Foundations of Reinforcement Learning

Pablo Villanueva Domingo

April 2026

Contents

Abstract

Reinforcement learning (RL) is a framework for sequential decision making under uncertainty, in which an agent interacts with an environment by observing states, selecting actions, and receiving rewards, with the goal of maximizing cumulative discounted return. These notes provide a self-contained mathematical treatment of the foundations, progressing from first principles to modern deep RL algorithms. The notes are organized as follows. We begin by building the probabilistic scaffolding from the ground up: Markov Processes, Markov Reward Processes, and Markov Decision Processes, establishing Bellman equations and

closed-form value functions along the way. Classical solution methods are then covered, including dynamic programming, Monte Carlo, temporal difference learning, and the Generalized Advantage Estimator. The bulk of the notes concerns policy gradient methods: the Policy Gradient Theorem, and the practical algorithms TRPO and PPO. Deterministic policies are treated separately through the Deterministic Policy Gradient theorem and the DDPG algorithm. The notes also cover more advanced contents such as rewards engineering and continuous time RL. These notes are mainly based on the David Silver lectures [1], Sutton & Barto book [2], and OpenAI's Spinning Up [3].

1 Markov Processes

The mathematical foundation of reinforcement learning is the Markov Decision Process, which we build up progressively in the next sections. We start from reviewing the fundamental features of stochastic processes and the Markov property, which are of great importance in RL.

1.1 Stochastic processes

We define a stochastic process as a tuple $\langle \mathcal{S}, P \rangle$, where $\mathcal{S}$ is the set state, and P the probability distribution. We indicate random variables in upper case as S_t, while s or s_t refer to some state $s \in \mathcal{S}$. In a stochastic process, $P(s_0, t_0; s_1, t_1; ...; s_T, t_k)$ indicates the probability of being at state s_0 at time t_0, at state s_1 at time t_1 etc. Hereafter we won't explicitly show the time, making use of the shorthand notation $P(s_0, s_1, ..., s_k)$.

The probability of transitioning from state s at time t to s' at time $t+1$ in one step, or the transition probability, is given by $P(S_{t+1} = s'|S_t = s)$. A stochastic process is time homogeneous if the transition probability is time invariant: $P(S_{t+1} = s'|S_t = s) = P(S_{t++k+1} = s'|S_{t+k} = s)\ \forall k$. In that case we can just write it as

$$\mathcal{P}_{ss'} = P(s'|s) = P(S_{t+1} = s'|S_t = s) \tag{1.1.1}$$

In the following we will always assume time homogeneity. The transition probability fulfills the requirement $\sum_{s'} \mathcal{P}_{ss'} = 1$, since every state s should transition to some other one (maybe itself). Therefore, $\mathcal{P}$ can be viewed as a stochastic matrix with rows summing to 1.

In a stochastic process, a state S_t transitions to the next step state S_{t+1}, which is sampled from the transition probability by $S_{t+1} \sim P(S_{t+1}|S_t)$. For a deterministic scenario, they follow a deterministic relation of the form $S_{t+1} = f(S_t)$. Note that in this case, the probability is just $P(S_{t+1}|S_t) = \mathbb{1}(S_{t+1} = f(S_t))$, where $\mathbb{1}(x)$ is 1 when x is fulfilled, 0 otherwise (acting thus as a Kronecker or Dirac delta for the discrete and continuous case respectively). Thus, deterministic processes can also be modeled as stochastic ones in some cases.

To derive the probability of a multi-step sequence, we use the definition of conditional probability, $P(B|A) = P(A, B)/P(A)$. Applying this to $P(s_0, s_1, ..., s_k)$ leads to the chain rule $P(s_0, s_1, ..., s_k) = P(s_k|s_0, s_1, ..., s_{k-1})P(s_0, s_1, ..., s_{k-1})$. Equivalently, we also have $P(s_0, s_1, ..., s_k) = P(s_1, ..., s_k|s_0)P(s_0)$.

1.2 Markov property

A *Markov process* is a stochastic process which fulfills the *Markov property*: the next state only depend on the current state:

$$P(s_k|s_0, s_1, ..., s_{k-1}) = P(s_k|s_{k-1}) \tag{1.2.1}$$

i.e., the future only depends on the present state, not on its history. This is what makes the process memoryless.

A Markov Process (MP) is a memoryless random process over a state space $\mathcal{S}$. It is defined by the tuple $\langle \mathcal{S}, \mathcal{P} \rangle$, where $\mathcal{S}$ is a set of states and $\mathcal{P}$ is the transition probability matrix.

