq r(x,a) + \gamma\sum_{y\in\mathcal{X}} p(y|x,a)V_2(y),\\
&\underset{(d)}{\Longrightarrow} \forall x \in \mathcal{X}, \max_{a\in\mathcal{A}}\left[r(x,a) + \gamma\sum_{y\in\mathcal{X}} p(y|x,a)V_1(y)\right] \leq \max_{a\in\mathcal{A}}\left[r(x,a) + \gamma\sum_{y\in\mathcal{X}} p(y|x,a)V_2(y)\right],\\
&\underset{(e)}{\Longrightarrow} B^\star[V_1] \leq B^\star[V_2],
\end{aligned}$$

where (a) is true because $\forall (x,a,y) \in \mathcal{X}\times\mathcal{A}\times\mathcal{X}, p(y|x,a)$ is non-negative, (b) is true because $\gamma \in [0,1)$, (c) is true because we add $r(x,a)$ on both sides of the inequality, (d) is true because the max operator is order preserving and (e) is true by definition of $B^\star$.

A similar proof can be derived for evaluation and optimality operators for Q-functions. □

Property 2. *Domination of the optimality operators.*
The optimality operator for V-functions dominates the evaluation 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.

Proof. First we recall the following elementary extremal property:

$$\forall s \in \mathcal{S}, \min_{s\in\mathcal{S}} f(s) \leq \sum_{s\in\mathcal{S}} f(s)p(s) \leq \max_{s\in\mathcal{S}} f(s),$$

where $\mathcal{S}$ is a finite set, $f \in \mathbb{R}^{\mathcal{S}}$ and p is a real distribution over $\mathcal{S}$. We also say that the max operator dominates the expectation operator. Therefore, we have by the extremal property:

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

A similar proof can be derived for evaluation and optimality operators for Q-functions. □

Here are the 5 fundamental theorems of Dynamical Programming that allows us to show the existence of an optimal policy:

Theorem 7. *Fixed-points of the evaluation operators.*
The operator B^π is a γ-contraction with unique fixed-point V^π. Similarly, the operator 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. We have already shown via the Bellman equations that V^π is a fixed point of B^π and Q^π a fixed point of T^π. To show the uniqueness we just need to show that B^π and T^π are γ-contractions and the result will follow by the Banach fixed point-theorem. Therefore, let show that B^π is a γ-contraction. Let $V_1 \in \mathbb{R}^{\mathcal{X}}$ and $V_2 \in \mathbb{R}^{\mathcal{X}}$ and $x \in \mathcal{X}$:

$$\begin{aligned}
|B^\pi[V_1](x) - B^\pi[V_2](x)| &\underset{(a)}{=} |\gamma\sum_{y\in\mathcal{X}}\sum_{a\in\mathcal{A}} p(y|x,a)\pi(a|x)(V_1(y) - V_2(y))|,\\
&\underset{(b)}{\leq} \gamma\sum_{y\in\mathcal{X}}\sum_{a\in\mathcal{A}} p(y|x,a)\pi(a|x)|V_1(y) - V_2(y)|,\\
&\underset{(c)}{=} \gamma\sum_{y\in\mathcal{X}} p_{\mathcal{X}}(y|x)|V_1(y) - V_2(y)|,\\
&\underset{(d)}{\leq} \gamma\max_{y\in\mathcal{X}}|V_1(y) - V_2(y)|,\\
&\underset{(e)}{=} \gamma\|V_1 - V_2\|_\infty,
\end{aligned}$$

where (a) is true by definition of B^π, (b) is true because the absolute value of a sum is lower than the sum of the absolute values and because $p(y|x,a)\pi(a|x) \geq 0$, (c) is true by definition of $p_\mathcal{X}(y|x)$, (d) is true by the extremal property and (e) is true by definition of the infinite norm: if $\mathcal{S}$ is a finite space and $f \in \mathbb{R}^\mathcal{S}$ then $\max_{s\in\mathcal{S}} |f(s)| = \|f\|_\infty$. Finally, as this inequality is true for any $x \in \mathcal{X}$, then we have $\|B^\pi[V_1] - B^\pi[V_2]\|_\infty \leq \gamma\|V_1 - V_2\|_\infty$.

Let show that T^π is a γ-contraction. Let $Q_1 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ and $Q_2 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ and $(x,a) \in \mathcal{X}\times\mathcal{A}$:

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

where (a) is true by definition of Y^π, (b) is true because the absolute value of a sum is lower than the sum of the absolute values and because $p(y|x,a)\pi(b|x) \geq 0$, (c) is true by definition of $p_{\mathcal{X}\times\mathcal{A}}((y,b)|(x,a))$, (d) is true by the extremal property and (e) is true by definition of the infinite norm. Finally, as this inequality is true for any $(x,a) \in \mathcal{X}\times\mathcal{A}$, then we have $\|T^\pi[Q_1] - T^\pi[Q_2]\|_\infty \leq \gamma\|Q_1 - Q_2\|_\infty$.

□

Theorem 8. *Fixed-points of the optimality operators.*
The operator $B^\star$ is a γ-contraction, its unique fixed-point is noted $V^\star$ and called the optimal value function. Similarly, the operator $T^\star$ is a γ-contraction, its unique fixed-point is noted $Q^\star$ and called the optimal action-value function:

$$\begin{aligned}
B^\star[V^\star] &= V^\star, \\
T^\star[Q^\star] &= Q^\star.
\end{aligned}$$

Proof. To prove the theorem, we just need to show that $B^\star$ and $T^\star$ are contractions and the result follows by the Banach fixed-point theorem. This proof need the following property. Let $\mathcal{S}$ a finite state, $f \in \mathbb{R}^\mathcal{S}$ and $g \in \mathbb{R}^\mathcal{S}$, then:

$$|\max_{s\in\mathcal{S}} f(s) - \max_{s\in\mathcal{S}} g(s)| \leq \max_{s\in\mathcal{S}} |f(s) - g(s)|.$$

This property is not so trivial and can be proven by remarking that:

$$\forall s \in \mathcal{S}, f(s) \leq |f(s) - g(s)| + g(s).$$

Therefore:

$$\begin{aligned}
\max_{s\in\mathcal{S}} f(s) &\underset{(a)}{\leq} \max_{s\in\mathcal{S}}(|f(s) - g(s)| + g(s)), \\
&\underset{(b)}{\leq} \max_{s\in\mathcal{S}} |f(s) - g(s)| + \max_{s\in\mathcal{S}} g(s), \\
\max_{s\in\mathcal{S}} f(s) - \max_{s\in\mathcal{S}} g(s) &\underset{(b)}{\leq} \max_{s\in\mathcal{S}} |f(s) - g(s)|.
\end{aligned}$$

where (a) is true by taking the maximum on both sides, (b) is true by triangular inequality and (c) is true by subtracting $\max_{s\in\mathcal{S}} g(s)$ on both sides. By permuting the roles of f and g we get the property. Now let us start by proving that $B^\star$ is a

γ-contraction. Let $V_1 \in \mathbb{R}^{\mathcal{X}}$ and $V_2 \in \mathbb{R}^{\mathcal{X}}$ and $x \in \mathcal{X}$:

$$\begin{aligned}
|B^\star[V_1](x) - B^\star[V_2](x)| &\underset{(a)}{=} \left|\max_{a\in\mathcal{A}}\left[r(x,a) + \gamma\sum_{y\in\mathcal{X}} p(y|x,a)V_1(y)\right] - \max_{a\in\mathcal{A}}\left[r(x,a) + \gamma\sum_{y\in\mathcal{X}} p(y|x,a)V_2(y)\right]\right|,\\
&\underset{(b)}{\leq} \max_{a\in\mathcal{A}}\left|\gamma\sum_{y\in\mathcal{X}} p(y|x,a)[V_1(y) - V_2(y)]\right|,\\
&\underset{(c)}{\leq} \gamma\max_{a\in\mathcal{A}}\sum_{y\in\mathcal{X}} p(y|x,a)\left|V_1(y) - V_2(y)\right|,\\
&\underset{(d)}{\leq} \gamma\max_{a\in\mathcal{A}}\max_{y\in\mathcal{X}}\left|V_1(y) - V_2(y)\right|,\\
&\underset{(e)}{\leq} \gamma\max_{y\in\mathcal{X}}\left|V_1(y) - V_2(y)\right|,\\
&\underset{(f)}{\leq} \gamma\|V1 - V2\|_\infty,
\end{aligned}$$

where (a) is true by definition of $B^\star$, (b) is true because $|\max_{s\in\mathcal{S}} f(s) - \max_{s\in\mathcal{S}} g(s)| \leq \max_{s\in\mathcal{S}} |f(s) - g(s)|$, (c) is true because the absolute value of a sum is lower than the sum of the absolute values and because $p(y|x,a) \geq 0$, (d) is true by the extremal property, (e) is true because the quantity $|V_1(y) - V_2(y)|$ does not depend on $a \in \mathcal{A}$ and (f) is by definition of the infinite norm. Finally, as this inequality is true for any $x \in \mathcal{X}$, then we have $\|B^\star[V_1] - B^\star[V_2]\|_\infty \leq \gamma\|V_1 - V_2\|_\infty$.

Now let us prove that $T^\star$ is a γ-contraction. Let $Q_1 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ and $V_2 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ and $(x,a) \in^{\mathcal{X}} \times\mathcal{A}$:

$$\begin{aligned}
|T^\star[Q_1](x,a) - T^\star[Q_2](x,a)| &\underset{(a)}{=} \left|\gamma\sum_{y\in\mathcal{X}} p(y|x,a)[\max_{b\in\mathcal{A}} Q_1(y,b) - \max_{b\in\mathcal{A}} Q_2(y,b)]\right|,\\
&\underset{(b)}{\leq} \gamma\sum_{y\in\mathcal{X}} p(y|x,a)\left|\max_{b\in\mathcal{A}} Q_1(y,b) - \max_{b\in\mathcal{A}} Q_2(y,b)\right|,\\
&\underset{(c)}{\leq} \gamma\max_{y\in\mathcal{X}}\left|\max_{b\in\mathcal{A}} Q_1(y,b) - \max_{b\in\mathcal{A}} Q_2(y,b)\right|,\\
&\underset{(d)}{\leq} \gamma\max_{y\in\mathcal{X}}\max_{b\in\mathcal{A}}\left|Q_1(y,b) - Q_2(y,b)\right|,\\
&\underset{(e)}{\leq} \gamma\max_{(y,b)\in\mathcal{X}\times\mathcal{A}}\left|Q_1(y,b) - Q_2(y,b)\right|,\\
&\underset{(f)}{\leq} \gamma\|Q_1 - Q_2\|_\infty,
\end{aligned}$$

where (a) is true by definition of $T^\star$, (b) is true because the absolute value of a sum is lower than the sum of the absolute values and because $p(y|x,a) \geq 0$, c is true by the extremal property, d is true because $|\max_{s\in\mathcal{S}} f(s) - \max_{s\in\mathcal{S}} g(s)| \leq \max_{s\in\mathcal{S}} |f(s) - g(s)|$, e is true by definition of the maximum and f by definition of the infinite norm. Finally, as this inequality is true for any $(x,a) \in \mathcal{X}\times\mathcal{A}$, then we have $\|T^\star[Q_1] - T^\star[Q_2]\|_\infty \leq \gamma\|Q_1 - Q_2\|_\infty$. □

Corollary 1. *Relations between V^π and Q^π and between $V^\star$ and $Q^\star$.*
For a given policy π, $\forall x \in \mathcal{X}, V^\pi(x) = \sum_{a\in\mathcal{A}} \pi(a|x)Q^\pi(x,a)$ and $\forall x \in \mathcal{X}, V^\star(x) = \max_{a\in\mathcal{A}} Q^\star(x,a)$.

Proof. Let define $\hat{V} \in \mathbb{R}^{\mathcal{X}}$ such that:

$$\forall x \in \mathcal{X}, \hat{V}(x) = \sum_{a\in\mathcal{A}} \pi(a|x)Q^\pi(x,a).$$

Then:

$$\begin{aligned}
\forall x \in \mathcal{X}, B^\pi[\hat{V}](x) &= \sum_{a\in\mathcal{A}} \pi(a|x)\left[r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a)\hat{V}(y)\right], \\
&\underset{(a)}{=} \sum_{a\in\mathcal{A}} \pi(a|x)\left[r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a)\sum_{b\in\mathcal{A}} \pi(b|y)Q^\pi(y,b)\right], \\
&\underset{(b)}{=} \sum_{a\in\mathcal{A}} \pi(a|x)\left[T^\pi[Q^\pi](x,a)\right], \\
&\underset{(c)}{=} \sum_{a\in\mathcal{A}} \pi(a|x)Q^\pi(x,a), \\
&\underset{(d)}{=} \hat{V}(x),
\end{aligned}$$

where (a) is true by definition of $\hat{V}$, (b) is true by definition of T^π, (c) is true because $Q^\pi = T^\pi Q^\pi$ and (d) is true by definition of $\hat{V}$. Therefore $\hat{V}$ is a fixed-point of B^π and by uniqueness of the fixed point we have $V^\pi = \hat{V}$ which concludes the proof for the relation between V^π and Q^π.

Now let define $\tilde{V} \in \mathbb{R}^\mathcal{X}$ such that:

$$\forall x \in \mathcal{X}, \tilde{V}(x) = \max_{a\in\mathcal{A}} Q^\pi(x,a).$$

Then:

$$\begin{aligned}
\forall x \in \mathcal{X}, B^\star[\tilde{V}](x) &= \max_{a\in\mathcal{A}} \left[r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a)\hat{V}(y)\right], \\
&\underset{(a)}{=} \max_{a\in\mathcal{A}} \left[r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a)\max_{b\in\mathcal{A}} Q^\pi(y,b)\right], \\
&\underset{(b)}{=} \max_{a\in\mathcal{A}} \left[T^\star[Q^\pi](x,a)\right], \\
&\underset{(c)}{=} \max_{a\in\mathcal{A}} Q^\star(x,a), \\
&\underset{(d)}{=} \tilde{V}(x),
\end{aligned}$$

where (a) is true by definition of $\tilde{V}$, (b) is true by definition of $T^\star$, (c) is true because $Q^\star = T^\star Q^\star$ and (d) is true by definition of $\tilde{V}$. Therefore $\tilde{V}$ is a fixed-point of $B^\star$ and by uniqueness of the fixed point we have $V^\star = \tilde{V}$ which concludes the proof for the relation between $V^\star$ and $Q^\star$. □

Theorem 9. *Majorants of the value functions.*
$V^\star$ is a majorant of the set $\{V^\pi\}_{\pi\in\Pi}$, respectively $Q^\star$ is a majorant of the set $\{Q^\pi\}_{\pi\in\Pi}$:

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

Proof. Let $\pi \in \Pi$, we have

$$V^\pi \underset{(a)}{=} B^\pi V^\pi \underset{(b)}{\leq} B^\star V^\pi,$$

where (a) is true because of the Bellman equations and (b) because of the domination of $B^\star$ over B^π. Therefore, we have:

$$V^\pi \leq B^\star V^\pi \underset{(a)}{\leq} (B^\star)^2 V^\pi \underset{(a)}{\leq} (B^\star)^3 V^\pi \ldots \underset{(a)}{\leq} (B^\star)^n V^\pi,$$

where (a) is true by monotonicity of the operator $B^\star$. At the limit, we have:

$$V^\pi \leq \lim_{n\to\infty} (B^\star)^n V^\pi \underset{(a)}{=} V^\star,$$

where (a) is true by the contraction property which concludes the proof for $V^\star$.

Let $\pi \in \Pi$, we have

$$Q^\pi \underset{(a)}{=} T^\pi Q^\pi \underset{(b)}{\leq} T^\star Q^\pi,$$

where (a) is true because of the Bellman equations and (b) because of the domination of $T^\star$ over T^π. Therefore, we have:

$$Q^\pi \leq T^\star Q^\pi \underset{(a)}{\leq} (T^\star)^2 Q^\pi \underset{(a)}{\leq} (T^\star)^3 Q^\pi \dots \underset{(a)}{\leq} (T^\star)^n Q^\pi,$$

where (a) is true by monotonicity of the operator $T^\star$. At the limit, we have:

$$Q^\pi \leq \lim_{n\to\infty} (T^\star)^n Q^\pi \underset{(a)}{=} Q^\star,$$

where (a) is true by the contraction property which concludes the proof for $Q^\star$.

□

Theorem 10. *Existence and construction of optimal stochastic policies.*
A stochastic policy $\pi \in \Pi$ is optimal if and only if $V^\pi = V^\star$. Optimal stochastic policies exist and can be explicitly constructed from $Q^\star$ or Q^π:

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

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

Proof. Before starting the proof, it is important to note the following equivalence:

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

Therefore, if $Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ verifies $\forall x \in \mathcal{X}, \max_{a\in\mathcal{A}} Q(x,a) = \sum_{a\in\mathcal{A}} \pi(a|x) Q(x,a)$, then for $(x,a) \in \mathcal{X} \times \mathcal{A}$:

$$\begin{aligned} T^\pi[Q](x,a) &\underset{(a)}{=} r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a) \sum_{b\in\mathcal{A}} \pi(b|y) Q(y,b), \\ &\underset{(b)}{=} r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a) \max_{b\in\mathcal{A}} Q(y,b), \\ &\underset{(c)}{=} T^\star[Q](x,a), \end{aligned}$$

where (a) is true by definition of T^π, (b) by hypothesis and (c) by definition of $T^\star$. Therefore, we have the following result:

$$\forall x \in \mathcal{X}, \max_{a\in\mathcal{A}} Q(x,a) = \sum_{a\in\mathcal{A}} \pi(a|x) Q(x,a) \implies T^\pi[Q] = T^\star[Q].$$

Now let us prove that $V^\pi = V^\star \implies \forall x \in \mathcal{X}, \quad \sum_{a\in\mathcal{A}} \pi(a|x) Q^\star(x,a) = \max_{a\in A} Q^\star(x,a)$ and also that $V^\pi = V^\star \implies \forall x \in \mathcal{X}, \quad \sum_{a\in\mathcal{A}} \pi(a|x) Q^\pi(x,a) = \max_{a\in A} Q^\pi(x,a)$.
First we show that $V^\pi = V^\star \implies Q^\pi = Q^\star$. Indeed, let $(x,a) \in \mathcal{X} \times \mathcal{A}$:

$$\begin{aligned} Q^\pi(x,a) &\underset{(a)}{=} T^\pi[Q^\pi](x,a), \\ &\underset{(b)}{=} r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a) \sum_{b\in\mathcal{A}} \pi(b|y) Q^\pi(y,b), \\ &\underset{(c)}{=} r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a) V^\pi(y), \\ &\underset{(d)}{=} r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a) V^\star(y), \\ &\underset{(e)}{=} r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a) \max_{b\in\mathcal{A}} Q^\star(y,b), \\ &\underset{(f)}{=} T^\star[Q^\star](x,a), \\ &\underset{(g)}{=} Q^\star(x,a), \end{aligned}$$

where a is true by Bellman equations, (b) is true by definition of T^π, (c) is true because $V^\pi(y) = \sum_{b\in\mathcal{A}} \pi(b|y)Q^\pi(y,b)$, (d) is true by hypothesis $V^\pi = V^\star$, (e) is true because $V^\star(y) = \max_{b\in\mathcal{A}} Q^\star(y,b)$, f is true by definition of $T^\star$ and (g) is true because $T^\star[Q^\star] = Q^\star$. Now, by hypothesis $V^\pi = V^\star$ and corollary 1, we have:

$$\forall x \in \mathcal{X}, V^\pi(x) = \sum_{a\in\mathcal{A}} \pi(a|x)Q^\pi(x,a) = V^\star(x) = \max_{a\in A} Q^\star(x,a).$$

Moreover, we have just shown that $Q^\pi = Q^\star$ under the hypothesis $V^\pi = V^\star$, therefore:

$$\forall x \in \mathcal{X}, \sum_{a\in\mathcal{A}} \pi(a|x)Q^\star(x,a) = \max_{a\in A} Q^\star(x,a),$$
$$\forall x \in \mathcal{X}, \sum_{a\in\mathcal{A}} \pi(a|x)Q^\pi(x,a) = \max_{a\in A} Q^\pi(x,a),$$

which concludes the implication proof.

Now let show that $\forall x \in \mathcal{X}, \sum_{a\in\mathcal{A}} \pi(a|x)Q^\star(x,a) = \max_{a\in A} Q^\star(x,a) \implies V^\pi = V^\star$. First let show that $\forall x \in \mathcal{X}, \sum_{a\in\mathcal{A}} \pi(a|x)Q^\star(x,a) = \max_{a\in A} Q^\star(x,a) \implies Q^\pi = Q^\star$. We have:

$$Q^\star \underset{(a)}{=} T^\star[Q^\star] \underset{(b)}{=} T^\pi[Q^\star],$$

where (a) is true because $T^\star[Q^\star] = Q^\star$ and (b) because hypothesis implies $T^\pi[Q^\star] = T^\star[Q^\star]$. Therefore, $Q^\star$ is a fixed point of T^π and by uniqueness this implies $Q^\star = Q^\pi$. Therefore we have:

$$\forall x \in \mathcal{X}, V^\pi(x) \underset{(a)}{=} \sum_{a\in\mathcal{X}} \pi(a|x)Q^\pi(a|x) \underset{(b)}{=} \sum_{a\in\mathcal{X}} \pi(a|x)Q^\star(a|x) \underset{(c)}{=} \max_{a\in A} Q^\star(x,a) \underset{(d)}{=} V^\star(x).$$

where (a) is due to corollary 1, (b) under our hypothesis $Q^\star = Q^\pi$, (c) by hypothesis and (d) because of corollary 1.