From the Markov property, we can write the joint probability of being in states $s_0, s_1, ..., s_k$ as

$$P(s_0, s_1, ..., s_k) = P_0(s_0) \prod_{i=0}^{k-1} P(s_{i+1}|s_i) \tag{1.2.2}$$

where $P(s_0)$ is the probability distribution of the initial state. Therefore, the joint probability of a sequence factorizes into a product of one-step transitions.

The probability of getting a trajectory $\tau = (s_0, s_1, ..., s_k)$ given we start at state s_0 is $P(\tau|s_0) = P(s_1, ..., s_k|s_0)$. Applying the Markov property, this becomes a product over the transition matrix entries:

$$P(s_1, ..., s_k|s_0) = \mathcal{P}_{s_0 s_1 ... s_k} = \mathcal{P}_{s_0 s_1} \mathcal{P}_{s_1 s_2} ... \mathcal{P}_{s_{k-1} s_k} = \prod_{t=0}^{k-1} \mathcal{P}_{s_t s_{t+1}} \tag{1.2.3}$$

Finally, the probability of being in a state s at time t, $P_t(s) = P(S_t = s)$ fulfills a Chapman-Kolmogorov-like equation:

$$P(S_{t+1}) = \sum_s P(S_t, S_{t+1}) = \sum_s P(S_{t+1}|S_t) P(S_t) = \sum_s \mathcal{P}_{S_t S_{t+1}} P(S_t) \tag{1.2.4}$$

which can be viewed as a vector equation $\boldsymbol{P}_{t+1} = \mathcal{P}\boldsymbol{P}_t$ (sorry for the notation). A stationary distribution, if exists, is a fixed point of the above equation: $\boldsymbol{P}_{sta} = \mathcal{P}\boldsymbol{P}_{sta}$.

1.3 k-step transition probability

The k-step transition probability is the probability of going from s to s' in k steps, i.e., the probability to finish after k steps at state s' given that we start at state s:

$$P_k(s'|s) = P(S_{t+k} = s'|S_t = s) \tag{1.3.1}$$

Using the Markov property, it can be written in a concise form as

$$P_k(s_k|s_0) = \sum_{s_1 ... s_{k-1}} \prod_{i=0}^{k-1} \mathcal{P}_{s_i s_{i+1}} \tag{1.3.2}$$

For instance, $P^{(3)}(s_3|s_0; k) = \sum_{s_1, s_2} \mathcal{P}_{s_0 s_1} \mathcal{P}_{s_1 s_2} \mathcal{P}_{s_2 s_3}$.It is useful to also define $P_0(s'|s) = \mathbb{1}(s' = s)$. The k-step transition probability fulfills the recursive relations

$$P_k(s'|s) = \sum_{s''} P(s'|s'') P_{k-1}(s''|s) \quad \text{and} \quad P_k(s'|s) = \sum_{s''} P_{k-1}(s'|s'') P(s''|s) \tag{1.3.3}$$

And in general the so-called Chapman-Kolmogorov equation in discrete time.

$$P_{m+n}(s'|s) = \sum_{s''} P_m(s'|s'') P_n(s''|s) \tag{1.3.4}$$

In matrix form is given by $\mathcal{P}^k$

$$P_k(s'|s) = \left(\mathcal{P}^k\right)_{ss'} \tag{1.3.5}$$

Thus, in matrix form, the k-step transition probability from state s_i to s_j is the (i, j) entry of $\mathcal{P}^k$. The Chapman-Kolmogorov equation just takes the form of the matrix product $\mathcal{P}^{m+n} = \mathcal{P}^m \mathcal{P}^n$.

Note that this is the probability to be after k steps at s', but it can arrive earlier, so it is not the probability of first arriving at step k. The probability to reach state s' ever is thus $P_{(\infty)}(s'|s)$.

1.4 Expectation values

It is straightforward to compute expectation values in Markov processes. For a function $f(s)$ we can compute the expectation value at time $t + 1$ conditioned at time t as

$$\mathbb{E}[f(S_{t+1})|S_t = s] = \sum_{s'} \mathcal{P}_{ss'} f(s') = \sum_{s'} P_1(s'|s) f(s') \tag{1.4.1}$$

At time $t+2$, and using the law of total expectation

$$\mathbb{E}[f(S_{t+2})|S_t = s] = \mathbb{E}\left[\mathbb{E}[f(S_{t+2})|S_{t+1} = s']|S_t = s\right] \tag{1.4.2}$$

$$= \mathbb{E}\left[\sum_{s''} \mathcal{P}_{s's''} f(s'')|S_t = s\right] \tag{1.4.3}$$

$$= \sum_{s',s''} \mathcal{P}_{ss'}\mathcal{P}_{s's''} f(s'') = \sum_{s''} P_2(s''|s) f(s'') \tag{1.4.4}$$

We can easily see a pattern arising. In general for any $k \geq 0$:

$$\mathbb{E}[f(s_{t+k})|s_t = s] = \sum_{s'} P_k(s'|s) f(s') \tag{1.4.5}$$

These expectation values will be commonly used in the following sections.

1.5 Examples of Markov processes

Here we briefly discuss some examples of stochastic and Markov processes.

- **Random walk** Consider an uncorrelated gaussian noise $\eta_t \sim \mathcal{N}(0, \sigma^2)$, $Cov(\eta_t, \eta_s) = \sigma^2 \delta_{ts}$ and the constant drift μ. Then, a random walk with drift is given by

$$X_{t+1} = X_t + \mu + \eta_t \tag{1.5.1}$$

  Unrolling the recurrence, it can be written as $X_{t+k} = X_t + \mu k + \sum_{i=0}^{k-1} \eta_{t+i}$. Thus, the conditioned mean is $\mathbb{E}[X_{t+k}|X_t] = X_t + \mu k$, and the variance $Var[X_{t+k}|X_t] = k\,\sigma^2$. Also, the transition probability can be written as a gaussian in η_t, from the equation $\eta_t = X_{t+1} - X_t - \mu$: $P(x'|x) = \exp[(x' - x - \mu)^2/(2\sigma^2)]/\sqrt{2\pi\sigma^2}$. That allows to recover e.g. $\mathbb{E}[X_{t+1}|X_t = x] = \int dx' P(x'|x) = x + \mu$, consistent with the above result. In general, since a sum of uncorrelated normally distributed η_t are also normally distributed, we have $P_k(x'|x) = \exp[(x' - x - k\mu)^2/(2k\sigma^2)]/\sqrt{2\pi\sigma^2}$.

- **Autoregressive model of order 1** The AR(1) model is the discrete-time analogy of the continuous Ornstein-Uhlenbeck process and reads

$$X_{t+1} = X_t + (1-\theta)(\mu - X_t) + \eta_t \tag{1.5.2}$$

  where μ stands for the mean. For a mean zero process, it just reads $X_{t+1} = \phi X_t + \eta_t$. From the recurrence equation, we get by induction $X_{t+k} = \theta^k X_t + (1-\theta^k)\mu + \sum_{i=0}^{k-1} \theta^{k-1-i}\eta_{t+i}$. Note that we recover the random walk in the limit $\theta \to 1$ with $(1-\theta)\mu \to \mu$. The conditioned mean reads $\mathbb{E}[X_{t+k}|X_t] = \theta^k X_t + (1-\theta^k)\mu$ and the variance $Var[X_{t+k}|X_t] = \sigma^2 \frac{1-\theta^{2k}}{1-\theta^2}$. Equivalently, $X_t = \theta^k X_{t-k} + (1-\theta^k)\mu + \sum_{i=0}^{k-1} \theta^i \eta_{t-1-i}$. If $|\theta| < 1$, for $k \to \infty$, we can write $X_t = \mu + \sum_{i=0}^{\infty} \theta^i \eta_{t-1-i}$, so $\mathbb{E}[X_t] = \mu$, $Var(X_t) = \sigma^2 \frac{1}{1-\theta^2}$ and the covariance $Cov(X_{t+k}, X_t) = \sigma^2 \theta^k \frac{1}{1-\theta^2}$.

2 Markov Reward Process

A Markov Reward Process (MRP) is a tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ that augments a Markov Process with a scalar reward signal. At each step the process visits a state $s \in \mathcal{S}$, transitions to s' with probability $\mathcal{P}_{ss'}$, and emits a reward R_{t+1} drawn from $\mathcal{R}$. There are no actions yet — the dynamics are entirely driven by the transition operator. The reward function $\mathcal{R}$ encodes the desirability of states, and the central quantity of interest is the *value function* $V(s)$, the expected cumulative discounted reward the process will accumulate starting from s. Solving for V in closed form via the Bellman equation is the main goal of this section, and the resulting tools carry over directly to the full MDP.