Finally, let show that $\forall x \in \mathcal{X}, \sum_{a\in\mathcal{A}} \pi(a|x)Q^\pi(x,a) = \max_{a\in A} Q^\pi(x,a) \implies V^\pi = V^\star$. First let show that $\forall x \in \mathcal{X}, \sum_{a\in\mathcal{A}} \pi(a|x)Q^\pi(x,a) = \max_{a\in A} Q^\pi(x,a) \implies Q^\pi = Q^\star$. We have:

$$Q^\pi \underset{(a)}{=} T^\pi[Q^\pi] \underset{(b)}{=} T^\star[Q^\pi],$$

where (a) is true because $T^\pi[Q^\pi] = Q^\pi$, (b) because hypothesis implies $T^\pi[Q^\pi] = T^\star[Q^\pi]$. Therefore, Q^π is a fixed point of $T^\star$ and by uniqueness this implies $Q^\star = Q^\pi$. Therefore we have:

$$\forall x \in \mathcal{X}, V^\star(x) \underset{(a)}{=} \max_{a\in A} Q^\star(x,a) \underset{(b)}{=} \max_{a\in A} Q^\pi(x,a) \underset{(c)}{=} \sum_{a\in\mathcal{X}} \pi(a|x)Q^\pi(a|x) \underset{(d)}{=} V^\pi(x).$$

where (a) is due to corollary 1, (b) under our hypothesis $Q^\star = Q^\pi$, (c) by hypothesis and (d) because of corollary 1. This concludes the proof. $\square$

Remark 5. *For a given function $Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$, the set of policies:*

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

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 by definition :

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

Finally, the optimality theorem becomes: π is optimal $\iff \pi \in \mathcal{G}(Q^\star)$ or $\pi \in \mathcal{G}(Q^\pi)$.

Theorem 11. *The Greedy Policy Improvement.*
Let $\pi \in \Pi$ be a stochastic policy, then a greedy policy with respect to Q^π noted $\pi_\mathcal{G} \in \mathcal{G}(Q^\pi)$ is such that:

$$Q^\pi \leq Q^{\pi_\mathcal{G}}.$$

In addition:

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

Proof. Let $\pi \in \Pi$ and $\pi_{\mathcal{G}} \in \mathcal{G}(Q^\pi)$, then:

$$Q^\pi \underset{(a)}{=} T^\pi Q^\pi \underset{(b)}{\leq} T^\star Q^\pi \underset{(c)}{=} T^{\pi_{\mathcal{G}}} Q^\pi,$$

where (a) is true by Bellman equation, (b) is true by domination of $T^\star$ over T^π and (c) is true by greediness of $\pi_{\mathcal{G}}$ over Q^π. Now, we have:

$$Q^\pi \underset{(a)}{\leq} T^{\pi_{\mathcal{G}}} Q^\pi \underset{(a)}{\leq} (T^{\pi_{\mathcal{G}}})^2 Q^\pi \underset{(a)}{\leq} \dots \underset{(a)}{\leq} (T^{\pi_{\mathcal{G}}})^n Q^\pi,$$

where (a) is true by monotonicity of the operator $T^{\pi_{\mathcal{G}}}$. Therefore at the limit, we have:

$$Q^\pi \leq \lim_{n\to\infty} (T^{\pi_{\mathcal{G}}})^n Q^\pi \underset{(a)}{=} Q^{\pi_{\mathcal{G}}},$$

where (a) is true by contraction of the operator $T^{\pi_{\mathcal{G}}}$ and the Banach fixed-point theorem. Finally, we just need to prove:

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

Let start by hypothesizing that $Q^\pi = Q^{\pi_{\mathcal{G}}}$. In that case, we have:

$$Q^{\pi_{\mathcal{G}}} \underset{(a)}{=} T^{\pi_{\mathcal{G}}} Q^{\pi_{\mathcal{G}}} \underset{(b)}{\implies} Q^\pi = T^{\pi_{\mathcal{G}}} Q^\pi,$$

where (a) is true by Bellman equation and (b) by hypothesis. In addition, we have:

$$Q^\pi = T^{\pi_{\mathcal{G}}} Q^\pi \underset{(a)}{=} T^\star Q^\pi,$$

where (a) is true by greediness of $\pi_{\mathcal{G}}$ over Q^π. Therefore, Q^π is a fixed point of $T^\star$ and by uniqueness $Q^\pi = Q^\star$. This also implies that $\pi_{\mathcal{G}} \in \mathcal{G}(Q^\star)$ which makes $\pi_{\mathcal{G}}$ optimal.

Now let hypothesize that $Q^\pi = Q^\star$, then we have:

$$Q^\pi = Q^\star \underset{(a)}{\implies} \pi_{\mathcal{G}} \in \mathcal{G}(Q^\star) \underset{(b)}{\implies} V^{\pi_{\mathcal{G}}} = V^\star \underset{(c)}{\implies} Q^\star = Q^{\pi_{\mathcal{G}}} \underset{(d)}{\implies} Q^\pi = Q^{\pi_{\mathcal{G}}},$$

where (a) by greediness of $\pi_{\mathcal{G}}$ over Q^π and by hypothesis over $Q^\star$, (b) by the optimality theorem, (c) was proven in the optimality theorem proofs and (d) by hypothesis. This concludes the proof. □

We have all the tools and results to present the different DP schemes.

Value Iteration: The previous theorems are fundamental because they show how we can explicitly construct an optimal policy from the optimal action-value function $Q^\star$. In addition, as $Q^\star$ is the fixed point of a γ-contraction operator, there is also an explicit iteration scheme to build $Q^\star$ called Value Iteration (VI).

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

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

Moreover, knowing that $\pi \in \mathcal{G}(Q) \implies T^\star Q = T^\pi Q$, one can rewrite Value Iteration:

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

One can easily show that $\lim_{k\to\infty} Q^{\pi_k} = Q^\star$ and even more precisely:

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

under the assumption that we start from $Q_0 = 0$.

Proof. We have for $k \geq 1$:

$$\begin{aligned} Q^\star - Q^{\pi_k} &\underset{(a)}{=} Q^\star - T^{\pi_k} Q_{k-1} + T^{\pi_k} Q_{k-1} - Q^{\pi_k}, \\ &\underset{(b)}{=} Q^\star - T^\star Q_{k-1} + T^{\pi_k} Q_{k-1} - Q^{\pi_k}, \\ &\underset{(c)}{=} T^\star Q^\star - T^\star Q_{k-1} + T^{\pi_k} Q_{k-1} - T^{\pi_k} Q^{\pi_k}, \end{aligned}$$

where (a) is true by adding and subtracting $T^{\pi_k}Q_{k-1}$, (b) is true by greediness of π_k over Q_{k-1}, (c) is true because $Q^\star = T^\star Q^\star$ and $Q^{\pi_k} = T^{\pi_k}Q^{\pi_k}$. Therefore:

$$\begin{aligned}
\forall k \geq 1, \quad \|Q^\star - Q^{\pi_k}\|_\infty &\underset{(a)}{\leq} \|T^\star Q^\star - T^\star Q_{k-1}\|_\infty + \|T^{\pi_k}Q_{k-1} - T^{\pi_k}Q^{\pi_k}\|_\infty, \\
&\underset{(b)}{\leq} \gamma\|Q^\star - Q_{k-1}\|_\infty + \gamma\|Q_{k-1} - Q^{\pi_k}\|_\infty, \\
&\underset{(c)}{=} \gamma\|Q^\star - Q_{k-1}\|_\infty + \gamma\|Q_{k-1} - Q^\star + Q^\star - Q^{\pi_k}\|_\infty, \\
&\underset{(d)}{\leq} \gamma\|Q^\star - Q_{k-1}\|_\infty + \gamma\|Q_{k-1} - Q^\star\|_\infty + \gamma\|Q^\star - Q^{\pi_k}\|_\infty, \\
&\underset{(e)}{=} 2\gamma\|Q^\star - Q_{k-1}\|_\infty + \gamma\|Q^\star - Q^{\pi_k}\|_\infty,
\end{aligned}$$

where (a) is true by triangular inequality, (b) is true because $T^\star$ and T^{π_k} are γ contractions, (c) is true by adding and subtracting $Q^\star$, (d) is true by triangular inequality, (e) is true by rearranging the terms. Therefore:

$$\|Q^\star - Q^{\pi_k}\|_\infty \leq 2\gamma\|Q^\star - Q_{k-1}\|_\infty + \gamma\|Q^\star - Q^{\pi_k}\|_\infty \underset{(a)}{\iff} \|Q^\star - Q^{\pi_k}\|_\infty \leq \frac{2\gamma}{1-\gamma}\|Q^\star - Q_{k-1}\|_\infty,$$

where (a) is true by first subtracting by $\gamma\|Q^\star - Q^{\pi_k}\|_\infty$ and then dividing by $1-\gamma$.

In addition, we have for $k \geq 2$:

$$\|Q^\star - Q_{k-1}\|_\infty \underset{(a)}{=} \|T^\star Q^\star - T^\star Q_{k-2}\|_\infty \underset{(b)}{\leq} \gamma\|Q^\star - Q_{k-2}\|_\infty,$$

where (a) is true because $Q^\star = T^\star Q^\star$ and $Q_k = T^\star Q_{k-1}$ and (b) is true because $T^\star$ is a γ-contraction.

Therefore by induction we have for $k \geq 2$:

$$\|Q^\star - Q_{k-1}\|_\infty \leq \gamma^{k-1}\|Q^\star - Q_0\|_\infty,$$

This inequality holds trivially for $k = 1$ too. Thus,:

$$\begin{aligned}
\forall k \geq 1, \quad \|Q^\star - Q^{\pi_k}\|_\infty &\leq \frac{2\gamma}{1-\gamma}\|Q^\star - Q_{k-1}\|_\infty, \\
&\leq \frac{2\gamma^k}{1-\gamma}\|Q^\star - Q_0\|_\infty.
\end{aligned}$$

Now let use the trivial upper bound over the Q-values:

$$\begin{aligned}
\forall \pi \in \Pi, \forall (x,a) \in \mathcal{X}\times\mathcal{A}, \quad Q^\pi(x,a) &\underset{(a)}{=} \mathbb{E}^\pi_{x,a}\Big[\sum_{t\geq 0} r(X_t, A_t)\Big], \\
&\underset{(b)}{\leq} \sum_{t\geq 0}\gamma^t\|r\|_\infty, \\
&\underset{(c)}{=} \frac{1}{1-\gamma}\|r\|_\infty,
\end{aligned}$$

where (a) is true by definition, (b) is true because the expectation is bounded by the maximum value and (c) because $\sum_{t\geq 0}\gamma^t = \frac{1}{1-\gamma}$. As this bound is true for all policies, it is also true for optimal ones, therefore $\|Q^\star\|_\infty \leq \frac{1}{1-\gamma}\|r\|_\infty$. Finally let assume that we initialise with $Q_0 = 0$, then we obtain:

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

which concludes the proof. □

Policy Iteration: This DP scheme relies mainly on the greedy policy improvement theorem.

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

In PI, the evaluation is a full evaluation $Q_{k+1} = (T^{\pi_{k+1}})^\infty Q_k$ which means that $Q_{k+1} = Q^{\pi_{k+1}}$. Because of the greedy policy improvement theorem, we have:

$$Q_1 = Q^{\pi_1} \leq Q_2 = Q^{\pi_2} \leq Q_3 = Q^{\pi_3} \dots$$

where each step is strictly improving unless optimality is reached. 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: This DP scheme generalizes VI and PI in one common framework.

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

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

1.6 Reinforcement Learning in Practice

In most practical application, we consider the setting where a *local* agent acts in the MDP contrary to Dynamic Programming where full access of the MDP is granted. More precisely, in this setting, an agent observes at time t the state x_t, chooses action a_t, receives the reward $r_t = r(x_t, a_t)$ and transitions 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).

Therefore, practical applications differ heavily from classical Dynamic Programming and presents three major difficulties:

1. 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.

2. Sampling problem: Dynamics are not fully known. This puts an even bigger burden on the exploration process.

3. 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. This will be the subject of the next chapters.

Chapter 2

Deep Q-Network

2.1 Introduction

In this chapter, we will show that the seminal paper introducing Deep Q-Networks [Mnih et al., 2015] is essentially an instantiation of an Approximate Value Iteration (AVI) scheme.

2.2 Approximate Value Iteration

Approximate Value Iteration (AVI) is an approximation of the VI scheme. Indeed, in most practical cases, it is not possible to apply the operator $T^\star$ to a given function $Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$. Therefore, the recurrence step $Q_{k+1} = T^\star Q_k$ is going to be approximated by $Q_{k+1} = \widehat{T^\star Q_k}$, where $\widehat{T^\star Q_k}$ is an approximation of $T^\star Q_k$. The approximation error at step k is noted $\epsilon_k = \widehat{T^\star Q_k} - T^\star Q_k$. With those notations, we can write the general AVI scheme:

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

Let $\pi_k \in \mathcal{G}(Q_k)$, then one can show that [Munos, 2007, Scherrer et al., 2015, Vieillard et al., 2020]:

$$\|Q^\star - Q^{\pi_k}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2}\left((1-\gamma)\sum_{j=1}^{k}\gamma^{k-j}\|\epsilon_j\|_\infty\right) + \frac{2\gamma^k\|r\|_\infty}{(1-\gamma)^2}.$$

Now let us explain how we can get a good approximation $\widehat{T^\star Q_k}$ of $T^\star Q_k$ in practice. Contrary to theoretical RL where we have full knowledge of the MDP, here we have only access to transitions of the type $\xi = (x \in \mathcal{X}, a \in \mathcal{A}, r(x,a), y \sim p(.|x,a))$. However, from this transition ξ and any function $Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$, it is possible to compute the following quantity $t(\xi, Q)$ called target:

$$t(\xi, Q) = r(x,a) + \gamma \max_{b\in\mathcal{A}} Q(y,b),$$

which is an unbiased estimate of $T^\star Q(x,a)$. Since from transitions we can build unbiased estimates of $T^\star Q$, we can apply any regression algorithm to get an approximation $\widehat{T^\star Q}$ that fits $T^\star Q$.

Indeed, at step k of AVI, let say that we have at our disposal a set of transitions $\mathcal{D} = \{\xi_n = (x_n, a_n, r_n = r(x_n,a_n), y_n)\}_{n=1}^N$ and that we construct the set of targets $t(\xi_n, Q_k)$ which are unbiased estimates of $T^\star Q_k(x_n,a_n)$. Then, we can simply define $\widehat{T^\star Q_k}$ as the output of this general L_2-regression algorithm:

$$\widehat{T^\star Q_k} = \arg\min_{f\in\mathcal{F}} \sum_{n=1}^{N} (t(\xi_n, Q_k) - f(x_n,a_n))^2,$$

where $\mathcal{F} \subset \mathcal{R}^{\mathcal{X}\times\mathcal{A}}$ is a functional space. In practice, getting the arg min over a functional space is quite hard but we can do our best and get the opt min which is simply the result obtained by the optimization technique chosen:

$$\widehat{T^\star Q_k} = \operatorname*{opt\,min}_{f\in\mathcal{F}} \sum_{n=1}^{N} (t(\xi_n, Q_k) - f(x_n,a_n))^2.$$

To summarize, here is a general and high-level instantiation of AVI:

$$\textbf{Instantiation of AVI:} \begin{cases} Q_0 \in \mathcal{F} \quad \textbf{Initialisation} \\ \forall k \in \mathbb{N}, \begin{cases} \mathcal{D} = \{\xi_n\}_{n=1}^N \quad \textbf{Data Collection} \\ Q_{k+1} = \operatorname{opt\,min}_{f\in\mathcal{F}} \sum_{n=1}^N (t(\xi_n, Q_k) - f(x_n,a_n))^2 \quad \textbf{Regression} \end{cases} \end{cases}$$

In deep RL, the data collection step is often called the acting and the regression step is called the learning. Now, we are going to see how we can derive DQN and its predecessor Neural Fitted-Q from this high level instantiation of AVI.

2.3 Neural Fitted-Q

In this section, we will present the Neural Fitted-Q algorithm [Riedmiller, 2005] which is the ancestor of DQN. It is simpler than DQN because it consists merely in learning a good policy from a fixed data set $\mathcal{D} = \{\xi_n\}_{n=1}^N$. Therefore the data collection step (the acting) is not existent here. Thus, we will focus on explaining how the regression step is done.

The functional space chosen by Neural Fitted-Q is $\mathcal{F}_\theta = \{Q_\theta | \theta \in \mathbb{R}^{\mathcal{N}}\}$ where θ is the vector of weights (parameters) of a neural network $\mathcal{N}_\theta$ for which Q_θ is the activation of the last layer and $\mathcal{N} \in \mathbb{N}$ is the number of parameters. At high-level Neural Fitted-Q is an instantiation of AVI without data collection and parameterize by a neural network:

$$\textbf{Neural Fitted-Q:} \begin{cases} \mathcal{D} = \{\xi_n\}_{n=1}^N & \textbf{Fixed Data} \\ \theta_0 \in \mathbb{R}^{\mathcal{N}} & \textbf{Initialisation} \\ \forall k \in \mathbb{N}, Q_{\theta_{k+1}} = \operatorname{opt}\min_{Q_\theta \in \mathcal{F}_\theta} \sum_{n=1}^N (t(\xi_n, Q_{\theta_k}) - Q_\theta(x_n, a_n))^2 & \textbf{Regression} \end{cases}$$

The regression step, also called learning, is practically implemented with stochastic gradient descent. In Algorithm 1, we show how this can be done.

```
Algorithm 1: Regression step of Neural Fitted Q.
  Input  : online weights: $\theta \in \mathbb{R}^{\mathcal{N}}$, target weights: $\theta_k \in \mathbb{R}^{\mathcal{N}}$, Dataset of transitions: $\mathcal{D}$, update period: $I \in \mathbb{N}$, learning
           rate: $\alpha \in \mathbb{R}$, batch size: $B \in \mathbb{N}$, discount factor: $\gamma$
  Output: $\theta_{k+1}$
1 for $i \in \{0, \dots, I-1\}$ do
2     Draw uniformly a batch $\mathcal{B} = \{(x_j, a_j, r_j, y_j)\}_{j=1}^B$ from $\mathcal{D}$
3     Compute the targets: $\forall 1 \leq j \leq B, \quad t_j \leftarrow r_j + \gamma \max_{b \in \mathcal{A}} Q_{\theta_k}(y_j, b)$
4     Compute the loss: $\mathcal{L}(\theta) \leftarrow \sum_{j=1}^B \left(Q_\theta(x_j, a_j) - t_j\right)^2$
5     Update the online weights: $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$
6 end
7 $\theta_{k+1} = \theta$
```

One observation is that only two set of weights is necessary to perform the regression described in Algorithm 1. A fixed set of weights θ_k that allows to generate the targets and a set of weights θ that allows to optimize the loss. The fixed set of weights is called the target network and the one used to optimize the loss is the online network. After I steps of stochastic gradient descent, we consider the regression finished and we can output the online weights θ as θ_{k+1}.

Neural Fitted-Q is simply K steps of the previous regression. More precisely, a target network $\theta' \in \mathbb{R}^{\mathcal{N}}$ and an online network $\theta \in \mathbb{R}^{\mathcal{N}}$ are initialized by the neural network architecture and θ' plays the role of θ_k in the regression, then at the end of each regression step the target network is updated by the weights of the online network and so on.

```
Algorithm 2: Neural Fitted-Q.
   Input  : Neural network architecture: $\mathcal{N}_\theta$, dataset of transitions: $\mathcal{D}$, learning steps: $K \in \mathbb{N}$, update period: $I \in \mathbb{N}$,
            learning rate: $\alpha \in \mathbb{R}$, batch size: $B \in \mathbb{N}$, discount factor: $\gamma$
   Output: $\theta_{\mathtt{NFQ}}$
 1 Initialize online weights: $\theta \leftarrow \mathtt{Init}(\mathcal{N}_\theta)$
 2 Initialize target weights: $\theta' \leftarrow \mathtt{Init}(\mathcal{N}_\theta)$
 3 for $k \in \{0, \dots, K-1\}$ do
 4     for $i \in \{0, \dots, I-1\}$ do
 5         Draw uniformly a batch $\mathcal{B} = \{(x_j, a_j, r_j, y_j)\}_{j=1}^B$ from $\mathcal{D}$
 6         Compute the targets: $\forall 1 \leq j \leq B, \quad t_j \leftarrow r_j + \gamma \max_{b \in \mathcal{A}} Q_{\theta'}(y_j, b)$
 7         Compute the loss: $\mathcal{L}(\theta) \leftarrow \sum_{j=1}^B \left(Q_\theta(x_j, a_j) - t_j\right)^2$
 8         Update the online weights: $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$
 9     end
10     Update the target weights: $\theta' \leftarrow \theta$
11 end
12 $\theta_{\mathtt{NFQ}} = \theta$
```

2.4 Deep Q-Network

Deep Q-Network (DQN) is also an instantiation of AVI with a neural network used as functional space for the regression step (learning). Contrary to Neural Fitted-Q where the data set $\mathcal{D}$ is fixed, the data collection (acting) is a continuous process that happens in parallel of the learning. More precisely, the data set $\mathcal{D}$ (also called the replay buffer) is implemented as a Fist-In First-Out (FIFO) queue and the data inserted in $\mathcal{D}$ is collected by interacting with the environment with an ϵ-greedy policy $\pi_{\theta,\epsilon}$:

$$\pi_{\theta,\epsilon} = (1-\epsilon)\pi_\theta + \epsilon\pi_{\mathtt{U}},$$

where $\pi_\theta \in \mathcal{G}(Q_\theta)$ and $\pi_{\mathtt{U}}$ is the uniform policy. In the remaining, we present the two processes, acting and learning, that makes the DQN algorithm. Those two processes share the weights θ of the online network as well as the replay buffer $\mathcal{D}$.

In Algorithm 3, we present the acting process to collect data from the environment.