2.1 Reward and return

The reward signal comes in two related forms. The state reward $\mathcal{R}_s$ is the expected immediate reward upon leaving state s, regardless of where the process goes next:

$$\mathcal{R}_s = \mathbb{E}[R_{t+1}|S_t = s] \tag{2.1.1}$$

The transition reward $\mathcal{R}_{ss'}$ conditions additionally on the next state, capturing situations where the reward depends on which successor is reached:

$$\mathcal{R}_{ss'} = \mathbb{E}[R_{t+1}|S_t = s, S_{t+1} = s'] \tag{2.1.2}$$

The two are related by marginalizing over the next state with the transition probabilities:

$$\mathcal{R}_s = \sum_{s'} \mathcal{P}_{ss'}\mathcal{R}_{ss'} \tag{2.1.3}$$

so $\mathcal{R}_s$ is simply the expectation of $\mathcal{R}_{ss'}$ over the distribution of successor states. Throughout the notes we mostly work with $\mathcal{R}_s$ for brevity. Note also that all relations use R_{t+1} rather than R_t, reflecting the convention that the reward is emitted upon leaving S_t and arriving at S_{t+1}.

The *return* G_t aggregates all future rewards into a single scalar by summing them with exponentially decaying weights. The *discount factor* $\gamma \in [0, 1)$ controls how rapidly this decay occurs (see the motivation below). The return from time t is defined by

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \tag{2.1.4}$$

Note that G_t starts with R_{t+1}, since it accumulates the rewards received *after* being in state S_t. A key structural property is that the return satisfies the recurrence

$$G_t = R_{t+1} + \gamma \sum_{k=1}^{\infty} \gamma^{k-1} R_{t+k+1} = R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} = R_{t+1} + \gamma G_{t+1} \tag{2.1.5}$$

which expresses the return as immediate reward plus discounted future return. This one-step decomposition is the engine behind the Bellman equation derived in the next subsection.

where we have rearrange the dummy indices in the sum.

The discount factor γ controls how much the agent values future rewards relative to immediate ones. Three complementary motivations justify it. First, it ensures mathematical well-posedness: for bounded rewards $|R_t| \leq R_{\max}$, the infinite sum G_t converges absolutely since $\sum_{k=0}^{\infty} \gamma^k R_{\max} = R_{\max}/(1-\gamma) < \infty$; without discounting ($\gamma = 1$) the return may diverge. Second, it has an economic interpretation as a preference for rewards sooner rather than later, analogous to a per-step interest rate of $1/\gamma - 1$. Third, in stochastic environments γ encodes uncertainty about the future: if each step has an independent probability $1 - \gamma$ of being terminal, then γ^k is the probability that the episode has not ended by step k, so G_t is the expected undiscounted reward under this random horizon. The limit $\gamma \to 1$ recovers the undiscounted (average-reward) setting, which requires additional conditions for the value function to remain finite.

2.2 Value function and Bellman expectation equation

We define the state value function, or just the value function, as the expected return at state s

$$V(s) = \mathbb{E}[G_t|S_t = s] \tag{2.2.1}$$

Using the recurrence relation from Eq. 2.1.5, we can write the following expression for the value function:

$$V(s) = \mathbb{E}[R_{t+1} + \gamma G_{t+1}|S_t = s] = \mathcal{R}_s + \gamma \mathbb{E}[G_{t+1}|S_t = s] \tag{2.2.2}$$

Using the law of total expectation (see Appendix A.1)

$$\mathbb{E}[G_{t+1}|S_t = s] = \mathop{\mathbb{E}}_{s'\sim P(\cdot|S_t)}[\,\mathbb{E}[G_{t+1}|S_{t+1} = s']\,|S_t = s] = \mathop{\mathbb{E}}_{s'\sim P(\cdot|S_t)}[\,V(s')\,|S_t = s] = \sum_{s'} \mathcal{P}_{ss'}\,V(s') \quad (2.2.3)$$

The above leads to a recurrence relation for V known as the *Bellman expectation equation*:

$$\boxed{V(s) = \mathcal{R}_s + \gamma \sum_{s'} \mathcal{P}_{ss'}\,V(s')} \quad (2.2.4)$$

This can be casted in vector form

$$\boldsymbol{V} = \boldsymbol{\mathcal{R}} + \gamma \mathcal{P}\boldsymbol{V} \quad (2.2.5)$$

Note that this is a linear equation for $\boldsymbol{V}$, and thus can be solved as

$$\boldsymbol{V} = (I - \gamma\mathcal{P})^{-1}\,\boldsymbol{\mathcal{R}} \quad (2.2.6)$$

But this solution is usually very computationally expensive, and there are other more efficient ways to solve it.