Algorithm 3: Acting process of DQN.

```
Input  : replay buffer: $\mathcal{D}$, environment: $\mathcal{E}$
Shared : online weights: $\theta \in \mathbb{R}^{\mathcal{N}}$
Output : None
1  $x \leftarrow \mathtt{Init}(\mathcal{E})$
2  while True do
3      $a \leftarrow \mathtt{Sample}(\pi_{\theta,\epsilon}(.|x))$
4      $r(x,a), y \leftarrow \mathtt{Step}(\mathcal{E}, a)$
5      Put $(x, a, r(x,a), y)$ in $\mathcal{D}$
6      $x \leftarrow y$
7  end
```

The learning process is unchanged compared to Neural Fitted-Q and is presented in Algorithm 4 for completeness.

Algorithm 4: Learning process of DQN.

```
Input  : Neural network architecture: $\mathcal{N}_\theta$ learning steps: $K \in \mathbb{N}$, update period: $I \in \mathbb{N}$, learning rate: $\alpha \in \mathbb{R}$, batch
         size: $B \in \mathbb{N}$, discount factor: $\gamma$
Shared : replay buffer: $\mathcal{D}$
Output : $\theta_{\mathtt{DQN}}$
1  Initialize online weights: $\theta \leftarrow \mathtt{Init}(\mathcal{N}_\theta)$
2  Initialize target weights: $\theta' \leftarrow \mathtt{Init}(\mathcal{N}_\theta)$
3  for $k \in \{0, \dots, K-1\}$ do
4      for $i \in \{0, \dots, I-1\}$ do
5          Draw uniformly a batch $\mathcal{B} = \{(x_j, a_j, r_j, y_j)\}_{j=1}^{B}$ from $\mathcal{D}$
6          Compute the targets: $\forall 1 \leq j \leq B, \quad t_j \leftarrow r_j + \gamma \max_{b\in\mathcal{A}} Q_{\theta'}(y_j, b)$
7          Compute the loss: $\mathcal{L}(\theta) \leftarrow \sum_{j=1}^{B} (Q_\theta(x_j, a_j) - t_j)^2$
8          Update the online weights: $\theta \leftarrow \theta - \alpha\nabla_\theta\mathcal{L}(\theta)$
9      end
10     Update the target weights: $\theta' \leftarrow \theta$
11 end
12 $\theta_{\mathtt{DQN}} = \theta$
```

2.5 Details on the DQN Implementation for Atari

In this section, we present some of the important details to understand and reproduce the DQN paper [Mnih et al., 2015]. However, we strongly encourage readers to read the original DQN paper to find out more on the detailed implementation.

2.5.1 Atari Games

The DQN paper uses Atari Games as an environment. More precisely, it uses the Arcade Learning Environment (ALE) [Bellemare et al., 2013] which has been built for the RL community as a test-bed for their algorithms. Here are some important details on the ALE environment:

- Around 50 Atari Games
- Raw observations o_t are composed by a RAM state of 128 bytes (a byte range is $0 - 255$) and a 2-D RGB image i_t of size $[160, 210, 3]$ where each pixel is a byte.
- There are 18 discrete actions such as up, down, left, right, jump, fire and so on.
- The rewards are simply the deltas of the score of the games between each observation.
- The frequency of the games are $60Hz$ which means that each second 60 actions are taken.
- A game can last up to 30 minutes which makes the horizon very long.

This environment has been judged interesting by the RL community for the following reasons. First, it has been designed outside of the community and it has been used by millions of human. Second, it has a huge state space, very long optimisation horizon and no canonical features. Third, there is a great diversity of games ranging from easy to solve like `Pong` or `Boxing` to extremely difficult such as `Montezuma's Revenge` or `Pitfall!` that needs exploration and good representation learning.

2.5.2 Preprocessing

For the DQN algorithm, the state x_t is not the raw observation o_t. There is a step of preprocessing to transform the raw observations into states fed to the algorithm. The preprocessing allows to reduce the computation and the partial-observability. To reduce the computation, a filter function ϕ is applied to the couple of raw observations (o_{t-1}, o_t) to obtain the filtered observation $\phi_t = \phi(o_{t-1}, o_t)$ following those steps in order:

- Only the RGB images are kept: (i_{t-1}, i_t).
- The maximum over each pixels of each image $\max(i_{t-1}, i_t)$ is chosen. We are at size $[160, 210, 3]$.
- The luminance is extracted from the RGB to get an image of $[160, 210, 1]$.
- The image is finally downscale to $[84, 84, 1]$.

To reduce the partial-observability, a state x_t is the stacking of the previous 4 filtered observations: $x_t = [\phi_{t-3}, \phi_{t-2}, \phi_{t-1}, \phi_t]$. The dimensions of x_t are $[84, 84, 4]$. Finally, an action is not taken at every step but every 4 steps and repeated for 4 steps. This allows to reduce the optimization horizon.

2.5.3 Neural Network Architecture

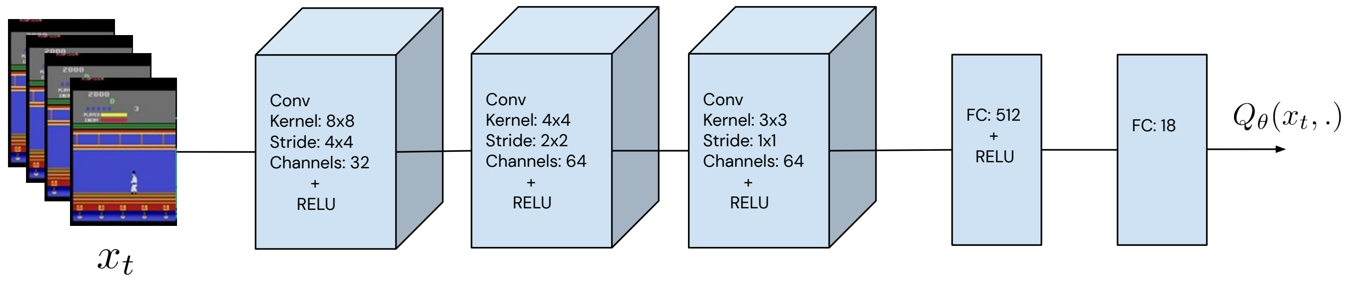

Figure 2.1: DQN's Neural Architecture.

The input x_t is the preprocessed observation as explained in the previous section. The output $Q_\theta(x_t, .) \in \mathbb{R}^{\mathcal{A}}$ is the activation of the last fully-connected layer.

2.5.4 Optimizer

In Deep RL, the optimization of the loss is rarely done through simple SGD but with SGD-based optimization algorithms such as RMSprop [Mukkamala and Hein, 2017] or Adam [Kingma and Ba, 2014].

2.6 Going Further

Now that the reader is familiar with the main concepts relative to DQN, we encourage them to start reading articles of the deep reinforcement learning literature. More particularly, there is an entire family of works that build directly on top of DQN. Those works provide different types of improvements to the vanilla DQN algorithm such as algorithmic improvements, architectural improvements, memory additions, exploration mechanisms and meta-controllers. In the following, we present a list of those works, however this list is far from being exhaustive.

Algorithmic improvements: Double DQN (DDQN) aims to solve the overestimation of the targets [Van Hasselt et al., 2016], Prioritized Experience Replay aims to sample transitions with higher TD-errors [Schaul et al., 2015], Distributional Reinforcement Learning aims to learn not only the expected returns but the entire distributions of returns via a categorical approach [Bellemare et al., 2017] or a quantile approach [Dabney et al., 2018].

Architectural improvements: Dueling architecture allows an agent to have less-biased estimates of unused actions [Wang et al., 2016], Distributed settings allows to collect more diverse data for the learning process [Nair et al., 2015, Horgan et al., 2018, Kapturowski et al., 2018] and can be combine with different prioritization mechanisms.

Memory additions: Working memories such as Long-Short Term Memory (LSTM)[Hochreiter and Schmidhuber, 1997] or Gated-Recurrent Unit (GRU) [Cho et al., 2014, Chung et al., 2014] have been added to the vanilla DQN agent to better handle partial-observability [Hausknecht and Stone, 2015]. Combining LSTM and distributed actors is done in the Recurrent Replay Distributed DQN (R2D2) agent [Kapturowski et al., 2018]. Episodic memory [Badia et al., 2020b] has also been added to help exploration.

Meta controllers: Tuning some hyper-parameters while learning can also be beneficial. This can be done with bandits [Schaul et al., 2019], via meta-gradients [Xu et al., 2018, Xu et al., 2020] or via population-based training [Jaderberg et al., 2017].

Finally, in 2020, 5-year after the DQN publication in Nature, an agent called Agent57 [Badia et al., 2020a] managed to be superhuman on all Atari games by combining most of the improvements cited above.

Chapter 3

Policy Gradient Algorithms

3.1 Introduction

The high-level idea behind policy gradient algorithms is to improve the value function V^{π_θ} of a parameterized policy π_θ via gradient ascent:

$$\theta \leftarrow \theta + \alpha \partial_\theta V^{\pi_\theta}.$$

Therefore, to implement such an idea it is essential to compute the following quantity: $\partial_\theta V^{\pi_\theta}$. The policy gradient theorem [Sutton et al., 2000] and its derivations show how we can compute this quantity.

3.2 Policy Gradient Theorem

Policy gradient algorithms rely on the policy gradient theorem. But before stating the theorem, let us introduce some notations. First, we will consider a general set of policies Π_θ parameterized by the parameters $\theta \in \mathbb{R}^{\mathcal{N}}$:

$$\Pi_\theta = \{\pi_\theta \in \Pi | \theta \in \mathbb{R}^{\mathcal{N}}\}.$$

In deep RL, π_θ is generally a softmax over the last-layer activations of a neural network. Then, for the remaining of the chapter, we make the assumption that the quantities:

$$\begin{aligned} \forall \theta \in \mathbb{R}^{\mathcal{N}}, \forall (x,a) \in \mathcal{X} \times \mathcal{A}, \quad & \partial_\theta \pi_\theta(a|x) \in \mathbb{R}^{\mathcal{N}}, \\ \forall \theta \in \mathbb{R}^{\mathcal{N}}, \forall x \in \mathcal{X}, \quad & \partial_\theta V^{\pi_\theta}(x) \in \mathbb{R}^{\mathcal{N}}, \end{aligned}$$

exist and we define $\partial_\theta \pi \in \mathcal{X} \times \mathcal{A} \to \mathbb{R}^{\mathcal{N}}$ the function that associates (x,a) to $\partial_\theta \pi_\theta(a|x)$ and $\partial_\theta V^{\pi_\theta} \in \mathcal{X} \to \mathbb{R}^{\mathcal{N}}$ the function that associates x to $\partial_\theta V^{\pi_\theta}(x)$.

Theorem 12. *Policy Gradient Theorem.*
Let Π_θ *be a set of policies. Then:*

$$\forall \theta \in \mathbb{R}^{\mathcal{N}}, \forall x \in \mathcal{X}, \quad \partial_\theta V^{\pi_\theta}(x) = \mathbb{E}_x^{\pi_\theta}\left[\sum_{t \geq 0} \gamma^t \sum_{a \in \mathcal{A}} Q^{\pi_\theta}(X_t, a) \partial_\theta \pi_\theta(a|X_t)\right].$$

Proof. Let $\pi_\theta \in \Pi_\theta$ and $x \in \mathcal{X}$, we recall that:

$$V^{\pi_\theta}(x) = \sum_{a \in \mathcal{A}} \pi_\theta(a|x) Q^{\pi_\theta}(x,a).$$

Let θ_i be one dimension of the vector of parameters θ and let derive V^{π_θ} with respect to θ_i:

$$\partial_{\theta_i} V^{\pi_\theta}(x) = \sum_{a \in \mathcal{A}} \partial_{\theta_i} \pi_\theta(a|x) Q^{\pi_\theta}(x,a) + \sum_{a \in \mathcal{A}} \pi_\theta(a|x) \partial_{\theta_i} Q^{\pi_\theta}(x,a).$$

Let's take a closer look at $\partial_{\theta_i} Q^{\pi_\theta}(x,a)$:

$$\begin{aligned} \partial_{\theta_i} Q^{\pi_\theta}(x,a) &= \partial_{\theta_i}(r(x,a) + \gamma \sum_{y \in \mathcal{X}} p(y|x,a) V^{\pi_\theta}(y)), \\ &= \gamma \sum_{y \in \mathcal{X}} p(y|x,a) \partial_{\theta_i} V^{\pi_\theta}(y). \end{aligned}$$

Therefore:

$$\partial_{\theta_i} V^{\pi_\theta}(x) = \sum_{a\in\mathcal{A}} \partial_{\theta_i}\pi_\theta(a|x)Q^{\pi_\theta}(x,a) + \gamma \sum_{a\in\mathcal{A}}\sum_{y\in\mathcal{X}} \pi_\theta(a|x)p(y|x,a)\partial_{\theta_i}V^{\pi_\theta}(y).$$

We recognize a Bellman equation where the quantity $\sum_{a\in\mathcal{A}} \partial_{\theta_i}\pi_\theta(a|x)Q^{\pi_\theta}(x,a)$ plays the role of the reward $r^\pi(x)$, therefore we have by uniqueness (we recall that $V^\pi(x) = \mathbb{E}_x^\pi\left[\sum_{t\geq 0}\gamma^t r^\pi(X_t)\right]$):

$$\partial_{\theta_i} V^{\pi_\theta}(x) = \mathbb{E}_x^{\pi_\theta}\left[\sum_{t\geq 0}\gamma^t \sum_{a\in\mathcal{A}} Q^{\pi_\theta}(X_t,a)\partial_{\theta_i}\pi_\theta(a|X_t)\right].$$

This is true for every dimension θ_i so we conclude that:

$$\partial_{\theta} V^{\pi_\theta}(x) = \mathbb{E}_x^{\pi_\theta}\left[\sum_{t\geq 0}\gamma^t \sum_{a\in\mathcal{A}} Q^{\pi_\theta}(X_t,a)\partial_{\theta}\pi_\theta(a|X_t)\right].$$

□

Other equivalent formulations of the policy gradient theorems are also used in practical algorithms.

Theorem 13. *Other formulations of the policy gradient theorem.*
Value-trick formulation:

$$\forall\theta\in\mathbb{R}^{\mathcal{N}}, \forall x\in\mathcal{X},\quad \partial_{\theta} V^{\pi_\theta}(x) = \mathbb{E}_x^{\pi_\theta}\left[\sum_{t\geq 0}\gamma^t \sum_{a\in\mathcal{A}} (Q^{\pi_\theta}(X_t,a) - V^{\pi_\theta}(X_t))\partial_{\theta}\pi_\theta(a|X_t)\right].$$

Log-trick formulation (under the hypothesis $\forall x\in\mathcal{X}, \operatorname{Supp}\pi_\theta(.|x) = \mathcal{A}$):

$$\forall\theta\in\mathbb{R}^{\mathcal{N}}, \forall x\in\mathcal{X},\quad \partial_{\theta} V^{\pi_\theta}(x) = \mathbb{E}_x^{\pi_\theta}\left[\sum_{t\geq 0}\gamma^t Q^{\pi_\theta}(X_t,A_t)\partial_{\theta}\ln(\pi_\theta(A_t|X_t))\right].$$

Sample-trick formulation (under the hypothesis $\forall x\in\mathcal{X}, \operatorname{Supp}\pi_\theta(.|x) = \mathcal{A}$):

$$\forall\theta\in\mathbb{R}^{\mathcal{N}}, \forall x\in\mathcal{X},\quad \partial_{\theta} V^{\pi_\theta}(x) = \mathbb{E}_x^{\pi_\theta}\left[\sum_{t\geq 0}\gamma^t (r(X_t,A_t) + \gamma V^{\pi_\theta}(X_{t+1}))\partial_{\theta}\ln(\pi_\theta(A_t|X_t))\right].$$

Combination of log and value tricks (under the hypothesis $\forall x\in\mathcal{X}, \operatorname{Supp}\pi_\theta(.|x) = \mathcal{A}$):

$$\forall\theta\in\mathbb{R}^{\mathcal{N}}, \forall x\in\mathcal{X},\quad \partial_{\theta} V^{\pi_\theta}(x) = \mathbb{E}_x^{\pi_\theta}\left[\sum_{t\geq 0}\gamma^t (Q^{\pi_\theta}(X_t,A_t) - V^{\pi_\theta}(X_t))\partial_{\theta}\ln(\pi_\theta(A_t|X_t))\right].$$

Combination of sample and value tricks (under the hypothesis $\forall x\in\mathcal{X}, \operatorname{Supp}\pi_\theta(.|x) = \mathcal{A}$):

$$\forall\theta\in\mathbb{R}^{\mathcal{N}}, \forall x\in\mathcal{X},\quad \partial_{\theta} V^{\pi_\theta}(x) = \mathbb{E}_x^{\pi_\theta}\left[\sum_{t\geq 0}\gamma^t (r(X_t,A_t) + \gamma V^{\pi_\theta}(X_{t+1}) - V^{\pi_\theta}(X_t))\partial_{\theta}\ln(\pi_\theta(A_t|X_t))\right].$$

Proof. Let us start with the value-trick formulation. Noticing that $\sum_{a\in\mathcal{A}}\pi_\theta(a|x) = 1$ then:

$$\forall\theta\in\mathbb{R}^{\mathcal{N}}, \forall(x,a)\in\mathcal{X}\times\mathcal{A}, \sum_{a\in\mathcal{A}}\partial_\theta\pi_\theta(a|x) = \partial_\theta\sum_{a\in\mathcal{A}}\pi_\theta(a|x) = \partial_\theta 1 = 0.$$

Therefore, we also have:

$$\forall\theta\in\mathbb{R}^{\mathcal{N}}, \forall V\in\mathbb{R}^{\mathcal{X}}, \forall(x,a)\in\mathcal{X}\times\mathcal{A}, \sum_{a\in\mathcal{A}} V(x)\partial_\theta\pi_\theta(a|x) = V(x)\sum_{a\in\mathcal{A}}\partial_\theta\pi_\theta(a|x) = 0.$$

Thus we can rewrite the policy gradient theorem (Value-trick or baseline-trick):

$$\forall\theta\in\mathbb{R}^{\mathcal{N}}, \forall V\in\mathbb{R}^{\mathcal{X}}, \forall x\in\mathcal{X},\quad \partial_{\theta} V^{\pi_\theta}(x) = \mathbb{E}_x^{\pi_\theta}\left[\sum_{t\geq 0}\gamma^t \sum_{a\in\mathcal{A}} (Q^{\pi_\theta}(X_t,a) - V(X_t))\partial_{\theta}\pi_\theta(a|X_t)\right].$$

In particular we can choose $V = V^{\pi_\theta}$ and then have:

$$\forall \theta \in \mathbb{R}^{\mathcal{N}}, \forall x \in \mathcal{X}, \quad \partial_\theta V^{\pi_\theta}(x) = \mathbb{E}_x^{\pi_\theta}\left[\sum_{t\geq 0}\gamma^t \sum_{a\in\mathcal{A}}(Q^{\pi_\theta}(X_t, a) - V^{\pi_\theta}(X_t))\partial_\theta \pi_\theta(a|X_t)\right].$$

Now, let us show the the log-trick formulation where we add the hypothesis $\forall x \in \mathcal{X}, \operatorname{Supp}\pi_\theta(.|x) = \mathcal{A}$.

$$\begin{aligned}\sum_{a\in\mathcal{A}} Q^{\pi_\theta}(x,a)\partial_\theta\pi_\theta(a|x) &= \sum_{a\in\mathcal{A}} Q^{\pi_\theta}(x,a)\partial_\theta\pi_\theta(a|x)\frac{\pi_\theta(a|x)}{\pi_\theta(a|x)},\\ &= \sum_{a\in\mathcal{A}} \pi_\theta(a|x)Q^{\pi_\theta}(x,a)\frac{\partial_\theta\pi_\theta(a|x)}{\pi_\theta(a|x)},\\ &= \sum_{a\in\mathcal{A}} \pi_\theta(a|x)Q^{\pi_\theta}(x,a)\partial_\theta\ln(\pi_\theta(a|x)),\end{aligned}$$

Therefore, we have that:

$$\forall \theta \in \mathbb{R}^{\mathcal{N}}, \forall x \in \mathcal{X}, \quad \partial_\theta V^{\pi_\theta}(x) = \mathbb{E}_x^{\pi_\theta}\left[\sum_{t\geq 0}\gamma^t \sum_{a\in\mathcal{A}}\pi_\theta(a|X_t)Q^{\pi_\theta}(X_t, a)\partial_\theta \ln(\pi_\theta(a|X_t))\right].$$

For a given $t \in \mathbb{N}$ and $\forall \theta \in \mathbb{R}^{\mathcal{N}}, \forall x \in \mathcal{X}$, we have:

$$\begin{aligned}\mathbb{E}_x^{\pi_\theta}\left[\sum_{a\in\mathcal{A}}\pi_\theta(a|X_t)Q^{\pi_\theta}(X_t,a)\partial_\theta\ln(\pi_\theta(a|X_t))\right] &= \sum_{y\in\mathcal{X}}\mathbb{P}_x^{\pi_\theta}(X_t = y)\sum_{a\in\mathcal{A}}\pi_\theta(a|y)Q^{\pi_\theta}(y,a)\partial_\theta\ln(\pi_\theta(a|y)),\\ &= \sum_{y\in\mathcal{X}}\sum_{a\in\mathcal{A}}\mathbb{P}_x^{\pi_\theta}(X_t = y)\mathbb{P}_x^{\pi_\theta}(A_t = a|X_t = y)Q^{\pi_\theta}(y,a)\partial_\theta\ln(\pi_\theta(a|y)),\\ &= \sum_{y\in\mathcal{X}}\sum_{a\in\mathcal{A}}\mathbb{P}_x^{\pi_\theta}(X_t = y, A_t = a)Q^{\pi_\theta}(y,a)\partial_\theta\ln(\pi_\theta(a|y)),\\ &= \mathbb{E}_x^{\pi_\theta}\left[Q^{\pi_\theta}(X_t,A_t)\partial_\theta\ln(\pi_\theta(A_t|X_t))\right].\end{aligned}$$

This allows us to conclude that the log-trick formulation is correct. Finally, let us prove the sample-trick formulation by starting from the log-trick formulation:

$$\begin{aligned}\forall \theta \in \mathbb{R}^{\mathcal{N}}, \forall x \in \mathcal{X}, \quad V^{\pi_\theta}(x) &= \mathbb{E}_x^{\pi_\theta}\left[\sum_{t\geq 0}\gamma^t Q^{\pi_\theta}(X_t, A_t)\partial_\theta\ln(\pi_\theta(A_t|X_t))\right],\\ &= \mathbb{E}_x^{\pi_\theta}\left[\sum_{t\geq 0}\gamma^t (r(X_t,A_t) + \gamma\sum_{z\in\mathcal{X}}p(z|X_t,A_t)V^\pi(z))\partial_\theta\ln(\pi_\theta(A_t|X_t))\right],\end{aligned}$$

For a given $t \in \mathbb{N}$ and $\forall \theta \in \mathbb{R}^{\mathcal{N}}, \forall x \in \mathcal{X}$, we have:

$$\begin{aligned}&\mathbb{E}_x^{\pi_\theta}\left[(r(X_t,A_t) + \gamma\sum_{z\in\mathcal{X}}p(z|X_t,A_t)V^\pi(z))\partial_\theta\ln(\pi_\theta(A_t|X_t))\right]\\ &= \sum_{y\in\mathcal{X}}\sum_{a\in\mathcal{A}}\mathbb{P}_x^{\pi_\theta}(X_t = y, A_t = a)(r(y,a) + \gamma\sum_{z\in\mathcal{X}}p(z|y,a)V^\pi(z))\partial_\theta\ln(\pi_\theta(a|y)),\\ &= \sum_{y\in\mathcal{X}}\sum_{a\in\mathcal{A}}\sum_{z\in\mathcal{X}}\mathbb{P}_x^{\pi_\theta}(X_t = y, A_t = a)p(z|y,a)(r(y,a) + \gamma V^\pi(z))\partial_\theta\ln(\pi_\theta(a|y)),\\ &= \sum_{y\in\mathcal{X}}\sum_{a\in\mathcal{A}}\sum_{z\in\mathcal{X}}\mathbb{P}_x^{\pi_\theta}(X_t = y, A_t = a)\mathbb{P}_x^{\pi_\theta}(X_{t+1} = z|X_t = y, A_t = a)(r(y,a) + \gamma V^\pi(z))\partial_\theta\ln(\pi_\theta(a|y)),\\ &= \sum_{y\in\mathcal{X}}\sum_{a\in\mathcal{A}}\sum_{z\in\mathcal{X}}\mathbb{P}_x^{\pi_\theta}(X_t = y, A_t = a, X_{t+1} = z)(r(y,a) + \gamma V^\pi(z))\partial_\theta\ln(\pi_\theta(a|y)),\\ &= \mathbb{E}_x^{\pi_\theta}\left[(r(X_t,A_t) + \gamma V^\pi(X_{t+1}))\partial_\theta\ln(\pi_\theta(A_t|X_t))\right].\end{aligned}$$

This allows us to conclude that the sample-trick formulation is correct. $\square$

3.3 Policy Gradient Algorithms in Practice.

In this section, we are going to present how we can implement one instance of the policy gradient theorem with deep neural networks. In particular, we will focus on the combination of log, value and sample tricks where we only need to estimate the value function. This version is probably the most famous and implemented on-policy policy gradient method and known as advantage actor-critic (A2C) [Mnih et al., 2016].

In practice, we can't compute the full gradient $\partial_\theta V^{\pi_\theta}(x)$ for mainly two reasons:

- we can't perfectly estimate the action-value function Q^{π_θ} and the value function V^{π_θ},
- we can't have access to full trajectories $\tau^{\pi_\theta} = (X_t, A_t, R_t)_{t\geq 0}$ of infinite length.

However, as we have seen for the DQN algorithm, it is possible to estimate Q-values from transitions. In addition, we can have access to a batch $(\hat{\tau}_i^{\pi_\theta})_{i=1}^B$ of B partial trajectories of length H collected by the policy π_θ:

$$(\hat{\tau}_i^{\pi_\theta})_{i=1}^B = \Big((X_{t,i}, A_{t,i}, R_{t,i}, X_{t+1,i})_{t=0}^{H-1}\Big)_{i=1}^B \tag{3.1}$$

where:

$$\begin{gathered} X_{0,i} \sim \rho \\ \forall H-1 \geq t \geq 0, A_{t,i} \sim \pi_\theta(.|X_{t,i}), \\ \forall H-1 \geq t \geq 0, R_{t,i} = r(X_{t,i}, A_{t,i}), \\ \forall H-1 \geq t \geq 0, X_{t+1,i} \sim p(.|X_{t,i}, A_{t,i}). \end{gathered}$$

With those partial trajectories we are going to train two neural networks:

- The policy (actor) network: $\mathcal{N}_\theta$. The policy π_θ is simply going to be a softmax over the last activation layer of $\mathcal{N}_\theta$.
- The value (critic) network: $\mathcal{N}_\phi$. The value V_ϕ is simply going to be the last activation layer of $\mathcal{N}_\phi$.

The actor network loss is:

$$\mathcal{L}_A(\theta, (\hat{\tau}_i^{\pi_\theta})_{i=1}^B, \phi) = -\frac{1}{B}\frac{1}{H}\sum_{i=1}^{B}\sum_{t=0}^{H-1}(R_{t,i} + \gamma V_\phi(X_{t+1,i}) - V_\phi(X_t))\log(\pi_\theta(A_{t,i}|X_{t,i})),$$

and its gradient:

$$\partial_\theta\mathcal{L}_A(\theta, (\hat{\tau}_i^{\pi_\theta})_{i=1}^B, \phi) = -\frac{1}{B}\frac{1}{H}\sum_{i=1}^{B}\sum_{t=0}^{H-1}(R_{t,i} + \gamma V_\phi(X_{t+1,i}) - V_\phi(X_t))\partial_\theta\log(\pi_\theta(A_{t,i}|X_{t,i})),$$

The critic network loss is:

$$\mathcal{L}_C((\hat{\tau}_i^{\pi_\theta})_{i=1}^B, \phi) = \frac{1}{B}\frac{1}{H}\sum_{i=1}^{B}\sum_{t=0}^{H-1}(R_{t,i} + \gamma V_\phi(X_{t+1,i}) - V_\phi(X_t))^2$$

One can remark that if the value network V_ϕ perfectly estimates V^{π_θ} and we have infinite data $(B, H) = (\infty, \infty)$, then:

$$\partial_\theta\mathcal{L}_A(\theta, (\hat{\tau}_i^{\pi_\theta})_{i=1}^B, \phi) = -\mathbb{E}_{X\sim\rho}\left[\partial_\theta V^{\pi_\theta}(X)\right].$$

Therefore, the actor network gradient is a correct approximation of the average gradient over the starting distribution ρ, under the conditions of proper estimation of the value of the policy π_θ and enough data being collected. In Algorithm 5, we present the vanilla version of the advantage actor-critic policy gradient algorithm.

This algorithm has the tendency to collapse quiet fast to bad deterministic policies if the initial policy is not providing enough diverse trajectories. For instance, it is possible to rapidly find a local optimum and never move from there if there is no mechanism to explore new actions in known states. This is the classical trade-off between exploration and exploitation. One way to circumvent this problem is regularization. There are several types of regularization that are used in deep reinforcement learning such as entropy regularization or KL regularization. This would be the subject of a separate chapter. For the moment, we will simply present entropy regularization. In policy-gradient algorithms, entropy regularization is simply introduced in the form of an additional loss function which goal is to maximize entropy of the policy along the collected trajectories:

$$\mathcal{L}_E(\theta, (\hat{\tau}_i^{\pi_\theta})_{i=1}^B) = -\mathcal{H}(\pi^\theta, (\hat{\tau}_i^{\pi_\theta})_{i=1}^B) = \frac{1}{B}\frac{1}{H}\frac{1}{|\mathcal{A}|}\sum_{i=1}^{B}\sum_{t=0}^{H-1}\sum_{a\in A}\pi(a|X_{t,i})\log(\pi(a|X_{t,i})).$$

More complex policy gradient algorithms exist in the literature. In particular, they consider that the data collection (acting) can happen asynchronously from the neural network updates (learning). This has clear advantages as the learning and acting processes can be independent which gives more flexibility and allows parallelisation of the data collection. However, this is often not compatible with having on-policy data. Therefore, the estimations of the gradient and the value function become biased if not done properly. How to build proper estimation of the quantities of interest with off-policy data will be the subject of another chapter.

Algorithm 5: Vanilla advantage actor-critic policy gradient.

```
Input  : Actor neural network architecture: $\mathcal{N}_\theta$, critic neural network architecture $\mathcal{N}_\phi$, learning steps: $K \in \mathbb{N}$,
         learning rate: $\alpha \in \mathbb{R}$, batch size: $B \in \mathbb{N}$, discount factor: $\gamma$
Output: $\theta_{\mathtt{PG}}$
1  Initialize actor weights: $\theta \leftarrow \mathtt{Init}(\mathcal{N}_\theta)$
2  Initialize target weights: $\phi \leftarrow \mathtt{Init}(\mathcal{N}_\phi)$
3  for $k \in \{0, \ldots, K-1\}$ do
4      Collect a batch of on-policy trajectories $(\hat{\tau}_i^{\pi_\theta})_{i=1}^B$
5      Compute the actor loss: $\mathcal{L}_A(\theta, (\hat{\tau}_i^{\pi_\theta})_{i=1}^B, \phi)$
6      Compute the critic loss: $\mathcal{L}_C((\hat{\tau}_i^{\pi_\theta})_{i=1}^B, \phi)$
7      Update actor weights: $\theta \leftarrow \theta - \alpha \partial_\theta \mathcal{L}_A(\theta, (\hat{\tau}_i^{\pi_\theta})_{i=1}^B, \phi)$
8      Update critic weights: $\phi \leftarrow \phi - \alpha \partial_\phi \mathcal{L}_C((\hat{\tau}_i^{\pi_\theta})_{i=1}^B, \phi)$
9  end
10 $\theta_{\mathtt{PG}} = \theta$
```

Algorithm 6: Advantage actor-critic policy gradient.

```
Input  : Actor neural network architecture: $\mathcal{N}_\theta$, critic neural network architecture $\mathcal{N}_\phi$, learning steps: $K \in \mathbb{N}$,
         learning rate: $\alpha \in \mathbb{R}$, batch size: $B \in \mathbb{N}$, discount factor: $\gamma$
Output: $\theta_{\mathtt{PG}}$
1  Initialize actor weights: $\theta \leftarrow \mathtt{Init}(\mathcal{N}_\theta)$
2  Initialize target weights: $\phi \leftarrow \mathtt{Init}(\mathcal{N}_\phi)$
3  for $k \in \{0, \ldots, K-1\}$ do
4      Collect a batch of on-policy trajectories $(\hat{\tau}_i^{\pi_\theta})_{i=1}^B$
5      Compute the actor loss: $\mathcal{L}_A(\theta, (\hat{\tau}_i^{\pi_\theta})_{i=1}^B, \phi)$
6      Compute the critic loss: $\mathcal{L}_C((\hat{\tau}_i^{\pi_\theta})_{i=1}^B, \phi)$
7      Compute the entropy regularization loss: $\mathcal{L}_E(\theta, (\hat{\tau}_i^{\pi_\theta})_{i=1}^B)$
8      Update actor weights: $\theta \leftarrow \theta - \alpha \partial_\theta \left(\mathcal{L}_A(\theta, (\hat{\tau}_i^{\pi_\theta})_{i=1}^B, \phi) + \mathcal{L}_E(\theta, (\hat{\tau}_i^{\pi_\theta})_{i=1}^B)\right)$
9      Update critic weights: $\phi \leftarrow \phi - \alpha \partial_\phi \mathcal{L}_C((\hat{\tau}_i^{\pi_\theta})_{i=1}^B, \phi)$
10 end
11 $\theta_{\mathtt{PG}} = \theta$
```

3.4 Neural Architecture

On Atari games, A2C as a similar architecture as the one of DQN. The only difference is that A2C as two-heads because it has to output two quantities $V_\phi(x_t) \in \mathbb{R}$ and $\pi_\theta(.|x_t) \in \mathbb{R}^{\mathcal{A}}$. The sets of parameters θ and ϕ share the same convolutional torso.

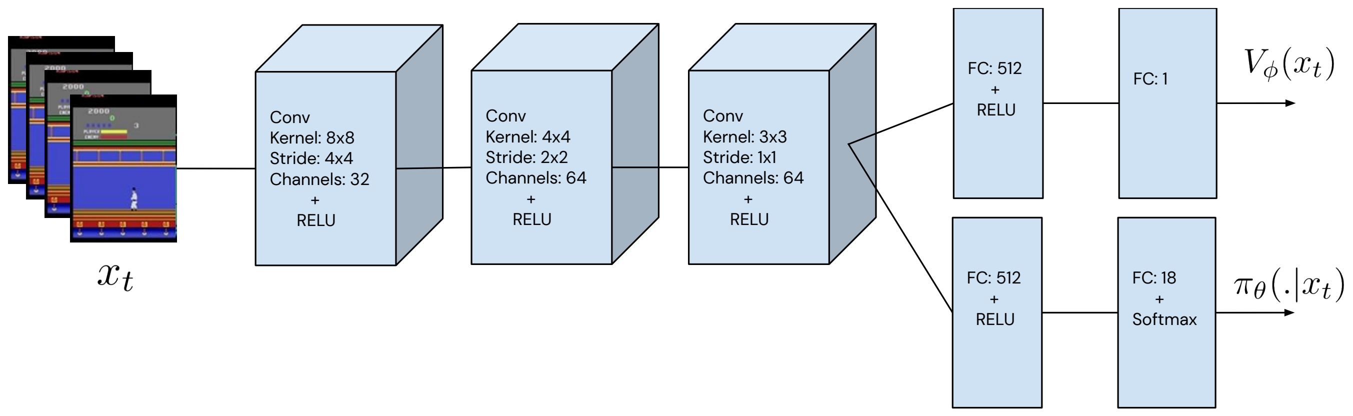

Figure 3.1: A2C Neural Architecture.

Chapter 4

Regularisation in Reinforcement Learning

4.1 Introduction, Background and Notations

In this chapter, we are going to study different regularisation schemes that have proven to be helpful in deep RL. Mainly, when using proper regularisation, the learning is generally more stable and leads sometimes to higher performance at convergence. This could be explained by smoother changes in the data distributions that train the deep neural networks. However, before introducing regularisation in the context of deep RL, we are going to present those regularisation schemes in the context of Dynamic Programming. We choose this approach for two reasons: first because regularisation is in fact a sound approach that benefits from the same theoretical guarantees than classical Dynamic Programming and second because a general theoretical approach on regularisation will help us better understand how all the existing regularized deep RL algorithms can be derived from a unified framework.

To begin, we consider only one type of regularisation. This regularisation is going to penalize policies $\pi \in \Pi$ that derive too much from a baseline policy $\mu \in \Pi$. To do so, we choose the concept of Kulback-Leibler divergence:

$$\forall x \in \mathcal{X}, \texttt{KL}(\pi(.|x)||\mu(.|x)) = \sum_{a \in \mathcal{A}} \pi(a|x) \ln\left(\frac{\pi(a|x)}{\mu(a|x)}\right),$$

where by convention we choose that $0\ln(0) = 0$ and $\ln(0) = -\infty$. Now, the question is how do we introduce this regularisation in the Reinforcement Learning problem to penalise policies that derive too much from the baseline policy μ? The answer is by defining, for each policy $\pi \in \Pi$, a new reward function $r_{\mu,\lambda} \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$, called the regularised reward function, that takes into account the regularisation. More precisely, we define $r_{\mu,\lambda} \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ such that:

$$\forall (x,a) \in \mathcal{X}\times\mathcal{A}, r_{\mu,\lambda}(x,a) = r(x,a) - \lambda \ln\left(\frac{\pi(a|x)}{\mu(a|x)}\right).$$

It is important to note that this reward depends now on the policy π (even if we omit this dependence to not surcharge the notations) as well as the baseline policy μ. Similarly to the state-dependent reward $r^\pi \in \mathbb{R}^{\mathcal{X}}$, we can also define a regularised state-dependent reward $r^\pi_{\mu,\lambda} \in \mathbb{R}^{\mathcal{X}}$ which corresponds to the average regularized reward starting at $x \in \mathcal{X}$ and following policy π:

$$\begin{aligned}\forall x \in \mathcal{X}, r^\pi_{\mu,\lambda}(x) &= \sum_{a \in \mathcal{A}} \pi(a|x)\left[r(x,a) - \lambda \ln\left(\frac{\pi(a|x)}{\mu(a|x)}\right)\right],\\ &= \sum_{a \in \mathcal{A}} \pi(a|x)(r(x,a) - \lambda \sum_{a \in \mathcal{A}} \pi(a|x) \ln\left(\frac{\pi(a|x)}{\mu(a|x)}\right),\\ &= r^\pi(x) - \lambda \texttt{KL}(\pi(.|x)||\mu(.|x)).\end{aligned}$$

Finally, we need to impose a condition on $\pi \in \Pi$ such that our new reward function is bounded. Indeed the Kulback-Leibler divergence $\texttt{KL}(\pi(.|x)||\mu(.|x))$ can be unbounded if $\text{Supp}(\pi(.|x))$ is not included in $\text{Supp}(\mu(.|x))$. Therefore we need to impose that:

$$\forall x \in \mathcal{X}, \quad \text{Supp}(\pi(.|x)) \subset \text{Supp}(\mu(.|x)).$$

In the remaining, we will uniquely consider such policies. The set of policies that verify the previous condition is noted Π_μ and we have:

$$\forall \pi \in \Pi_\mu, \quad \|r_{\mu,\lambda}\| < \infty \text{ and } \|r^\pi_{\mu,\lambda}\| < \infty.$$

Now that we have our new reward function, it is possible to define the concept of discounted cumulative returns in the regularised case and try to find algorithms to optimise these quantities.

- **Regularised Discounted Cumulative returns**: We define the regularised state discounted cumulative return as the discounted sum of regularised rewards over the state-trajectory process:

$$Z^{\mu,\lambda}_{\mathcal{X}}((X_t)_{t\in\mathbb{N}}) = \sum_{t\geq 0}\gamma^t r^\pi_{\mu,\lambda}(X_t),$$

  We also define the regularised state-action discounted cumulative return as the discounted sum of regularised rewards over the state-action-trajectory process:

$$Z^{\mu,\lambda}_{\mathcal{X}\times\mathcal{A}}((X_t,A_t)_{t\in\mathbb{N}}) = r(X_0,A_0) + \sum_{t\geq 1}\gamma^t r_{\mu,\lambda}(X_t,A_t).$$

  For the state-action-trajectory process, it is important to note that the first step (X_0, A_0) is not penalised by the regularisation because the action A_0 is not chosen according to π. Therefore we use as reward $r(X_0, A_0)$.

- **Expected cumulative returns**: We define the regularised value (or V-)function and regularised state-action value (or Q-)function as

$$\forall x\in\mathcal{X},\quad V^\pi_{\mu,\lambda}(x) = \mathbb{E}^\pi_x\Big[Z^{\mu,\lambda}_{\mathcal{X}}((X_t)_{t\in\mathbb{N}})\Big] = \mathbb{E}^\pi_x\Big[\sum_{t\geq 0}\gamma^t r^\pi_{\mu,\lambda}(X_t)\Big],$$
$$\forall (x,a)\in\mathcal{X}\times\mathcal{A},\quad Q^\pi_{\mu,\lambda}(x,a) = \mathbb{E}^\pi_{x,a}\Big[Z^{\mu,\lambda}_{\mathcal{X}\times\mathcal{A}}((X_t,A_t)_{t\in\mathbb{N}})\Big] = \mathbb{E}^\pi_{x,a}\Big[r(X_0,A_0) + \sum_{t\geq 1}\gamma^t r_{\mu,\lambda}(X_t,A_t)\Big].$$

  Intuitively, $V^\pi_{\mu,\lambda}(x)$ is the regularised discounted and expected cumulative return starting from state x and then following policy π. Similarly, $Q^\pi_{\mu,\lambda}(x,a)$ is the regularised discounted and expected cumulative return starting from state-action couple (x,a) and then following policy π.

  The quantities $V^\pi_{\mu,\lambda}(x)$ and $Q^\pi_{\mu,\lambda}(x,a)$ are well defined. Indeed $Z^{\mu,\lambda}_{\mathcal{X}}$ can be seen a function from $\mathcal{X}^{\mathbb{N}}$ to $\mathbb{R}$:

$$\forall\omega = (x_n)_{n\in\mathbb{N}}\in\mathcal{X}^{\mathbb{N}}, Z^{\mu,\lambda}_{\mathcal{X}}(\omega) = \sum_{t\geq 0}\gamma^t r^\pi_{\mu,\lambda}(x_t).$$

  Therefore:

$$\begin{aligned}\forall\omega = (x_n)_{n\in\mathbb{N}}\in\mathcal{X}^{\mathbb{N}}, |Z^{\mu,\lambda}_{\mathcal{X}}(\omega)| &= \sum_{t\geq 0}\gamma^t |r^\pi_{\mu,\lambda}(x_t)|,\\ &= \sum_{t\geq 0}\gamma^t\sum_{a\in\mathcal{A}}\pi(a|x)|r^\pi_{\mu,\lambda}(x_t,a)|,\\ &\leq \|r^\pi_{\mu,\lambda}\|_\infty\sum_{t\geq 0}\gamma^t,\\ &= \frac{\|r^\pi_{\mu,\lambda}\|_\infty}{1-\gamma}.\end{aligned}$$

  Thus, $Z^{\mu,\lambda}_{\mathcal{X}}$ is a bounded function which makes $V^\pi_{\mu,\lambda}(x)$ the expectation of a bounded random variable and therefore exists. The same reasoning can be applied to $Q^\pi_{\mu,\lambda}(x,a)$.