2.3 Value function in terms of k-step transitions

We can obtain a more concise formula for the value function. Unrolling the Bellman expectation equation

$$V(s) = \mathcal{R}_s + \gamma \sum_{s'} \mathcal{P}_{ss'} V(s') \quad (2.3.1)$$

$$= \mathcal{R}_s + \gamma \sum_{s'} \mathcal{P}_{ss'} \left(\mathcal{R}_{s'} + \gamma \sum_{s''} \mathcal{P}_{s's''} V(s'') + ... \right) \quad (2.3.2)$$

$$= \mathcal{R}_s + \gamma \sum_{s'} \mathcal{P}_{ss'} \mathcal{R}_{s'} + \gamma^2 \sum_{s',s''} \mathcal{P}_{ss'} \mathcal{P}_{s's''} \mathcal{R}_{s''} + \gamma^3 \sum_{s',s'',s'''} \mathcal{P}_{ss'} \mathcal{P}_{s's''} \mathcal{P}_{s''s'''} \mathcal{R}_{s'''} + ... \quad (2.3.3)$$

$$(2.3.4)$$

where we can identify the the k-step transition probability in each term, getting finally

$$\boxed{V(s) = \sum_{k=0}^{\infty} \gamma^k \sum_{s'} P_k(s'|s) \mathcal{R}_{s'}} \quad (2.3.5)$$

with $P_0(s'|s) = \mathbb{1}(s' = s)$. Note that one could have reached the same result from expanding the operator $(I - \gamma\mathcal{P})^{-1}$ in $\boldsymbol{V} = (I - \gamma\mathcal{P})^{-1}\,\boldsymbol{\mathcal{R}}$. Expanding the operator in powers of $\mathcal{P}$:

$$\left[(I - \gamma\mathcal{P})^{-1}\right]_{ss'} = \sum_{k=0}^{\infty} \gamma^k \left(\mathcal{P}^k\right)_{ss'} = \sum_{k=0}^{\infty} \gamma^k P_k(s'|s) \quad (2.3.6)$$

There is an alternative way to derive the above result without using the recurrence relation, directly from the expectation. Applying Eq. 1.2.3 for $P(s_1...s_{k+1}|s_0)$ to the definition of the value function, we can find the following concise equivalent expression

$$V(s_0) = \sum_{k=0}^{\infty} \gamma^k \sum_{s_1,...s_{k+1}} P(s_1...s_{k+1}|s_0) \mathcal{R}_{s_k s_{k+1}} = \sum_{k=0}^{\infty} \gamma^k \sum_{s_1,...s_{k+1}} \left(\prod_{t=0}^{k} \mathcal{P}_{s_t s_{t+1}} \right) \mathcal{R}_{s_k s_{k+1}} \quad (2.3.7)$$

which, using $\mathcal{R}_s = \sum_{s'} \mathcal{P}_{ss'} \mathcal{R}_{ss'}$, can be rearranged to recover again Eq. 2.3.5.

2.4 Examples of MRPs

Here we discuss some generic examples of MRPs that are typically employed:

- **Terminal reward** Consider a MRP with reward only at the terminal state s_*. This is the case of many games where a player can either win or lose. The reward can be written as

$$\mathcal{R}_{ss'} = R_* \times [\mathbb{1}(s' = s_*) - \mathbb{1}(s = s_*)] \tag{2.4.1}$$

  This can be interpreted as getting a reward R_* only when the final state is s_* and the initial state different from s_*, and only arriving there. Since s_* is absorbing, we want to avoid rewarding staying there. Hence, we get $\mathcal{R}_s = R_*(\mathcal{P}_{ss_*} - \mathbb{1}(s = s_*))$. The discounted return can be written as $G_t = \gamma^{K-1} R_*$, with K the number of steps followed from t until termination. Using Eqs. 2.3.5 and 1.3.3, the value function takes the form

$$V(s) = R_* \sum_{k=0}^{\infty} \gamma^k \left[P_{k+1}(s_*|s) - P_k(s_*|s)\right] \tag{2.4.2}$$