4.1.1 The Regularised Bellman Equations

Similarly to the classical Reinforcement Learning framework, in the regularised case, the regularised value function and the regularised state-action value function verify a Bellman-like equation.

Theorem 14. *For a given policy $\pi\in\Pi_\mu$, the value function $V^\pi_{\mu,\lambda}$ verifies:*

$$\begin{aligned}\forall x\in\mathcal{X},\quad V^\pi_{\mu,\lambda}(x) &= \sum_{a\in\mathcal{A}}\pi(a|x)r_{\mu,\lambda}(x,a) + \gamma\sum_{y\in\mathcal{X}}p^\pi_{\mathcal{X}}(y|x)V^\pi_{\mu,\lambda}(y),\\ &= \sum_{a\in\mathcal{A}}\pi(a|x)r_{\mu,\lambda}(x,a) + \gamma\sum_{y\in\mathcal{X}}\sum_{a\in\mathcal{A}}\pi(a|x)p(y|x,a)V^\pi_{\mu,\lambda}(y),\end{aligned}$$

and the action value function $Q^\pi_{\mu,\lambda}$ verifies:

$$\begin{aligned}\forall (x,a)\in\mathcal{X}\times\mathcal{A},\quad Q^\pi_{\mu,\lambda}(x,a) &= r(x,a) + \gamma\sum_{y\in\mathcal{X}}\sum_{b\in\mathcal{A}}p^\pi_{\mathcal{X}\times\mathcal{A}}((y,b)|(x,a))\left[Q^\pi_{\mu,\lambda}(y,b) - \lambda\ln\left(\frac{\pi(b|y)}{\mu(b|y)}\right)\right],\\ &= r(x,a) + \gamma\sum_{y\in\mathcal{X}}\sum_{b\in\mathcal{A}}p(y|x,a)\pi(b|y)\left[Q^\pi_{\mu,\lambda}(y,b) - \lambda\ln\left(\frac{\pi(b|y)}{\mu(b|y)}\right)\right].\end{aligned}$$

Proof. Identical to the proof for the classical Bellman equations. □

4.2 Regularised Bellman Operators and Dynamic Programming

The goal of this section is to show that we can find optimal policies in the regularisation case. More precisely, find an **optimal policy** $\pi^\star \in \Pi_\mu$, such that for all $x \in \mathcal{X}$, and for any policy $\pi \in \Pi_\mu$, $V^{\pi^\star}_{\mu,\lambda}(x) \geq V^{\pi}_{\mu,\lambda}(x)$. Similarly to classical Reinforcement Learning, we can define regularised Bellman operators and prove the counterparts of the 5 theorems that lead to the existence and the construction of optimal policies. The proofs will be identical as we will specifically construct the regularised Bellman operators to share the same properties as the classical operators.

Because we have shown that $V^\pi_{\mu,\lambda}$ and $Q^\pi_{\mu,\lambda}$ verifies Bellman-like equations, we can canonically define the regularised evaluation Bellman operators as operators for which $V^\pi_{\mu,\lambda}$ and $Q^\pi_{\mu,\lambda}$ are fixed-points of these equations. Therefore we have for $\pi \in \Pi_\mu$:

- **Regularised evaluation operator for V-functions**: The regularised evaluation operator $B^\pi_{\mu,\lambda} \in \mathbb{R}^{\mathcal{X}^{\mathbb{R}^{\mathcal{X}}}}$ is defined as follows:

$$\forall V \in \mathbb{R}^{\mathcal{X}}, \forall x \in \mathcal{X}, \quad B^\pi_{\mu,\lambda}[V](x) = \sum_{a\in\mathcal{A}} \pi(a|x) \left[r_{\mu,\lambda}(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a)V(y) \right],$$
$$= \sum_{a\in\mathcal{A}} \pi(a|x) \left[r(x,a) - \lambda \ln\left(\frac{\pi(a|x)}{\mu(a|x)}\right) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a)V(y) \right].$$

- **Regularised evaluation operator for Q-functions**: The regularised evaluation operator $T^\pi_{\mu,\lambda} \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}^{\mathbb{R}^{\mathcal{X}\times\mathcal{A}}}}$ is defined as follows:

$$\forall Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}, \forall (x,a) \in \mathcal{X}\times\mathcal{A}, \quad T^\pi_{\mu,\lambda}[Q](x,a) = r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a) \sum_{b\in\mathcal{A}} \pi(b|y) \left[Q(y,b) - \lambda \ln\left(\frac{\pi(b|y)}{\mu(b|y)}\right)\right].$$

Now the question is how should we define the regularised optimality operators ? The answer to that question is by creating operators that share the same properties as their classical counterparts. For instance the optimality operator $T^\star$ had four important properties that we use in our proofs to show existence of optimal policies in the classical case:

- Monotonicity.
- Domination over the evaluation operator.
- Reachability: for a given $Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$, there exists $\pi \in \Pi$ such that $T^\pi Q = T^\star Q$ ($\pi \in \mathcal{G}(Q)$ verifies this).
- γ-contraction.

Therefore if we are able to build an operator $T^\star_{\mu,\lambda}$ that verifies the same properties, then we could easily prove the existence of optimal policies for the regularised case. The domination and reachability properties of $T^\star$ comes from the fact that the mean operation $\sum_{b\in\mathcal{A}} \pi(b|y)Q(y,b)$ in T^π is replaced by a max operation $\max_{b\in\mathcal{A}} Q(y,b)$ in $T^\star$. We are going to follow the same strategy and replace the mean operation $\sum_{b\in\mathcal{A}} \pi(b|y)\left[Q(y,b) - \lambda\ln\left(\frac{\pi(b|y)}{\mu(b|y)}\right)\right]$ by a max operation $\max_{\delta\in\Delta_A} \sum_{b\in\mathcal{A}} \delta(b)\left[Q(y,b) - \lambda\ln\left(\frac{\delta(b)}{\mu(b|y)}\right)\right]$, where Δ_A is the set of probability distributions over $\mathcal{A}$. We decided to take the maximum over the set of probability distributions over $\mathcal{A}$ and not only over the set of actions because it is a more general concept adapted to stochastic policies. Indeed, in the regularised case, maximum over the set of probability distributions over $\mathcal{A}$ is not always equal to the maximum over the set of actions which is not true for the classical case where you have:

$$\forall Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}, \forall x \in \mathcal{X}, \quad \max_{a\in\mathcal{A}} Q(x,a) = \max_{\delta\in\Delta_A} \sum_{a\in\mathcal{A}} \delta(a)Q(x,a).$$

This explains why we can always find deterministic policies which are optimal in the classical case. This will not be true anymore in the regularised case. Therefore we have:

- **Regularised optimality operator for Q-functions**: The regularised optimality operator $T^\star_{\mu,\lambda} \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}^{\mathbb{R}^{\mathcal{X}\times\mathcal{A}}}}$ is defined as follows:

$$\forall Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}, \forall (x,a) \in \mathcal{X}\times\mathcal{A}, \quad T^\star_{\mu,\lambda}[Q](x,a) = r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a) \max_{\delta\in\Delta_A} \sum_{b\in\mathcal{A}} \delta(b) \left[Q(y,b) - \lambda \ln\left(\frac{\delta(b)}{\mu(b|y)}\right)\right].$$

  The same reasoning can be done for the evaluation operator $B^\star_{\mu,\lambda}$ where we use a max over the probability distributions over actions to replace the expectation.

- **Regularised optimality operator for V-functions**: The regularised optimality operator $B^\star_{\mu,\lambda} \in \mathbb{R}^{\mathcal{X}^{\mathbb{R}^{\mathcal{X}}}}$ is defined as follows:

$$\forall V \in \mathbb{R}^{\mathcal{X}}, \forall x \in \mathcal{X}, \quad B^\star_{\mu,\lambda}[V](x) = \max_{\delta\in\Delta_A} \sum_{a\in\mathcal{A}} \delta(a) \left[r(x,a) - \lambda \ln\left(\frac{\delta(a)}{\mu(a|y)}\right) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a)V(y) \right].$$

Those operators seem quite complex and we are going to introduce some additional notations to not only make them more compact but also to gain some intuition. To each function $V \in \mathbb{R}^{\mathcal{X}}$ we are going to associate a function $Q_V \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ such that:

$$\forall (x,a) \in \mathcal{X}\times\mathcal{A}, Q_V(x,a) = r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a)V(y).$$

Moreover, we have for $\pi \in \Pi$:

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

where (a) is true by definition of Q_V and (b) is true by definition of T^π. One can also remark that $Q_{V^\pi} = Q^\pi$.

In addition, let $\mathcal{S}$ be a finite space, $f \in \mathbb{R}^{\mathcal{S}}$ and $g \in \mathbb{R}^{\mathcal{S}}$, we define the dot product on $\mathcal{S}$ as:

$$\langle f, g\rangle_{\mathcal{S}} = \sum_{s\in\mathcal{S}} f(s)g(s).$$

With those notations, we can rewrite the regularised Bellman operators. Indeed, let $\pi \in \Pi_\mu$:

- **Regularised evaluation operator for V-functions**:

$$\begin{aligned}
\forall V \in \mathbb{R}^{\mathcal{X}}, \forall x \in \mathcal{X}, \quad B^\pi_{\mu,\lambda}[V](x) &= \sum_{a\in\mathcal{A}} \pi(a|x) \left[r_{\mu,\lambda}(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a)V(y) \right], \\
&= \sum_{a\in\mathcal{A}} \pi(a|x) \left[r(x,a) - \lambda \ln\left(\frac{\pi(a|x)}{\mu(a|x)}\right) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a)V(y) \right], \\
&= \sum_{a\in\mathcal{A}} \pi(a|x)Q_V(x,a) - \lambda \mathtt{KL}(\pi(.|x)||\mu(.|x)), \\
&= \langle \pi(.|x), Q_V(x,.)\rangle_{\mathcal{A}} - \lambda \mathtt{KL}(\pi(.|x)||\mu(.|x)), \\
&= T^\pi[V](x) - \lambda \mathtt{KL}(\pi(.|x)||\mu(.|x),
\end{aligned}$$

- **Regularised optimality operator for V-functions**:

$$\begin{aligned}
\forall V \in \mathbb{R}^{\mathcal{X}}, \forall x \in \mathcal{X}, \quad B^\star_{\mu,\lambda}[V](x) &= \max_{\delta\in\Delta_A} \sum_{a\in\mathcal{A}} \delta(a) \left[r(x,a) - \lambda \ln\left(\frac{\delta(a)}{\mu(a|y)}\right) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a)V(y) \right], \\
&= \max_{\delta\in\Delta_A} \left[\langle \delta, Q_V(x,.)\rangle_{\mathcal{A}} - \lambda \mathtt{KL}(\delta||\mu(.|x)) \right],
\end{aligned}$$

- **Regularised evaluation operator for Q-functions**:

$$\begin{aligned}
\forall Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}, \forall (x,a) \in \mathcal{X}\times\mathcal{A}, \quad T^\pi_{\mu,\lambda}[Q](x,a) &= r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a) \sum_{b\in\mathcal{A}} \pi(b|y) \left[Q(y,b) - \lambda \ln\left(\frac{\pi(b|y)}{\mu(b|y)}\right) \right], \\
&= r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a) \left[\langle \pi(.|y), Q(y,.)\rangle_{\mathcal{A}} - \lambda \mathtt{KL}(\pi(.|y)||\mu(.|y)) \right], \\
&= T^\pi[Q](x,a) - \gamma\lambda \sum_{y\in\mathcal{X}} p(y|x,a) \mathtt{KL}(\pi(.|y)||\mu(.|y)).
\end{aligned}$$

- **Regularised optimality operator for Q-functions**:

$$\begin{aligned}
\forall Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}, \forall (x,a) \in \mathcal{X}\times\mathcal{A}, \quad T^\star_{\mu,\lambda}[Q](x,a) &= r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a) \max_{\delta\in\Delta_A} \sum_{b\in\mathcal{A}} \delta(b) \left[Q(y,b) - \lambda \ln\left(\frac{\delta(b)}{\mu(b|y)}\right) \right], \\
&= r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y|x,a) \max_{\delta\in\Delta_A} \left[\langle \delta, Q(y,.)\rangle_{\mathcal{A}} - \lambda \mathtt{KL}(\delta||\mu(.|y)) \right].
\end{aligned}$$

By design, the regularised Bellman operators have the same basic properties than the classical Bellman operators.

Property 3. *Monotonocity.*
The regularised evaluation and optimality operators for V-functions are monotonous. Let $V_1 \in \mathbb{R}^X$ and $V_2 \in \mathbb{R}^X$ such that $V_1 \leq V_2$ and let $\pi \in \Pi_\mu$, then

$$B^\pi_{\mu,\lambda}[V_1] \leq B^\pi_{\mu,\lambda}[V_2],$$
$$B^\star_{\mu,\lambda}[V_1] \leq B^\star_{\mu,\lambda}[V_2].$$

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

Property 4. *Domination of the regularised optimality operators.*
The optimality operator for V-functions dominates the evaluation operator. Let $V \in \mathbb{R}^X$ and $\pi \in \Pi$, then:

$$B^\pi_{\mu,\lambda} V \leq B^\star_{\mu,\lambda} V.$$

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

Here are the 5 fundamental theorems of Dynamical Programming that allows us to show the existence of an optimal policy for the regularised case. They follow exactly the same order and the same program of proofs than the classical case.

Theorem 15. *Fixed-points of the regularised evaluation operators.*
The operator $B^\pi_{\mu,\lambda}$ is a γ-contraction with unique fixed-point $V^\pi_{\mu,\lambda}$. Similarly, the operator $T^\pi_{\mu,\lambda}$ is a γ-contraction with unique fixed-point $Q^\pi_{\mu,\lambda}$:

$$B^\pi_{\mu,\lambda}[V^\pi_{\mu,\lambda}] = V^\pi_{\mu,\lambda},$$
$$T^\pi_{\mu,\lambda}[Q^\pi_{\mu,\lambda}] = Q^\pi_{\mu,\lambda}.$$

Theorem 16. *Fixed-points of the regularised optimality operators.*
The operator $B^\star_{\mu,\lambda}$ is a γ-contraction, its unique fixed-point is noted $V^\star_{\mu,\lambda}$ and called the optimal value function. Similarly, the operator $T^\star_{\mu,\lambda}$ is a γ-contraction, its unique fixed-point is noted $Q^\star_{\mu,\lambda}$ and called the optimal action-value function:

$$B^\star_{\mu,\lambda}[V^\star_{\mu,\lambda}] = V^\star_{\mu,\lambda},$$
$$T^\star_{\mu,\lambda}[Q^\star_{\mu,\lambda}] = Q^\star_{\mu,\lambda}.$$

Corollary 2. *Relations between $V^\pi_{\mu,\lambda}$ and $Q^\pi_{\mu,\lambda}$ and between $V^\star_{\mu,\lambda}$ and $Q^\star_{\mu,\lambda}$.*
For a given policy $\pi \in \Pi_\mu$,

$$\forall x \in \mathcal{X}, V^\pi_{\mu,\lambda}(x) = \sum_{a \in \mathcal{A}} \pi(a|x) Q^\pi_{\mu,\lambda}(x,a) - \lambda \mathit{KL}(\pi(.|x)||\mu(.||x)),$$

and:

$$\forall x \in \mathcal{X}, V^\star_{\mu,\lambda}(x) = \max_{\delta \in \Delta_A} \left[\langle \delta, Q^\star_{\mu,\lambda}(x,.)\rangle_{\mathcal{A}} - \lambda \mathit{KL}(\delta||\mu(.|y)) \right].$$

Theorem 17. *Majorants of the value functions.*
$V^\star_{\mu,\lambda}$ is a majorant of the set $\{V^\pi_{\mu,\lambda}\}_{\pi \in \Pi_\mu}$, respectively $Q^\star_{\mu,\lambda}$ is a majorant of the set $\{Q^\pi_{\mu,\lambda}\}_{\pi \in \Pi_\mu}$:

$$\forall \pi \in \Pi_\mu, V^\pi_{\mu,\lambda} \leq V^\star_{\mu,\lambda},$$
$$\forall \pi \in \Pi_\mu, Q^\pi_{\mu,\lambda} \leq Q^\star_{\mu,\lambda}.$$

Remark 6. *Regularised maximum and argmaximum*
Let $\mathcal{S}$ be a finite state, $q \in \mathbb{R}^{\mathcal{S}}$ and $\eta \in \mathbb{R}^{\mathcal{S}}$, then the following regularised maximum $\max_{\delta \in \Delta_{\mathcal{S}}} [\langle \delta, q\rangle_{\mathcal{S}} - \mathit{KL}(\delta, \eta)]$ is such that:

$$\max_{\delta \in \Delta_{\mathcal{S}}} [\langle \delta, q\rangle_{\mathcal{S}} - \lambda \mathit{KL}(\delta, \eta)] = \lambda \ln \left(\sum_{s \in \mathcal{S}} \eta(s) \exp\left(\frac{q(s)}{\lambda}\right)\right),$$

and the argmaximum $\mathcal{G}_{\eta,\lambda}(q) = \arg\max_{\delta \in \Delta_{\mathcal{S}}} [\langle \delta, q\rangle_{\mathcal{S}} - \mathit{KL}(\delta, \eta)]$ is unique and $\mathcal{G}_{\eta,\lambda}(q) \in \Delta_{\mathcal{S}}$ is such that:

$$\forall s \in \mathcal{S}, \quad \mathcal{G}_{\eta,\lambda}(q)(s) = \frac{\eta(s)\exp\left(\frac{q(s)}{\lambda}\right)}{\sum_{s' \in \mathcal{S}} \eta(s') \exp\left(\frac{q(s')}{\lambda}\right)}.$$

Theorem 18. *Existence and construction of optimal stochastic policies.*
A stochastic policy $\pi \in \Pi_\mu$ is optimal if and only if $V^\pi_{\mu,\lambda} = V^\star_{\mu,\lambda}$. A unique regularised optimal stochastic policy exists and can be explicitly constructed from $Q^\star_{\mu,\lambda}$ or $Q^\pi_{\mu,\lambda}$:

$$\begin{aligned}
V^\pi_{\mu,\lambda} = V^\star_{\mu,\lambda} &\iff \forall x \in \mathcal{X}, \pi(.|x) = \underset{\delta \in \Delta_A}{\arg\max} \left[\langle \delta, Q^\star_{\mu,\lambda}(x,.)\rangle_{\mathcal{A}} - \lambda KL(\delta||\mu(.|y))\right],\\
&\iff \forall (x,a) \in \mathcal{X} \times \mathcal{A}, \pi(a|x) = \frac{\mu(a|x)\exp\left(\frac{Q^\star_{\mu,\lambda}(x,a)}{\lambda}\right)}{\sum_{b\in\mathcal{A}} \mu(b|x)\exp\left(\frac{Q^\star_{\mu,\lambda}(x,b)}{\lambda}\right)}\\
&\iff \forall x \in \mathcal{X}, \pi(.|x) = \underset{\delta \in \Delta_A}{\arg\max} \left[\langle \delta, Q^\pi_{\mu,\lambda}(x,.)\rangle_{\mathcal{A}} - \lambda KL(\delta||\mu(.|y))\right],\\
&\iff \forall (x,a) \in \mathcal{X} \times \mathcal{A}, \pi(a|x) = \frac{\mu(a|x)\exp\left(\frac{Q^\pi_{\mu,\lambda}(x,a)}{\lambda}\right)}{\sum_{b\in\mathcal{A}} \mu(b|x)\exp\left(\frac{Q^\pi_{\mu,\lambda}(x,b)}{\lambda}\right)}.
\end{aligned}$$

Remark 7. *For a given function $Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$, $\mathcal{G}_{\mu,\lambda}(Q)$ is the unique regularised greedy policy:*

$$\forall x \in \mathcal{X}, \mathcal{G}_{\mu,\lambda}(Q)(.|x) = \underset{\delta \in \Delta_A}{\arg\max} \left[\langle \delta, Q(x,.)\rangle_{\mathcal{A}} - \lambda KL(\delta||\mu(.|y))\right],$$

$$\forall (x,a) \in \mathcal{X} \times \mathcal{A}, \mathcal{G}_{\mu,\lambda}(Q)(a|x) = \frac{\mu(a|x)\exp\left(\frac{Q(x,a)}{\lambda}\right)}{\sum_{b\in\mathcal{A}} \mu(b|x)\exp\left(\frac{Q(x,b)}{\lambda}\right)}.$$

In addition, we have:

$$\begin{aligned}
\forall x \in \mathcal{X}, \langle \mathcal{G}_{\mu,\lambda}(Q)(.|x), Q(x,.)\rangle_{\mathcal{A}} - \lambda KL(\mathcal{G}_{\mu,\lambda}(Q)(.|x)||\mu(.|y)) &= \max_{\delta \in \Delta_A} \left[\langle \delta, Q(x,.)\rangle_{\mathcal{A}} - \lambda KL(\delta||\mu(.|y))\right]\\
&= \lambda \ln\left(\sum_{a\in\mathcal{A}} \mu(a|x)\exp\left(\frac{Q(x,a)}{\lambda}\right)\right).
\end{aligned}$$

Therefore by definition :

$$\pi = \mathcal{G}_{\mu,\lambda}(Q) \implies T^\star_{\mu,\lambda}Q = T^\pi_{\mu,\lambda}Q.$$

Finally, the optimality theorem becomes: π is optimal in the regularised case $\iff \pi = \mathcal{G}_{\mu,\lambda}(Q^\star) \iff \pi = \mathcal{G}_{\mu,\lambda}(Q^\pi)$.