  The quantity $P_{k+1}(s_*|s) - P_k(s_*|s)$ could be interpreted as the increment in probability to reach s_* in an additional step, or equivalently, the probability of reaching the terminal state in the next state minus the prob of being there in the current. Given that s_* is terminal, $P_k(s|s_*) = \mathbb{1}(s = s_*)\ \forall k$ and thus the value function at s_* vanishes $V(s_*) = 0$, as is expected for terminal states (since one cannot gain more starting from that state). The second term from $\mathcal{R}_{ss'}$ is required to fulfill that. The above expression can also be written as

$$V(s) = R_* \left[\mathcal{T}(s_*|s) - \mathbb{1}(s = s_*)\right] \tag{2.4.3}$$

  where $\mathcal{T}(s'|s) = (1-\gamma)\sum_{k=0}^{\infty} \gamma^k P_{k+1}(s'|s)$ is a discounted transition probability (so that sums to 1 $\sum_{s'} \mathcal{T}(s'|s) = 1$). Note that $\mathcal{T}(s'|s_*) = \mathbb{1}(s' = s_*)$. Then, it can be interpreted as R_* times the probability of eventually reaching the terminal state.

  It is immediate to generalize the above result to multiple terminal states s_{*i} (for instance winning and losing a single player game): $V(s) = \sum_i R_{*i} \sum_{k=0}^{\infty} \gamma^k (P_{k+1}(s_{*i}|s) - P_k(s_{*i}|s))$.

  It is straightforward to extend the above results to the undiscounted finite horizon T case, summing up to $T-1$, the telescopic sum cancels most of terms, getting $V(s) = R_* \left[P_T(s_*|s) - \mathbb{1}(s = s_*)\right]$, so it is basically the probability of reaching the terminal state within the time horizon.

- **Constant reward** In this case, every state provides the same reward R, except the terminal one. This reward may be negative and act as a cost, and thus it is valuable to get to the terminal state asap, or could be positive as a being alive reward, and then an agent would try to avoid reaching the termination. The reward is written as

$$\mathcal{R}_{ss'} = R \times [1 - \mathbb{1}(s' = s_*)] \tag{2.4.4}$$

  The second term accounts for not receiving a reward when reaching the terminal state. Then, applying Eq. 2.3.5 we get

$$V(s) = R \sum_{k=0}^{\infty} \gamma^k \left[1 - P_{k+1}(s_*|s)\right] = \frac{R}{(1-\gamma)} \left[1 - \mathcal{T}(s_*|s)\right] \tag{2.4.5}$$

  The term $1 - P_{k+1}(s_*|s)$ can be interpreted as the probability of not reaching the terminal state in $k+1$ steps, and $1 - \mathcal{T}(s_*|s)$ the probability of not reaching s_* ever. Then, the value is essentially the reward times the discounted probability of not finishing starting from s. Note that it vanishes for the terminal state, $V(s_*) = 0$.

The finite horizon case is simply $V(s) = RT\left(1 - (1/T)\sum_{k=0}^{T-1} P_{k+1}(s_*|s)\right) = RT\left[1 - \mathcal{T}(s_*|s)\right]$, where we can see the equivalence $(1-\gamma)^{-1} \to T$. In practice, many MRPs (and MDPs) are modelled as a mix between constant and terminal rewards, for instance Gridworld.

- **Rewarded state with finite horizon** Here we consider an undiscounted MRP with finite horizon T, and some non-terminal state s_* which is the only one rewarded. An agent should have to reach that state and remain there (or come back many times) to maximize reward within the time window. An example would be a game where the player has to collect the maximum number of points during some restricted time. Similarly to previous cases, the reward is given by $\mathcal{R}_{ss'} = R \times \mathbb{1}(s' = s_*)$, while now s_* is not terminal. The value function is now

$$V(s) = R_* \sum_{k=0}^{T} P_{k+1}(s_*|s) = R_*\, T\, \mathcal{T}(s_*|s) \tag{2.4.6}$$

  $V(s_*)$ is then given by the probability of staying in the rewarded state or leaving it and coming back.

- **Autoregressive reward** Consider a state space which follows an autoregressive process such as the one discussed in Sec. 1.5, with $X_{t+k} = \theta^k X_t + (1-\theta^k)\mu + \sum_{i=0}^{k-1} \theta^{k-1-i}\eta_{t+i}$. Lets assume that the rewards are given just by $R_t = \alpha X_t$. Then, the value function gets the concise form

$$V(x) = \frac{\alpha}{1-\gamma}\left[\mu + \frac{1-\gamma}{1-\gamma\theta}(x-\mu)\right] \tag{2.4.7}$$

2.5 Variance of the return

Lets estimate the variance of the return G_t. In this section we do not aim to get an accurate prediction of the variance, but instead a scaling argument to identify which parameters are relevant to set the scale of the variance. For that, we need to take some statistical assumptions on the distribution of rewards. First, lets assume that the variance of the rewards is approximately constant $\mathrm{Var}(r_t) = \sigma^2$. We also need to take some assumption regarding the covariance between rewards. We can expect that two reward at close times would be more correlated than those at greater separation. Defining the correlation of rewards between the current and next step as ρ, with $|\rho| \leq 1$, we can write the covariance as $\mathrm{Cov}(r_t, r_{t+k}) = \sigma^2\rho^k$ for $k \geq 0$. Then, using the identity $\mathrm{Var}(\sum_i a_i X_i) = \sum_{i,j} a_i a_j \,\mathrm{Cov}(X_i, X_j) = \sum_{i,j} a_i^2 \,\mathrm{Var}(X_i) + 2\sum_{i<j} a_i a_j \,\mathrm{Cov}(X_i, X_j)$, we can split the total variance expression into diagonal ($i = j$) and off-diagonal terms ($i \neq j$):

$$\mathrm{Var}(G_t) = \sigma^2 \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \gamma^{i+j}\rho^{|i-j|} = \sigma^2 \sum_{i=0}^{\infty} \gamma^{2i} + 2\sigma^2 \sum_{i<j} \gamma^{i+j}\rho^{j-i} \tag{2.5.1}$$

The diagonal terms sum to $\sigma^2/(1-\gamma^2)$. For the off-diagonal, let $k = j - i > 0$, so $j = i + k$, getting also geometric series:

$$2\sigma^2 \sum_{i=0}^{\infty}\sum_{k=1}^{\infty} \gamma^{i+(i+k)}\rho^k = 2\sigma^2 \sum_{i=0}^{\infty} \gamma^{2i} \sum_{k=1}^{\infty} (\gamma\rho)^k = 2\sigma^2 \cdot \frac{1}{1-\gamma^2} \cdot \frac{\gamma\rho}{1-\gamma\rho} \tag{2.5.2}$$

We thus get the final expression for the variance of the return

$$\mathrm{Var}(G_t) = \frac{\sigma^2}{1-\gamma^2}\left(1 + \frac{2\gamma\rho}{1-\gamma\rho}\right) = \frac{\sigma^2}{1-\gamma^2} \cdot \frac{1+\gamma\rho}{1-\gamma\rho} \tag{2.5.3}$$

From this expression we can see that either high γ or high correlations ρ increase the variance. We can also estimate the signal-to-noise ratio $SNR = \mathbb{E}[G_t]^2/\,\mathrm{Var}(G_t)$. Defining $\overline{r}_t = (1-\gamma)\sum_{k=0}^{\infty} \gamma^k \mathbb{E}[R_{t+k+1}]$, we can write $\mathbb{E}[G_t] = \overline{r}_t/(1-\gamma)$. Then, the SNR reads

$$SNR = \frac{1+\gamma}{1-\gamma}\frac{1-\gamma\rho}{1+\gamma\rho}\frac{\overline{r}_t^2}{\sigma^2} \tag{2.5.4}$$

where we can notice that high γ increases the signal-to-noise ratio, while the correlation ρ reduces it.

We can apply similar arguments for the undiscounted case with time horizon T. Instead of the geometric series of γ the sum would sum up to T, so we should replace $1/(1-\gamma^2)$ by T: $G_t \simeq \sigma^2 T \cdot \frac{1+\rho}{1-\rho}$. Performing properly the sumations we get an additional (likely lower) factor suppressed by T: $G_t = \sigma^2 T \cdot \frac{1+\rho}{1-\rho}\left(1 - 2\frac{\rho}{T}\frac{1-\rho^T}{1-\rho^2}\right)$. Again, higher time horizons linearly increase the variance. However, the signal-to-noise ratio gets approximately independent of T: $SNR \simeq \frac{1-\rho}{1+\rho}\frac{\bar{r}_t^2}{\sigma^2}$.

3 Markov Decision Process

A Markov Decision Process (MDP) extends the Markov Reward Process by introducing an *action space* $\mathcal{A}$. At each step, the agent observes the current state $s \in \mathcal{S}$, selects an action $a \in \mathcal{A}$ according to a policy $\pi(a|s)$, and the environment transitions to a new state s' with probability $P(s'|s,a)$ while emitting a reward R_{t+1}. An MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$; the goal is to find a policy π^* that maximizes the expected return $J(\pi) = \mathbb{E}_\pi[G_0]$. Since a fixed policy π marginalizes out the actions, every MDP reduces to an MRP $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi \rangle$ and all MRP results carry over.