Theorem 19. *The Regularised Greedy Policy Improvement.*
Let $\pi \in \Pi_\mu$ be a stochastic policy, then the regularised-greedy policy with respect to $Q^\pi_{\mu,\lambda}$ noted $\pi_{\mathcal{G}_{\mu,\lambda}} = \mathcal{G}_{\mu,\lambda}(Q^\pi_{\mu,\lambda})$ is such that:

$$Q^\pi_{\mu,\lambda} \leq Q^{\pi_{\mathcal{G}_{\mu,\lambda}}}_{\mu,\lambda}.$$

In addition:

$$Q^\pi_{\mu,\lambda} = Q^{\pi_{\mathcal{G}_{\mu,\lambda}}}_{\mu,\lambda} \iff Q^\pi_{\mu,\lambda} = Q^\star_{\mu,\lambda} \implies \pi_{\mathcal{G}_{\mu,\lambda}} \text{ is optimal in the regularised case.}$$

We have all the tools and results to present the different regularised DP schemes.

Regularised Value Iteration: The previous theorems are fundamental because they show how we can explicitly construct a regularised optimal policy from the regularised optimal action-value function $Q^\star_{\mu,\lambda}$. In addition, as $Q^\star_{\mu,\lambda}$ is the fixed point of a γ-contraction operator, there is also an explicit iteration scheme to build $Q^\star_{\mu,\lambda}$ called Regularised Value Iteration (RVI).

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

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

Moreover, knowing that $\pi = \mathcal{G}_{\mu,\lambda}(Q) \implies T^\star_{\mu,\lambda}Q = T^\pi_{\mu,\lambda}Q$, one can rewrite Regularised Value Iteration:

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

One can easily show that $\lim_{k\to\infty} Q^{\pi_k} = Q^\star$ and even more precisely:

$$\forall k \geq 1, \quad \|Q^\star_{\mu,\lambda} - Q^{\pi_k}\|_\infty \leq \frac{2\gamma^k \|Q^\star_{\mu,\lambda}\|_\infty}{(1-\gamma)},$$

under the assumption that we start from $Q_0 = 0$.

Regularised Policy Iteration: This regularised DP scheme relies mainly on the regularised greedy policy improvement theorem 19.

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

In PI, the evaluation is a full evaluation $Q_{k+1} = (T^{\pi_{k+1}}_{\mu,\lambda})^{\infty} Q_k$ which means that $Q_{k+1} = Q^{\pi_{k+1}}_{\mu,\lambda}$. Because of the greedy policy improvement theorem, we have:

$$Q_1 = Q^{\pi_1}_{\mu,\lambda} \leq Q_2 = Q^{\pi_2}_{\mu,\lambda} \leq Q_3 = Q^{\pi_3}_{\mu,\lambda} \cdots$$

where each step is strictly improving unless optimality is reached.

Regularised Modified Policy Iteration: This regularised DP scheme generalizes RVI and RPI in one common framework.

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

Here the evaluation is $Q_{k+1} = (T^{\pi_{k+1}}_{\mu,\lambda})^{m} Q_k$ with $m \in \overline{\mathbb{N}^*}$. This trivially generalizes to RVI with $m = 1$ and to RPI with $m = \infty$. One can show that $\lim_{k\to\infty} Q^{\pi_k} = Q^{\star}_{\mu,\lambda}$.

4.3 Adaptive Regularisation

As we have seen in the previous section, regularised DP schemes converge towards $Q^{\star}_{\mu,\lambda}$ which is not $Q^{\star}$. Therefore, we would like to have an algorithm that still benefits from the regularisation properties and converge towards $Q^{\star}$. One possible avenue to build such an algorithm is to change the baseline policy μ to a policy closer to the online policy π_{k+1} along the learning. Such an algorithm is an adaptive regularised DP scheme. The idea is pretty simple and simply consists in changing μ to π_k at each step k of Regularised Modified Policy Iteration. This algorithm is called Mirror Descent Modified Policy Iteration (MDMPI). It is initialised with $Q_0 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ and $\pi_0 = \mathcal{U}$ where $\mathcal{U}$ is the uniform policy.

$$\textbf{Mirror Descent Modified Policy Iteration:}\begin{cases} \quad Q_0 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}, \quad \pi_0 = \mathcal{U} & \textbf{Initialisation} \\ \forall k \in \mathbb{N}, \quad \pi_{k+1} = \mathcal{G}_{\pi_k,\lambda}(Q_k) & \textbf{Greedy-step} \\ \forall k \in \mathbb{N}, \quad Q_{k+1} = (T^{\pi_{k+1}}_{\pi_k,\lambda})^{m} Q_k & \textbf{Evaluation} \end{cases}$$

For $m = 1$, one can show that [Vieillard et al., 2020]:

$$\forall k \geq 1, \quad \|Q^{\star} - Q^{\pi_k}\|_{\infty} \leq \frac{4\|r\|_{\infty}}{(1-\gamma)^2 k},$$

4.4 Deep RL with Regularization

An important literature exists on deep RL leveraging regularization schemes. Algorithms such as DPP [Azar et al., 2012], SQL [Azar et al., 2011], TRPO [Schulman et al., 2015], PPO [Schulman et al., 2017], MPO [Abdolmaleki et al., 2018], V-MPO [Song et al., 2019], SAC [Haarnoja et al., 2018], Soft-Q Learning [Fox et al., 2015, Haarnoja et al., 2017] have been developed around the principle that constraining the policy to be updated smoothly is beneficial. In this section, we are going to present how we can build a deep RL algorithm from the approximate MDMPI scheme with $m = 1$:

$$\textbf{Approximate Mirror Descent Modified Policy Iteration:}\begin{cases} \quad Q_0 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}, \quad \pi_0 = \mathcal{U} & \textbf{Initialisation} \\ \forall k \in \mathbb{N}, \quad \pi_{k+1} = \mathcal{G}_{\pi_k,\lambda}(Q_k) + \epsilon'_k & \textbf{Greedy-step} \\ \forall k \in \mathbb{N}, \quad Q_{k+1} = T^{\pi_{k+1}}_{\pi_k,\lambda} Q_k + \epsilon_k & \textbf{Evaluation} \end{cases}$$

Actually, the process is extremely similar with how we built DQN. Instead of doing perfect greedy and evaluation steps, we are going to show how we can approximate those steps with well-known Machine Learning algorithms. From chapter 2, we already know that we can approximate the evaluation step with a regression algorithm. Here, we show how we can approximate the greedy-step with a KL minimization (or cross-entropy minimization) algorithm. It is important to remark that contrarily to AVI, the greedy-step at step k can't be executed perfectly because even if it is closed-form it depends on the policy at step $k-1$. Therefore, we need a target policy network to store the policy at step $k-1$ to build the targets to learn the policy at step k that is approximated by an online policy network.

More precisely, let us suppose that, at step k, we have a target state-action value network $Q_{\phi'}$, a target policy network $\pi_{\theta'}$ and a replay buffer D containing transitions previously collected by the agent. Our goal is to train the online policy network π_θ to be as close as possible to our closed-form target $\mathcal{G}_{\pi_{\theta'},\lambda}(Q_{\phi'})$. We use the notion of KL-divergence to build our loss on a batch of states sampled from the replay buffer, $\mathcal{B} = \{(x_j)\}_{j=1}^B$ with $B \in \mathbb{N}^*$ the batch size:

$$\begin{aligned}\mathcal{L}_{\texttt{KL}}(\theta) &= \sum_{j=1}^{B} \texttt{KL}\left(\pi_\theta(.|x_j)||\mathcal{G}_{\pi_{\theta'},\lambda}(Q_{\phi'})(.|x_j)\right),\\ &= \sum_{j=1}^{B} \texttt{KL}\left(\pi_\theta(.|x_j)||\frac{\pi_{\theta'}(.|x_j)\exp\left(\frac{Q_{\phi'}(x,.)}{\lambda}\right)}{\sum_{b\in\mathcal{A}}\pi_{\theta'}(b|x_j)\exp\left(\frac{Q_{\phi'}(x_j,b)}{\lambda}\right)}\right).\end{aligned}$$

Algorithm 7: Greedy Step of Approximate MDMPI.

```
Input  : online policy weights: $\theta \in \mathbb{R}^{\mathcal{N}_\theta}$, target policy weights: $\theta' \in \mathbb{R}^{\mathcal{N}_\theta}$, target state-action value network weights
         $\phi' \in \mathbb{R}^{\mathcal{N}_\phi}$, Dataset of transitions: $\mathcal{D}$, update period: $I \in \mathbb{N}$, learning rate: $\alpha \in \mathbb{R}$, batch size: $B \in \mathbb{N}$,
         discount factor: $\gamma$
Output: $\theta$, $\theta'$
1 for $i \in \{0,\dots,I-1\}$ do
2     Draw uniformly a batch $\mathcal{B} = \{(x_j)\}_{j=1}^B$ from $\mathcal{D}$
3     Compute the targets: $\forall 1 \le j \le B, \quad \pi_j \leftarrow \frac{\pi_{\theta'}(.|x_j)\exp\left(\frac{Q_{\phi'}(x,.)}{\lambda}\right)}{\sum_{b\in\mathcal{A}}\pi_{\theta'}(b|x_j)\exp\left(\frac{Q_{\phi'}(x_j,b)}{\lambda}\right)}$
4     Compute the loss: $\mathcal{L}_{\texttt{KL}}(\theta) \leftarrow \sum_{j=1}^B \texttt{KL}\left(\pi_\theta(.|x_j)||\pi_j\right)$
5     Update the online weights: $\theta \leftarrow \theta - \alpha\nabla_\theta\mathcal{L}_{\texttt{KL}}(\theta)$
6 end
7 $\theta' \leftarrow \theta$
```

The evaluation step can be done the same way that it is done in DQN except that the targets are computed with the operator $T_{\pi_{\theta'},\lambda}^{\mathcal{G}_{\pi_{\theta'},\lambda}(Q_{\phi'})}$ instead of $T^\star$. We provide a complete algorithm of the learning step for completeness:

Algorithm 8: Learning process of Approximate MDMPI.

```
Input  : Policy network architecture: $\mathcal{N}_\theta$, State-action value network architecture: $\mathcal{N}_\phi$, learning steps: $K \in \mathbb{N}$,
         update period: $I \in \mathbb{N}$, learning rate: $\alpha \in \mathbb{R}$, batch size: $B \in \mathbb{N}$, discount factor: $\gamma$
Shared : replay buffer: $\mathcal{D}$
Output : $\theta_{\texttt{MDMPI}}$
1  Initialize online policy weights: $\theta \leftarrow \texttt{Init}(\mathcal{N}_\theta)$
2  Initialize target policy weights: $\theta' \leftarrow \texttt{Init}(\mathcal{N}_\theta)$
3  Initialize online state-action value weights: $\phi \leftarrow \texttt{Init}(\mathcal{N}_\phi)$
4  Initialize target state-action value weights: $\phi' \leftarrow \texttt{Init}(\mathcal{N}_\phi)$
5  for $k \in \{0, \dots, K-1\}$ do
6      for $i \in \{0, \dots, I-1\}$ do
7          Draw uniformly a batch $\mathcal{B} = \{(x_j, a_j, r_j, y_j)\}_{j=1}^B$ from $\mathcal{D}$
8          Compute the state-action value targets: $\forall 1 \leq j \leq B, \quad t_j \leftarrow r_j + \gamma\lambda \ln\left(\sum_{b\in\mathcal{A}} \pi_{\theta'}(b|y_j) \exp\left(\frac{Q_{\phi'}(y_j,b)}{\lambda}\right)\right)$
9          Compute the policy targets: $\forall 1 \leq j \leq B, \quad \pi_j \leftarrow \frac{\pi_{\theta'}(.|x_j)\exp\left(\frac{Q_{\phi'}(x,.)}{\lambda}\right)}{\sum_{b\in\mathcal{A}} \pi_{\theta'}(b|x_j)\exp\left(\frac{Q_{\phi'}(x_j,b)}{\lambda}\right)}$
10         Compute the state-action value loss: $\mathcal{L}_Q(\phi) \leftarrow \sum_{j=1}^B \left(Q_\phi(x_j, a_j) - t_j\right)^2$
11         Compute the policy loss: $\mathcal{L}_{\texttt{KL}}(\theta) \leftarrow \sum_{j=1}^B \texttt{KL}\left(\pi_\theta(.|x_j)||\pi_j\right)$
12         Update the online state-action weights: $\phi \leftarrow \phi - \alpha\nabla_\phi \mathcal{L}_Q(\phi)$
13         Update the online policy weights: $\theta \leftarrow \theta - \alpha\nabla_\theta \mathcal{L}_{\texttt{KL}}(\theta)$
14     end
15     Update the target state-action value weights: $\phi' \leftarrow \phi$
16     Update the target policy weights: $\theta' \leftarrow \theta$
17 end
18 $\theta_{\texttt{MDMPI}} = \theta$ $\phi_{\texttt{MDMPI}} = \phi$
```

Chapter 5

Off-Policy Reinforcement Learning

5.1 Introduction

Off-policy Reinforcement Learning consists in building RL algorithms able to leverage off-policy data to learn an optimal policy. The simplest off-policy RL algorithm is DQN that is able to leverage data from any policy as we have seen in Chapter 2. One could wonder why do we need to look for other off-policy RL algorithms when we have DQN? The answer is that DQN is a one-step RL algorithm. Indeed, DQN uses only the reward at the current step and the bootstrapped action-value at the next state to compute its targets. This means that the targets are probably very far off from the expected-cumulative return of the current greedy policy, at least at the beginning of the learning. With a one-step algorithm, it may take a long time to back-propagate this information. In that regard, DQN is often referred has an algorithm with high bias. To reduce this bias, an obvious choice is to considered n-step RL algorithm where an n-step discounted cumulative reward is computed before adding the bootstrapped action-value at step $n+1$. As n increases, the less biased the algorithm is going to be. However, the problem with n-step algorithm is that they are by nature on-policy, as the n-step reward depends on the policy that collected it, and their variance increases as n increases. Therefore, is it possible to build an algorithm with all the following desired properties: low-bias, low-variance and off-policiness? In this chapter, our goal is to present new operators and algorithms that tackle the bias-variance trade-off and can handle off-policiness. In particular we will primarily focus on the Retrace [Munos et al., 2016] and V-trace [Espeholt et al., 2018] algorithms.

5.2 Background

We recall that the one-step Bellman operator is T^π:

$$\forall Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}, \quad T^\pi Q = r + \gamma P^\pi_{\mathcal{X}\times\mathcal{A}} Q.$$

To make notations more concise, we note in the remaining of this chapter $P^\pi_{\mathcal{X}\times\mathcal{A}} = P^\pi$, therefore:

$$\forall Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}, \quad T^\pi Q = r + \gamma P^\pi Q.$$

The n-step Bellman operator is defined as $(T^\pi)^n$ with $n \in \mathbb{N}^*$:

$$\forall Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}, \quad (T^\pi)^n Q = \sum_{k=0}^{n-1} (\gamma P^\pi)^k r + (\gamma P^\pi)^n Q.$$

Estimates of $T^\pi Q$ are in expectation far off Q^π but with low-variance and estimates of $(T^\pi)^n Q$ are closer in expectation to Q^π but with high variance. That is why the geometric-averaged Bellman operator T^π_λ, with parameter $\lambda \in [0,1)$ has been introduced [Sutton, 1988, Geist et al., 2014]:

$$\forall \lambda \in [0,1), \quad \forall Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}, \quad T^\pi_\lambda Q = (1-\lambda) \sum_{n\geq 0} \lambda^n (T^\pi)^{n+1} Q.$$

Theorem 20. *For all $\lambda \in [0,1)$ and all $Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$, the quantity $T^\pi_\lambda Q = (1-\lambda)\sum_{n\geq 0} \lambda^n (T^\pi)^{n+1} Q$ is well defined and such that:*

$$\forall \lambda \in [0,1), \quad \forall Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}, \quad T^\pi_\lambda Q = Q + (I - \lambda\gamma P^\pi)^{-1} (T^\pi Q - Q).$$

Proof. For a given $\lambda \in [0,1)$ and a given $Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$, let us look at the quantity:

$$(1-\lambda) \sum_{n=0}^{N} \lambda^n (T^\pi)^{n+1} Q,$$

with $N \in \mathbb{N}^*$. We have that:

$$\begin{aligned}
(1-\lambda)\sum_{n=0}^{N}\lambda^n (T^\pi)^{n+1}Q &= \sum_{n=0}^{N}\lambda^n (T^\pi)^{n+1}Q - \sum_{n=0}^{N}\lambda^{n+1}(T^\pi)^{n+1}Q,\\
&= Q + \sum_{n=0}^{N}\lambda^n (T^\pi)^{n+1}Q - Q - \sum_{n=0}^{N}\lambda^{n+1}(T^\pi)^{n+1}Q,\\
&= Q + \sum_{n=0}^{N}\lambda^n (T^\pi)^{n+1}Q - \sum_{n=0}^{N+1}\lambda^{n}(T^\pi)^{n}Q,\\
&= Q + \sum_{n=0}^{N}\lambda^n (T^\pi)^{n+1}Q - \sum_{n=0}^{N}\lambda^{n}(T^\pi)^{n}Q - \lambda^{N+1}(T^\pi)^{N+1}Q,\\
&= Q + \sum_{n=0}^{N}\lambda^n (T^\pi)^{n}T^\pi Q - \sum_{n=0}^{N}\lambda^{n}(T^\pi)^{n}Q - \lambda^{N+1}(T^\pi)^{N+1}Q,\\
&= Q + \sum_{n=0}^{N}(\lambda\gamma P^\pi)^n\left(T^\pi Q - Q\right) - \lambda^{N+1}(T^\pi)^{N+1}Q.
\end{aligned}$$

At the limit when $N \to \infty$, we know that $\lim_{N\to\infty}(T^\pi)^{N+1}Q = Q^\pi$ by the contraction property and $\lim_{N\to\infty}\lambda^{N+1} = 0$ because $\lambda \in [0,1)$, therefore $\lim_{N\to\infty}\lambda^{N+1}(T^\pi)^{N+1}Q = 0$. In addition, $\lim_{N\to\infty}\sum_{n=0}^{N}(\lambda\gamma P^\pi)^n = (I-\lambda\gamma P^\pi)^{-1}$, therefore:

$$\lim_{N\to\infty}(1-\lambda)\sum_{n=0}^{N}\lambda^n (T^\pi)^{n+1}Q = (1-\lambda)\sum_{n\geq 0}\lambda^n (T^\pi)^{n+1}Q = Q + (I-\lambda\gamma P^\pi)^{-1}\left(T^\pi Q - Q\right).$$

□

An interesting point is that even if the quantity $T^\pi_\lambda Q = Q + (I-\lambda\gamma P^\pi)^{-1}(T^\pi Q - Q)$ was not originally defined for $\lambda = 1$, we can still define it as:

$$T^\pi_{\lambda=1}Q = Q + (I-\gamma P^\pi)^{-1}\left(T^\pi Q - Q\right).$$

Theorem 21. *For all $Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$:*

$$T^\pi_{\lambda=1}Q = (T^\pi)^\infty Q = Q^\pi.$$

Proof. Let $Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$:

$$\begin{aligned}
Q^\pi - Q &= T^\pi Q^\pi - T^\pi Q + T^\pi Q - Q,\\
&= \gamma P^\pi\left(Q^\pi - Q\right) + T^\pi Q - Q,\\
(I-\gamma P^\pi)\left(Q^\pi - Q\right) &= T^\pi Q - Q,\\
Q^\pi - Q &= (I-\gamma P^\pi)^{-1}\left(T^\pi Q - Q\right),\\
Q^\pi &= Q + (I-\gamma P^\pi)^{-1}\left(T^\pi Q - Q\right).
\end{aligned}$$

This concludes the proof. □

Therefore the operator T^π_λ ranges from $T^\pi_{\lambda=0} = T^\pi$ to $T^\pi_{\lambda=1} = (T^\pi)^\infty$ and otherwise mixes the different n-step operators with the weights $(1-\lambda)\lambda^n$. Moreover, as the other Bellman operators, T^π_λ is a γ-contraction with fixed point Q^π.

Theorem 22. *For all $\lambda \in [0,1)$, Q^π is the unique fixed point of T^π_λ.*

Proof. Let $\lambda \in [0,1)$, we have:

$$\forall Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}, \quad T^\pi_\lambda Q = Q + (I-\lambda\gamma P^\pi)^{-1}\left(T^\pi Q - Q\right).$$

Therefore, we can easily see that Q^π is a fixed-point:

$$\begin{aligned}
T^\pi_\lambda Q^\pi &= Q^\pi + (I-\lambda\gamma P^\pi)^{-1}\left(T^\pi Q^\pi - Q^\pi\right),\\
&= Q^\pi + (I-\lambda\gamma P^\pi)^{-1}\left(Q^\pi - Q^\pi\right),\\
&= Q^\pi.
\end{aligned}$$

To show the uniqueness, we just need to prove that T^π_λ is a γ-contraction. Let $Q_1 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ and $Q_2 \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$:

$$\begin{aligned}
\|T^\pi_\lambda Q_1 - T^\pi_\lambda Q_2\|_\infty &= \|(1-\lambda)\sum_{n\geq 0}\lambda^n (T^\pi)^{n+1}Q_1 - (1-\lambda)\sum_{n\geq 0}\lambda^n (T^\pi)^{n+1}Q_2\|_\infty, \\
&= \|(1-\lambda)\sum_{n\geq 0}\lambda^n (\gamma P^\pi)^{n+1}(Q_1 - Q_2)\|_\infty, \\
&\leq \|(1-\lambda)\sum_{n\geq 0}\lambda^n (\gamma P^\pi)^{n+1}\|_\infty \|Q_1 - Q_2\|_\infty, \\
&\leq (1-\lambda)\sum_{n\geq 0}\lambda^n \gamma^{n+1}\|(P^\pi)^{n+1}\|_\infty \|Q_1 - Q_2\|_\infty, \\
&= (1-\lambda)\gamma\sum_{n\geq 0}(\lambda\gamma)^n \|Q_1 - Q_2\|_\infty, \\
&= \frac{(1-\lambda)\gamma}{1-\lambda\gamma}\|Q_1 - Q_2\|_\infty, \\
&\leq \gamma\|Q_1 - Q_2\|_\infty,
\end{aligned}$$

where the notation $\|M\|_\infty$ with M a squared matrix of size $|\mathcal{S}|$ corresponds to:

$$\|M\|_\infty = \max_{1\leq i\leq|\mathcal{S}|}\sum_{j=1}^{|\mathcal{S}|}|m_{i,j}|.$$

□

Now, that we are familiar with the equivalence between matricial and Markovian forms, we have that:

$$\forall(x,a)\in\mathcal{X}\times\mathcal{A},\quad T^\pi_\lambda Q(x,a) = Q(x,a) + \mathbb{E}^\pi_{x,a}\left[\sum_{t\geq 0}(\lambda\gamma)^t\left[T^\pi Q(X_t,A_t) - Q(X_t,A_t)\right]\right].$$

In addition, we have the following lemma.

Lemma 1. *For all $Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ and for all $\lambda\in[0,1)$:*

$$\forall(x,a)\in\mathcal{X}\times\mathcal{A},\quad T^\pi_\lambda Q(x,a) = Q(x,a) + \mathbb{E}^\pi_{x,a}\left[\sum_{t\geq 0}(\lambda\gamma)^t\left[r(X_t,A_t) + \gamma\sum_{b\in\mathcal{A}}\pi(b|X_{t+1})Q(X_{t+1},b) - Q(X_t,A_t)\right]\right].$$

Proof. For a given $\lambda\in[0,1)$, a given $Q\in\mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ and for a given $t\in\mathbb{N}$:

$$\begin{aligned}
&\mathbb{E}^\pi_{x,a}\left[T^\pi Q(X_t,A_t)\right] = \sum_{(y,b)\in\mathcal{X}\times\mathcal{A}}\mathbb{P}^\pi_{x,a}((X_t,A_t)=(y,b))\left[r(y,b) + \gamma\sum_{z\in\mathcal{X}}\sum_{c\in\mathcal{A}}p(z|y,b)\pi(c|z)Q(z,c)\right], \\
&= \sum_{(y,b)\in\mathcal{X}\times\mathcal{A}}\mathbb{P}^\pi_{x,a}((X_t,A_t)=(y,b))\left[r(y,b) + \gamma\sum_{z\in\mathcal{X}}\sum_{c\in\mathcal{A}}\mathbb{P}^\pi_{x,a}(X_{t+1}=z|(X_t,A_t)=(y,b))\pi(c|z)Q(z,c)\right], \\
&= \sum_{(y,b)\in\mathcal{X}\times\mathcal{A}}\mathbb{P}^\pi_{x,a}((X_t,A_t)=(y,b))r(y,b) + \gamma\sum_{(y,b)\in\mathcal{X}\times\mathcal{A}}\sum_{z\in\mathcal{X}}\sum_{c\in\mathcal{A}}\mathbb{P}^\pi_{x,a}((X_t,A_t)=(y,b))\mathbb{P}^\pi_{x,a}(X_{t+1}=z|(X_t,A_t)=(y,b))\pi(c|z)Q(z,c), \\
&= \sum_{(y,b)\in\mathcal{X}\times\mathcal{A}}\mathbb{P}^\pi_{x,a}((X_t,A_t)=(y,b))r(y,b) + \gamma\sum_{z\in\mathcal{X}}\sum_{c\in\mathcal{A}}\mathbb{P}^\pi_{x,a}(X_{t+1}=z)\pi(c|z)Q(z,c), \\
&= \mathbb{E}^\pi_{x,a}\left[r(X_t,A_t) + \gamma\sum_{b\in\mathcal{A}}\pi(b|X_{t+1})Q(X_{t+1},b)\right].
\end{aligned}$$

Therefore, we have:

$$\forall(x,a)\in\mathcal{X}\times\mathcal{A},\quad T^\pi_\lambda Q(x,a) = Q(x,a) + \mathbb{E}^\pi_{x,a}\left[\sum_{t\geq 0}(\lambda\gamma)^t\left[r(X_t,A_t) + \gamma\sum_{b\in\mathcal{A}}\pi(b|X_{t+1})Q(X_{t+1},b) - Q(X_t,A_t)\right]\right].$$

□

The operator T^π_λ is an on-policy multi-step operator that mixes different n-step operators. In the next section, we are going to see how we can generalize this operator to make it off-policy.

5.3 Retrace

The Retrace operator $\mathcal{R}_c^{\pi,\mu}$ is a multi-step off-policy operator that is a generalization of T_λ^π. Let $\pi \in \Pi$, $\mu \in \Pi$ and $c \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$, then $\forall Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ and $\forall (x,a) \in \mathcal{X}\times\mathcal{A}$:

$$R_c^{\pi,\mu}Q(x,a) = Q(x,a) + \mathbb{E}_{x,a}^{\mu}\left[\sum_{t\geq 0}\gamma^t \prod_{s=1}^{t} c(X_s,A_s)\left[r(X_t,A_t) + \gamma\sum_{b\in\mathcal{A}}\pi(b|X_{t+1})Q(X_{t+1},b) - Q(X_t,A_t)\right]\right],$$

where we define $\prod_{s=1}^{0} c(X_s,A_s)$ to be 1. The operator $R_c^{\pi,\mu}$ differs from T_λ^π in two aspects. First, the on-policy expectation $\mathbb{E}_{x,a}^{\pi}$ is replaced by an off-policy expectation $\mathbb{E}_{x,a}^{\mu}$ and the coefficients $(\lambda^t)_{t\in\mathbb{N}}$ are replaced by the coefficients $(\prod_{s=1}^{t} c(X_s,A_s))_{t\in\mathbb{N}}$. The natural question to ask is under which conditions on the function c, Q^π becomes the unique fixed-point of the operator $R_c^{\pi,\mu}$. This subject has been actively studied in the off-policy literature:

- The Importance Sampling (IS) operator with $c(X_s,A_s) = \frac{\pi(A_s|X_s)}{\mu(A_s|X_s)}$ guarantees that Q^π is the unique fixed point. However, this operator is known to be high variance.
- The Off Policy operator with $c(X_s,A_s) = \lambda$ avoids the high-variance but does not guarantee that Q^π is the unique fixed point.
- The Tree-back-up operator with $c(X_s,A_s) = \lambda\pi(A_s|X_s)$ avoids the high-variance and guarantees that Q^π is the unique fixed point. However, this operator is too conservative. Indeed, it is not efficient in the near on-policy case (where μ and π are similar) as it unnecessarily cuts the traces, preventing it to make use of full returns.
- The Retrace operator with $c(X_s,A_s) = \lambda\min(1,\frac{\pi(A_s|X_s)}{\mu(A_s|X_s)})$ avoids the high-variance and guarantees that Q^π is the unique fixed point. In addition, it is less conservative than the Tree-back-up operator because $\min(1,\frac{\pi(A_s|X_s)}{\mu(A_s|X_s)}) \geq \pi(A_s|X_s)$.

Theorem 23. *Under the condition that:*

$$\forall (x,a) \in \mathcal{X}\times\mathcal{A},\quad 0 \leq c(x,a) \leq \frac{\pi(a|x)}{\mu(a|x)},$$

where $c(x,a) = 42$ when $\mu(a|x) = 0$, Q^π is the unique fixed point of the operator $R_c^{\pi,\mu}$.

Proof. First defining $c(x,a) = 42$ when $\mu(a|x) = 0$ is of no importance because, under the probability $\mathbb{P}_{x,a}^{\mu}$, the event $\{(X_t,A_t) = (x,a), \mu(a|x) = 0\}$ has measure 0 for all $t \in \mathbb{N}^*$. Now, to prove the result we are first going to show that Q^π is a fixed-point of $R_c^{\pi,\mu}$. Then, under the previously mentioned conditions on c, we are going to show that $R_c^{\pi,\mu}$ is a γ-contraction around Q^π:

$$\forall Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}, \|R_c^{\pi,\mu}Q - Q^\pi\|_\infty \leq \gamma\|Q - Q^\pi\|_\infty,$$

which implies uniqueness of Q^π as a fixed point.

Let us start by showing that Q^π is a fixed-point of $R_c^{\pi,\mu}$. Using the same approach as the one used in Lemma 1, we can show that $\forall Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ and $\forall (x,a) \in \mathcal{X}\times\mathcal{A}$:

$$R_c^{\pi,\mu}Q(x,a) = Q(x,a) + \mathbb{E}_{x,a}^{\mu}\left[\sum_{t\geq 0}\gamma^t \prod_{s=1}^{t} c(X_s,A_s)\left[T^\pi Q(X_t,A_t) - Q(X_t,A_t)\right]\right].$$

Therefore as for all $t \in \mathbb{N}$, $T^\pi Q^\pi(X_t,A_t) - Q^\pi(X_t,A_t) = 0$, then $\forall (x,a) \in \mathcal{X}\times\mathcal{A}$, $\quad R_c^{\pi,\mu}Q^\pi(x,a) = Q^\pi(x,a)$.

Now, let us take a look at the difference $R_c^{\pi,\mu}Q(x,a) - Q^\pi(x,a)$. By combining Lemma 2 and Lemma 3, we have:

$$R_c^{\pi,\mu}Q(x,a) - Q^\pi(x,a) = \mathbb{E}_{x,a}^{\mu}\left[\sum_{t\geq 1}\gamma^t \prod_{s=1}^{t-1} c(X_s,A_s)\left[\sum_{b\in\mathcal{A}}\left(\pi(b|X_t) - c(X_t,b)\mu(b|X_t)\right)\left(Q(X_t,b) - Q^\pi(X_t,b)\right)\right]\right].$$

From the previous expression, our goal will be to show that for each couple $(x,a) \in \mathcal{X}\times\mathcal{A}$ there exists a function $w_{(x,a)} \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ such that:

$$R_c^{\pi,\mu}Q(x,a) - Q^\pi(x,a) = \sum_{(y,b)\in\mathcal{X}\times\mathcal{A}} w_{(x,a)}(y,b)(Q(y,b) - Q^\pi(y,b)),$$

which implies by Cauchy-Schwartz (for a function $f \in \mathbb{R}^{\mathcal{S}}$, the norm 1 is defined as $\|f\|_1 = \sum_{s\in\mathcal{S}}|f(s)|$):

$$|R_c^{\pi,\mu}Q(x,a) - Q^\pi(x,a)| = |\sum_{(y,b)\in\mathcal{X}\times\mathcal{A}} w_{(x,a)}(y,b)(Q(y,b) - Q^\pi(y,b))| \leq \|w_{(x,a)}\|_1\|Q - Q^\pi\|_\infty.$$

Therefore if:

$$\forall (x,a) \in \mathcal{X} \times \mathcal{A}, \quad \|w_{(x,a)}\|_1 \leq \gamma,$$

we will have our result:

$$\|R_c^{\pi,\mu} Q - Q^\pi|_\infty \leq \gamma \|Q - Q^\pi\|_\infty.$$

So let us go back to our expression and define, for $t \in \mathbb{N}^*$, $\omega \in (\mathcal{X} \times \mathcal{A})^{t-1}$ such that $\omega = ((x_s, a_s))_{s=0}^{t-1}$:

$$\begin{aligned}
R_c^{\pi,\mu} Q(x,a) - Q^\pi(x,a) &= \mathbb{E}_{x,a}^\mu \left[\sum_{t \geq 1} \gamma^t \prod_{s=1}^{t-1} c(X_s, A_s) \left[\sum_{b \in \mathcal{A}} \left(\pi(b|X_t) - c(X_t, b)\mu(b|X_t) \right) \left(Q(X_t, b) - Q^\pi(X_t, b) \right) \right] \right], \\
&= \sum_{t \geq 1} \gamma^t \sum_{\omega \in (\mathcal{X} \times \mathcal{A})^{t-1}} \sum_{y \in \mathcal{X}} \mathbb{P}_{x,a}^\mu ((X_s, A_s)_{s=0}^{t-1} = w, X_t = y) \left[\prod_{s=1}^{t-1} c(x_s, a_s) \left[\sum_{b \in \mathcal{A}} \left(\pi(b|y) - c(y,b)\mu(b|y) \right) \left(Q(y,b) - Q^\pi(y,b) \right) \right] \right], \\
&= \sum_{t \geq 1} \gamma^t \sum_{y \in \mathcal{X}} \sum_{b \in \mathcal{A}} \sum_{\omega \in (\mathcal{X} \times \mathcal{A})^{t-1}} \mathbb{P}_{x,a}^\mu ((X_s, A_s)_{s=0}^{t-1} = w, X_t = y) \left[\prod_{s=1}^{t-1} c(x_s, a_s) \left[\pi(b|y) - c(y,b)\mu(b|y) \right] \right] \left(Q(y,b) - Q^\pi(y,b) \right).
\end{aligned}$$

Therefore, for each couple $(x,a) \in \mathcal{X} \times \mathcal{A}$, we define $w_{(x,a)} \in \mathbb{R}^{\mathcal{X} \times \mathcal{A}}$ such that:

$$\forall (y,b) \in \mathcal{X} \times \mathcal{A}, w_{(x,a)}(y,b) = \sum_{t \geq 1} \gamma^t \sum_{\omega \in (\mathcal{X} \times \mathcal{A})^{t-1}} \mathbb{P}_{x,a}^\mu ((X_s, A_s)_{s=0}^{t-1} = w, X_t = y) \left[\prod_{s=1}^{t-1} c(x_s, a_s) \left[\pi(b|y) - c(y,b)\mu(b|y) \right] \right].$$

Now, to conclude, we just need to check that:

$$\forall (x,a) \in \mathcal{X} \times \mathcal{A}, \quad \|w_{(x,a)}\|_1 \leq \gamma.$$

First, it is easy to see that $w_{y,b} \geq 0$ because by hypothesis we have $\pi(b|y) - c(y,b)\mu(b|y) \geq 0$. Therefore, we have that:

$$\|w_{(x,a)}\|_1 = \sum_{y \in \mathcal{X}} \sum_{b \in \mathcal{A}} |w_{x,a}(y,b)| = \sum_{y \in \mathcal{X}} \sum_{b \in \mathcal{A}} w_{x,a}(y,b).$$

Finally:

$$\begin{aligned}
\sum_{y \in \mathcal{X}} \sum_{b \in \mathcal{A}} w_{x,a}(y,b) &= \sum_{t \geq 1} \gamma^t \sum_{y \in \mathcal{X}} \sum_{b \in \mathcal{A}} \sum_{\omega \in (\mathcal{X} \times \mathcal{A})^{t-1}} \mathbb{P}_{x,a}^\mu ((X_s, A_s)_{s=0}^{t-1} = w, X_t = y) \left[\prod_{s=1}^{t-1} c(x_s, a_s) \left[\pi(b|y) - c(y,b)\mu(b|y) \right] \right], \\
&= \sum_{t \geq 1} \gamma^t \sum_{\omega \in (\mathcal{X} \times \mathcal{A})^{t-1}} \sum_{y \in \mathcal{X}} \mathbb{P}_{x,a}^\mu ((X_s, A_s)_{s=0}^{t-1} = w, X_t = y) \left[\prod_{s=1}^{t-1} c(x_s, a_s) \left[\sum_{b \in \mathcal{A}} \pi(b|y) - c(y,b)\mu(b|y) \right] \right], \\
&= \sum_{t \geq 1} \gamma^t \sum_{\omega \in (\mathcal{X} \times \mathcal{A})^{t-1}} \sum_{y \in \mathcal{X}} \mathbb{P}_{x,a}^\mu ((X_s, A_s)_{s=0}^{t-1} = w, X_t = y) \left[\prod_{s=1}^{t-1} c(x_s, a_s) \left[1 - \sum_{b \in \mathcal{A}} c(y,b)\mu(b|y) \right] \right], \\
&= \sum_{t \geq 1} \gamma^t \sum_{\omega \in (\mathcal{X} \times \mathcal{A})^{t-1}} \sum_{y \in \mathcal{X}} \mathbb{P}_{x,a}^\mu ((X_s, A_s)_{s=0}^{t-1} = w, X_t = y) \left[\prod_{s=1}^{t-1} c(x_s, a_s) \sum_{b \in \mathcal{A}} \mathbb{P}_{x,a}^\mu (A_t = b | X_t = y) \left[1 - c(y,b) \right] \right], \\
&= \sum_{t \geq 1} \gamma^t \sum_{\omega \in (\mathcal{X} \times \mathcal{A})^{t-1}} \sum_{y \in \mathcal{X}} \sum_{b \in \mathcal{A}} \mathbb{P}_{x,a}^\mu ((X_s, A_s)_{s=0}^{t-1} = w, X_t = y) \mathbb{P}_{x,a}^\mu (A_t = b | X_t = y) \left[\prod_{s=1}^{t-1} c(x_s, a_s) \left[1 - c(y,b) \right] \right], \\
&= \sum_{t \geq 1} \gamma^t \sum_{\omega \in (\mathcal{X} \times \mathcal{A})^{t-1}} \sum_{y \in \mathcal{X}} \sum_{b \in \mathcal{A}} \mathbb{P}_{x,a}^\mu ((X_s, A_s)_{s=0}^{t-1} = w, X_t = y, A_t = b) \left[\prod_{s=1}^{t-1} c(x_s, a_s) \left[1 - c(y,b) \right] \right], \\
&= \sum_{t \geq 1} \gamma^t \mathbb{E}_{x,a}^\mu \left[\prod_{s=1}^{t-1} c(X_s, A_s) \left[1 - c(X_t, A_t) \right] \right], \\
&= \mathbb{E}_{x,a}^\mu \left[\sum_{t \geq 1} \gamma^t \prod_{s=1}^{t-1} c(X_s, A_s) - \sum_{t \geq 1} \gamma^t \prod_{s=1}^{t} c(X_s, A_s) \right] = \gamma C(x,a) - (C(x,a) - 1),
\end{aligned}$$

where $C(x,a) = \mathbb{E}_{x,a}^\mu \left[\sum_{t \geq 0} \gamma^t \prod_{s=1}^{t-1} c(X_s, A_s) \right]$. Since $C(x,a) \geq 1$, we have that $\sum_{y \in \mathcal{X}} \sum_{b \in \mathcal{A}} w_{x,a}(y,b) \leq \gamma$. This concludes the proof.

□

Lemma 2. *For all $Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$, the difference between $R_c^{\pi,\mu}Q$ and the fixed point Q^π is for all $(x,a) \in \mathcal{X}\times\mathcal{A}$:*

$$R_c^{\pi,\mu}Q(x,a)-Q^\pi(x,a) = \mathbb{E}^\mu_{x,a}\left[\sum_{t\geq 1}\gamma^t\prod_{s=1}^{t-1}c(X_s,A_s)\left[\sum_{b\in\mathcal{A}}\pi(b|X_t)(Q(X_t,b)-Q^\pi(X_t,b))-c(X_t,A_t)(Q(X_t,A_t)-Q^\pi(X_t,A_t))\right]\right].$$

Proof. First, we can remark that we can rewrite $R_c^{\pi,\mu}Q(x,a)$:

$$R_c^{\pi,\mu}Q(x,a) = \mathbb{E}^\mu_{x,a}\left[\sum_{t\geq 0}\gamma^t\prod_{s=1}^{t}c(X_s,A_s)\left[r(X_t,A_t)+\gamma\left(\sum_{b\in\mathcal{A}}\pi(b|X_{t+1})Q(X_{t+1},b)-c(X_{t+1},A_{t+1})Q(X_{t+1},A_{t+1})\right)\right]\right].$$

Because Q^π is the fixed point of $R_c^{\pi,\mu}Q(x,a)$, we have:

$$Q^\pi(x,a) = \mathbb{E}^\mu_{x,a}\left[\sum_{t\geq 0}\gamma^t\prod_{s=1}^{t}c(X_s,A_s)\left[r(X_t,A_t)+\gamma\left(\sum_{b\in\mathcal{A}}\pi(b|X_{t+1})Q^\pi(X_{t+1},b)-c(X_{t+1},A_{t+1})Q^\pi(X_{t+1},A_{t+1})\right)\right]\right],$$

form which we deduce that:

$$\begin{aligned}
&R_c^{\pi,\mu}Q(x,a)-Q^\pi(x,a)\\
&= \mathbb{E}^\mu_{x,a}\left[\sum_{t\geq 0}\gamma^t\prod_{s=1}^{t}c(X_s,A_s)\left[\gamma\left(\sum_{b\in\mathcal{A}}\pi(b|X_{t+1})(Q(X_{t+1})-Q^\pi(X_{t+1},b))-c(X_{t+1},A_{t+1})(Q(X_{t+1},A_{t+1})-Q^\pi(X_{t+1},A_{t+1}))\right)\right]\right],\\
&= \mathbb{E}^\mu_{x,a}\left[\sum_{t\geq 1}\gamma^t\prod_{s=1}^{t-1}c(X_s,A_s)\left[\sum_{b\in\mathcal{A}}\pi(b|X_t)(Q(X_t,b)-Q^\pi(X_t,b))-c(X_t,A_t)(Q(X_t,A_t)-Q^\pi(X_t,A_t))\right]\right].
\end{aligned}$$

□

Lemma 3. *Let $p\in\mathbb{N}$ and define $\Omega^p = (\mathcal{X}\times\mathcal{A})^p$. In addition, for $t\in\mathbb{N}^*$, let consider a function $g\in\mathbb{R}^{\Omega^{t-1}}$ and $f\in\mathbb{R}^{\mathcal{X}\times\mathcal{A}}$. Then for a policy $\mu\in\Pi$ and state-couple (x,a), we have:*

$$\mathbb{E}^\mu_{x,a}\left[g\left((X_s,A_s)_{s=0}^{t-1}\right)f(X_t,A_t)\right] = \mathbb{E}^\mu_{x,a}\left[g\left((X_s,A_s)_{s=0}^{t-1}\right)\sum_{b\in\mathcal{A}}\mu(b|X_t)f(X_t,b))\right].$$

Proof.

$$\begin{aligned}
\mathbb{E}^\mu_{x,a}\left[g\left((X_s,A_s)_{s=0}^{t-1}\right)f(X_s,A_s)\right] &= \sum_{\omega\in\Omega^{t-1}}\sum_{y\in\mathcal{X}}\sum_{b\in\mathcal{A}}\mathbb{P}^\mu_{x,a}((X_s,A_s)_{s=0}^{t-1}=w, X_t=y, A_t=b)g(\omega)f(y,b),\\
&= \sum_{\omega\in\Omega^{t-1}}\sum_{y\in\mathcal{X}}\sum_{b\in\mathcal{A}}\mathbb{P}^\mu_{x,a}((X_s,A_s)_{s=0}^{t-1}=w, X_t=y)\mathbb{P}^\mu_{x,a}(A_t=b|(X_s,A_s)_{s=0}^{t-1}=w, X_t=y)g(\omega)f(y,b),\\
&= \sum_{\omega\in\Omega^{t-1}}\sum_{y\in\mathcal{X}}\sum_{b\in\mathcal{A}}\mathbb{P}^\mu_{x,a}((X_s,A_s)_{s=0}^{t-1}=w, X_t=y)\mu(b|y)g(\omega)f(y,b),\\
&= \sum_{\omega\in\Omega^{t-1}}\sum_{y\in\mathcal{X}}\mathbb{P}^\mu_{x,a}((X_s,A_s)_{s=0}^{t-1}=w, X_t=y)g(\omega)\sum_{b\in\mathcal{A}}\mu(b|y)f(y,b),\\
&= \mathbb{E}^\mu_{x,a}\left[g\left((X_s,A_s)_{s=0}^{t-1}\right)\sum_{b\in\mathcal{A}}\mu(b|X_t)f(X_t,b))\right].
\end{aligned}$$

□

5.4 Vtrace

Chapter 6

Exploration in Deep Reinforcement Learning

6.1 Introduction

In the RL paradigm, an agent acts in an environment and must find a policy that optimises a global reward function. However, the agent has only access to transitions which represent a very local knowledge of the reward function and the transition dynamics of the environment. Therefore, to optimise the reward, the agent needs to increase its knowledge about the reward and the dynamics. To do so, the agent can interact with its environment to specifically collect information about the world.

We call exploration a process that actively collects information about the environment. In this chapter, we will focus on pure exploration algorithms and we will not study algorithms that try to achieve the optimal trade-off between exploration and exploitation. The main reason behind this choice is that in Deep RL those pure exploration approaches when combined with classical RL agents have the most success. In addition, we will not present theoretical guarantees but we will provide references to works which do. Finally, the readers should be aware that exploration in deep RL is still an active domain of research and the content of this chapter is suceptible to change in newer versions of the document.

6.2 Count-based Exploration

The high level idea behind count-based exploration methods is to incentivize the RL agent to go to less visited states. To do so, the agent should be able to compute the visitation-count $N(x, t)$ of a state $x \in \mathcal{X}$ after $t \in \mathbb{N}$ interactions with the environment:

$$N(x, t) = \sum_{k=0}^{t} \mathbf{1}_{\{X_k = x\}}.$$

Then one could use the inverse of any increasing monotonic function f of the visitation-count as a reward signal to encourage the agent to go to less visited states:

$$r(X_t, A_t) = \frac{1}{f(N(X_{t+1}, t+1))}.$$

This reward signal is called an intrinsic reward because it is generated by the agent itself and does not depend on any particular predefined task. The classical task-dependent reward is often referred as extrinsic reward in the exploration literature. Generally the function f used is either:

$$f(x) = x \quad \text{or } f(x) = \sqrt{x}.$$

To have a better theoretical understanding of why this intrinsic reward signal is sound we encourage the reader to refer to [Azar et al., 2017]. One important remark is that this intrinsic reward depends on all the previous states that the agent has seen so far and is thus not state dependent.

In practice, because the number of states can be extremely large, an agent may not encounter exactly the same state twice in all its life. Therefore, we prefer to use a kernel (symmetric and definite positive function) $\mathcal{K} : \mathcal{X}^2 \to \mathbb{R}$ instead of the indicator function $\mathbf{1}$ in the computation of the visitation count. More precisely, we obtain the smoothed visitation-count $N_{\mathcal{K}}(x, t)$ of a state $x \in \mathcal{X}$ after $t \in \mathbb{N}$ interactions and for kernel $\mathcal{K}$:

$$N_{\mathcal{K}}(x, t) = \sum_{k=0}^{t} \mathcal{K}(X_k, x).$$

When a distance $d : \mathcal{X}^2 \to \mathbb{R}$ is provided for the state space $\mathcal{X}$, one can define a Gaussian kernel:

$$\forall (x, y) \in \mathcal{X}^2, \mathcal{K}(x, y) = \exp(-d(x, y)^2),$$

or an inverse kernel:

$$\forall(x,y)\in\mathcal{X}^2, \mathcal{K}(x,y)=\frac{\epsilon}{d(x,y)^2+\epsilon},$$

where $\epsilon\in\mathbb{R}_+$. Therefore, the reward $r(X_t,A_t)$ corresponding to the smooth visitation count with an inverse kernel can be written as:

$$r(X_t,A_t)=\frac{1}{\sqrt{N_{\mathcal{K}}(X_{t+1},t+1)}}=\frac{1}{\sqrt{\sum_{k=0}^{t+1}\mathcal{K}(X_k,X_{t+1})}}=\frac{1}{\sqrt{\sum_{k=0}^{t+1}\frac{\epsilon}{d(X_k,X_{t+1})^2+\epsilon}}}.$$

In practice coming up with a smart notion of distance d for a given state space $\mathcal{X}$ is extremely complex. This problem is often framed in deep Reinforcement Learning as the Representation learning problem and consists in finding an embedding function $\phi:\mathcal{X}\to\mathbb{R}^N$ that canonically defines the distance $d_\phi:\mathcal{X}^2\to\mathbb{R}$ as:

$$\forall(x,y)\in\mathcal{X}^2, d_\phi(x,y)=d_{\mathbb{R}^N}(\phi(x),\phi(y)),$$

where $d_{\mathbb{R}^N}$ is the euclidean distance on the vector space $\mathbb{R}^N$. Therefore implementing count-based learning is mainly about learning a good embedding function ϕ which will naturally provide the distance d_ϕ as well as the intrinsic reward to optimize:

$$r(X_t,A_t)=\frac{1}{\sqrt{\sum_{k=0}^{t+1}\frac{\epsilon}{d_\phi(X_k,X_{t+1})^2+\epsilon}}}=\frac{1}{\sqrt{\sum_{k=0}^{t+1}\frac{\epsilon}{d_{\mathbb{R}^N}(\phi(X_k),\phi(X_{t+1}))^2+\epsilon}}}.$$

In the next section, we will show a simple representation learning method called action prediction.

6.3 Counting with Action Prediction and an Episodic Memory

As explained in the previous section, implementing count-based exploration can be reduced to representation learning. This means that we are looking for an embedding function $\phi:\mathcal{X}\to\mathbb{R}^N$ that preserves the *information* of a given state as well as the *geometry* between the states. The embedding function is a projection from a complex and messy state-space to a vector space of way smaller dimensions where the euclidean distance can be interpreted as a good notion of distance in the original state-space. This explanation is hand-wavy and not very formal. We encourage the reader to dive into the representation learning literature for a more rigorous introduction to the domain.

One simple way, used in the exploration literature to learning an embedding ϕ_θ[Pathak et al., 2017, Badia et al., 2020b] is via action prediction. Action prediction is a simple representation learning technique that consists in estimating the action a_t between two consecutive states x_t and x_{t+1}. The states are first embedded with a parameterized embedding $\phi_\theta:\mathcal{X}\to\mathbb{R}^N$ where θ are parameters of a neural network, then $\phi_\theta(x_t)$ and $\phi_\theta(x_{t+1})$ are fed to a predictor $p_\theta:\mathbb{R}^{2N}\to\Delta_{\mathcal{A}}$ that estimates the action a_t. The embedding ϕ_θ and the predictor p_θ are shaped with the log-likelihood loss $\mathcal{L}_{\texttt{AP}}(\theta,t)$:

$$\mathcal{L}_{\texttt{AP}}(\theta,t)=-\ln(p_\theta(a_t|\phi_\theta(x_t),\phi_\theta(x_{t+1}))).$$

The idea behind using the action prediction loss in order to shape the state-representation is to encourage the features of the representation to be the controllable features of the original state. We provide a sketch of the action prediction architecture in Fig. 6.1. Now that we have a trained embedding function ϕ_θ, we just have to keep in memory the past previous states

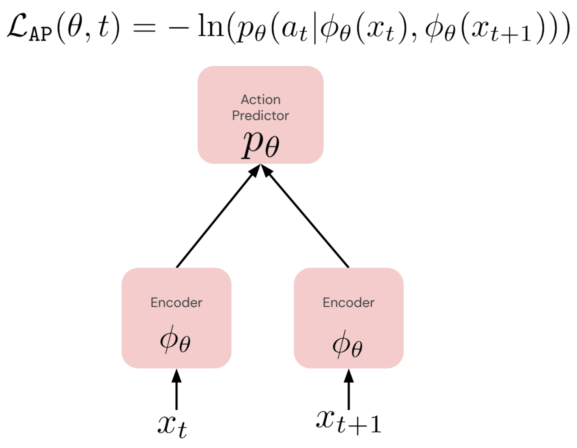
$\mathcal{L}_{\texttt{AP}}(\theta,t)=-\ln(p_\theta(a_t|\phi_\theta(x_t),\phi_\theta(x_{t+1})))$
Action Predictor
p_θ
Encoder
ϕ_θ
Encoder
ϕ_θ
x_t
x_{t+1}

Figure 6.1: Action Prediction Architecture.

$(X_k)_{k=0}^t$ to be able to compute the intrinsic reward for the state X_{t+1}. This memory is called an episodic memory because in practice it is easier to just keep in memory the actual episode. Then we compute the reward:

$$r(X_t, A_t) = \frac{1}{\sqrt{\sum_{k=0}^{t+1} \frac{\epsilon}{d_{\mathbb{R}^N}(\phi_\theta(X_k), \phi_\theta(X_{t+1}))^2 + \epsilon}}}.$$

A very similar technique is used in the Never Give Up (NGU) paper[Badia et al., 2020b] to compute an episodic intrinsic reward except that they do not use all the data in the memory to compute the smoothed visitation count but only the nearest neighbours of X_{t+1} in the episodic memory $(X_k)_{k=0}^t$. We encourage the reader to read this paper for more details.

6.4 Uncertainty-based Exploration

The high level idea behind uncertainty-based exploration methods is to incentivize the RL agent to go to more uncertain states. To do so, the agent should be able to compute an uncertainty estimate of how well it knows the actual state. One way to achieve this is to create a task for each state and having a way to measure how far is the agent from solving the task as a way to estimate how well the agent knows the actual state. More concretely, for each state $x \in \mathcal{X}$ we define a target quantity $f(x) \in \mathbb{R}^N$ unknown to the the agent unless the agent visits state x when $f(x)$ is revealed. The agent has to learn a function $g(x)$ to estimate $f(x)$. Then, the reward $r(X_t, A_t)$ associated to going to state X_{t+1} from the state-action couple (X_t, A_t) is simply $r(X_t, A_t) = \|f(X_{t+1}) - g(X_{t+1}))\|_2^2$. One instantiation of this simple idea is Random Network Distillation (RND) [Burda et al., 2019].

6.5 Random Network Distillation

Random Network Distillation (RND) [Burda et al., 2019] is a simple exploration method that consists in training an encoder such that its outputs fit the outputs of *another* fixed and randomly initialized encoder and using the training loss as an intrinsic reward to be optimized by an RL algorithm. More precisely, let $N \in \mathbb{N}^*$ be the embedding size and let us note $\phi_\theta : \mathcal{X} \to \mathbb{R}^N$ the encoder, also called predictor network, with trainable weights θ and $\phi_\psi : \mathcal{X} \to \mathbb{R}^N$ the fixed and randomly initialized encoder, also called target network, with fixed weights ψ. In addition, let us suppose that we have a trajectory $\big((X_t, A_t)_{t=0}^{T-1}\big)$ collected by an RL agent, then the loss $\mathcal{L}_{\texttt{RND}}(\theta)$ to minimize w.r.t. the online network parameters is defined as:

$$\mathcal{L}_{\texttt{RND}}(\theta, t) = \|\phi_\theta(X_t) - \texttt{sg}(\phi_\psi(X_t))\|_2^2, \quad \mathcal{L}_{\texttt{RND}}(\theta) = \frac{1}{T}\sum_{t=0}^{T-1} \mathcal{L}_{\texttt{RND}}(\theta, t),$$

and the unnormalized reward associated to the transition (X_t, A_t, X_{t+1}) is defined as $\ell_t = \ell(X_t, A_t) = \mathcal{L}_{\texttt{RND}}(\theta, t+1)$ where $0 \le t \le T-2$.

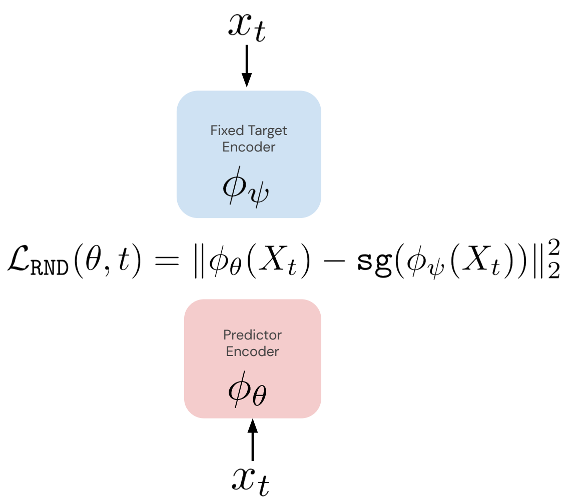
x_t
Fixed Target Encoder
ϕ_ψ
$\mathcal{L}_{\texttt{RND}}(\theta, t) = \|\phi_\theta(X_t) - \texttt{sg}(\phi_\psi(X_t))\|_2^2$
Predictor Encoder
ϕ_θ
x_t

Figure 6.2: RND's Neural Architecture.

6.5.1 Reward Normalization Scheme

In RND, the batched raw rewards $((\ell_t^j)_{t=0}^{T-2})_{j=0}^{B-1}$ are normalized by an EMA estimate of their standard deviation.

More precisely, we first set the EMA mean to $\overline{r} = 0$, the EMA mean of squares to $\overline{r^2} = 0$ and the counter to $c = 1$. Then, for the c-th batch of raw rewards $((\ell_t^j)_{t=0}^{T-2})_{j=0}^{B-1}$, we compute the batch mean $\overline{r_c}$ and the batch mean of squares $\overline{r_c^2}$:

$$\overline{r_c} = \frac{1}{B(T-1)} \sum_{j=0}^{B-1} \sum_{t=0}^{T-2} \ell_t^i, \qquad \overline{r_c^2} = \frac{1}{B(T-1)} \sum_{j=0}^{B-1} \sum_{t=0}^{T-2} (\ell_t^i)^2.$$

We then update $\overline{r}$, $\overline{r^2}$ and c:

$$\overline{r} \leftarrow \alpha_r \overline{r} + (1 - \alpha_r)\overline{r_c}, \qquad \overline{r^2} \leftarrow \alpha_r \overline{r^2} + (1 - \alpha_r)\overline{r_c^2}, \qquad c \leftarrow c + 1,$$

where $\alpha_r = 0.99$. We compute the adjusted EMA mean μ_r, the adjusted EMA mean of squares μ_{r^2}:

$$\mu_r = \frac{\overline{r}}{1 - \alpha_r^c}, \qquad \mu_{r^2} = \frac{\overline{r^2}}{1 - \alpha_r^c}.$$

Finally the EMA estimation of the standard deviation is $\sigma_r = \sqrt{\max(\mu_{r^2} - \mu_r^2, 0) + \epsilon}$, where $\varepsilon = 10^{-8}$ is a small numerical regularization. The normalized rewards are ℓ_t^j / σ_r.

Chapter 7

Planning

Bibliography

[Abdolmaleki et al., 2018] Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. (2018). Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920.*

[Azar et al., 2012] Azar, M. G., Gómez, V., and Kappen, H. J. (2012). Dynamic policy programming. *The Journal of Machine Learning Research*, 13(1):3207–3245.

[Azar et al., 2011] Azar, M. G., Munos, R., Ghavamzadaeh, M., and Kappen, H. J. (2011). Speedy q-learning. In *Advances in neural information processing systems*. Spain, Granada: NIPS.

[Azar et al., 2017] Azar, M. G., Osband, I., and Munos, R. (2017). Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning*, pages 263–272. PMLR.

[Badia et al., 2020a] Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., and Blundell, C. (2020a). Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*, pages 507–517. PMLR.

[Badia et al., 2020b] Badia, A. P., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, A., et al. (2020b). Never give up: Learning directed exploration strategies. *arXiv preprint arXiv:2002.06038.*

[Bellemare et al., 2017] Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR.

[Bellemare et al., 2013] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.

[Burda et al., 2019] Burda, Y., Edwards, H., Storkey, A., and Klimov, O. (2019). Exploration by random network distillation. In *Seventh International Conference on Learning Representations*, pages 1–17.

[Cho et al., 2014] Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259.*

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555.*

[Dabney et al., 2018] Dabney, W., Ostrovski, G., Silver, D., and Munos, R. (2018). Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pages 1096–1105. PMLR.

[Espeholt et al., 2018] Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. (2018). Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pages 1407–1416. PMLR.

[Fox et al., 2015] Fox, R., Pakman, A., and Tishby, N. (2015). Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562.*

[Geist et al., 2014] Geist, M., Scherrer, B., et al. (2014). Off-policy learning with eligibility traces: a survey. *J. Mach. Learn. Res.*, 15(1):289–333.

[Geist et al., 2019] Geist, M., Scherrer, B., and Pietquin, O. (2019). A theory of regularized markov decision processes. In *International Conference on Machine Learning*, pages 2160–2169. PMLR.

[Haarnoja et al., 2017] Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. (2017). Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361. PMLR.

[Haarnoja et al., 2018] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. (2018). Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

[Hausknecht and Stone, 2015] Hausknecht, M. and Stone, P. (2015). Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*.

[Hernández-Lerma and Lasserre, 2012] Hernández-Lerma, O. and Lasserre, J. B. (2012). *Discrete-time Markov control processes: basic optimality criteria*, volume 30. Springer Science & Business Media.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Horgan et al., 2018] Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Van Hasselt, H., and Silver, D. (2018). Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*.

[Jaderberg et al., 2017] Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., et al. (2017). Population based training of neural networks. *arXiv preprint arXiv:1711.09846*.

[Kapturowski et al., 2018] Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., and Dabney, W. (2018). Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Mnih et al., 2016] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

[Mukkamala and Hein, 2017] Mukkamala, M. C. and Hein, M. (2017). Variants of rmsprop and adagrad with logarithmic regret bounds. In *International Conference on Machine Learning*, pages 2545–2553. PMLR.

[Munos, 2007] Munos, R. (2007). Performance bounds in l_p-norm for approximate value iteration. *SIAM journal on control and optimization*, 46(2):541–561.

[Munos et al., 2016] Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. G. (2016). Safe and efficient off-policy reinforcement learning. *arXiv preprint arXiv:1606.02647*.

[Nair et al., 2015] Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., Panneershelvam, V., Suleyman, M., Beattie, C., Petersen, S., et al. (2015). Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*.

[Pathak et al., 2017] Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR.

[Puterman, 2014] Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[Riedmiller, 2005] Riedmiller, M. (2005). Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European conference on machine learning*, pages 317–328. Springer.

[Schaul et al., 2019] Schaul, T., Borsa, D., Ding, D., Szepesvari, D., Ostrovski, G., Dabney, W., and Osindero, S. (2019). Adapting behaviour for learning progress. *arXiv preprint arXiv:1912.06910*.

[Schaul et al., 2015] Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

[Scherrer et al., 2015] Scherrer, B., Ghavamzadeh, M., Gabillon, V., Lesner, B., and Geist, M. (2015). Approximate modified policy iteration and its application to the game of tetris. *J. Mach. Learn. Res.*, 16:1629–1676.

[Schulman et al., 2015] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR.

[Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[Song et al., 2019] Song, H. F., Abdolmaleki, A., Springenberg, J. T., Clark, A., Soyer, H., Rae, J. W., Noury, S., Ahuja, A., Liu, S., Tirumala, D., et al. (2019). V-mpo: On-policy maximum a posteriori policy optimization for discrete and continuous control. *arXiv preprint arXiv:1909.12238*.

[Sutton, 1988] Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44.

[Sutton et al., 2000] Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

[Van Hasselt et al., 2016] Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

[Vieillard et al., 2020] Vieillard, N., Kozuno, T., Scherrer, B., Pietquin, O., Munos, R., and Geist, M. (2020). Leverage the average: an analysis of kl regularization in rl. *arXiv preprint arXiv:2003.14089*.

[Wang et al., 2016] Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR.

[Xu et al., 2020] Xu, Z., van Hasselt, H., Hessel, M., Oh, J., Singh, S., and Silver, D. (2020). Meta-gradient reinforcement learning with an objective discovered online. *arXiv preprint arXiv:2007.08433*.

[Xu et al., 2018] Xu, Z., van Hasselt, H., and Silver, D. (2018). Meta-gradient reinforcement learning. *arXiv preprint arXiv:1805.09801*.