3.1 Reward function

The reward signal in an MDP depends on both the current state and the action taken, unlike the MRP where it depended on the state alone. The state-action reward $\mathcal{R}_s^a$ is the expected immediate reward upon taking action a in state s, averaging over the stochastic next state:

$$\mathcal{R}_s^a = \mathbb{E}[R_{t+1} | S_t = s, A_t = a] \tag{3.1.1}$$

The transition reward $\mathcal{R}_{ss'}^a$ conditions additionally on the successor state, and is specified by a reward function $R(s,a,s')$:

$$\mathcal{R}_{ss'}^a = \mathbb{E}[R_{t+1} | S_t = s, A_t = a, S_{t+1} = s'] = R(s,a,s') \tag{3.1.2}$$

The two are related by marginalizing over the next state using the transition kernel $\mathcal{P}_{ss'}^a = P(s'|s,a)$, the probability of reaching s' from s under action a:

$$\mathcal{R}_s^a = \sum_{s'} \mathcal{P}_{ss'}^a \, \mathcal{R}_{ss'}^a \tag{3.1.3}$$

The transition kernel is normalized, $\sum_{s'} \mathcal{P}_{ss'}^a = 1$, and for terminal states $\mathcal{P}_{s_T s'}^a = 0$ for all $s' \neq s_T$ and all a.

Since the policy $\pi(a|s)$ specifies a distribution over actions at each state, it marginalizes out the action and induces an effective transition operator and reward on the state space alone:

$$\mathcal{P}_{ss'}^\pi = \sum_a \pi(a|s) \mathcal{P}_{ss'}^a \tag{3.1.4}$$

$$\mathcal{R}_s^\pi = \sum_a \pi(a|s) \mathcal{R}_s^a \tag{3.1.5}$$

Finally, the fundamental expectation formulas under π for a function of a single transition and of a full trajectory are:

$$\mathbb{E}_\pi[f(s,a,s') \mid s,a] = \sum_{s'} \mathcal{P}_{ss'}^a \, f(s,a,s') \tag{3.1.6}$$

$$\mathbb{E}_\pi[f(s,a,s') \mid s] = \sum_a \pi(a|s) \sum_{s'} \mathcal{P}_{ss'}^a \, f(s,a,s') \tag{3.1.7}$$

For a full trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots)$, the expectation over the trajectory distribution induced by π is

$$\mathbb{E}_\pi[f(\tau)] = \sum_{s_0, a_0, \ldots} P(s_0, a_0, \ldots) \, f(\tau) \tag{3.1.8}$$

where the trajectory probability factorizes as $P(\tau) = \rho_0(s_0) \prod_{t\geq 0} \pi(a_t|s_t) \mathcal{P}^{a_t}_{s_t s_{t+1}}$, with ρ_0 the initial state distribution. Conditioning on the initial state-action pair (s_0, a_0), the k-step trajectory expectation is

$$\mathbb{E}_\pi[f(s_0, a_0, \ldots, s_k, a_k) \mid s_0, a_0] = \sum_{\substack{s_1,\ldots,s_k \\ a_1,\ldots,a_k}} \prod_{i=0}^{k-1} \mathcal{P}^{a_i}_{s_i s_{i+1}} \pi(a_{i+1}|s_{i+1})\, f(s_0, a_0, \ldots, s_k, a_k) \tag{3.1.9}$$

3.2 Value functions and Bellman expectation equations

The *action-value function* under a policy π, Q^π, is defined as the expected return starting from state s, taking action a and then following policy π:

$$Q^\pi(s, a) = \mathbb{E}[G_t|S_t = s, A_t = a] \tag{3.2.1}$$

The Q stands for *quality*. The *state value function* under a policy π, or just the value function, V^π, is defined as the expected return starting from state s and following policy π, and can be computed from $Q(s, a)$:

$$V^\pi(s) = \mathbb{E}_\pi[Q^\pi(s, a)] = \sum_a \pi(a|s) Q^\pi(s, a) \tag{3.2.2}$$

The advantage function measures how better an action a is relative to the average at state s under π:

$$\mathcal{A}^\pi(s, a) = Q^\pi(s, a) - V^\pi(s) \tag{3.2.3}$$

Similarly to MRPs, we can derive recursive relations for V and Q unrolling the return and using the law of total expectation:

$$\begin{aligned}
Q^\pi(s, a) &= \mathbb{E}[R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a] && (3.2.4)\\
&= \mathcal{R}^a_s + \gamma \mathop{\mathbb{E}}_{s'\sim P(\cdot|S_t)} [\mathbb{E}[G_{t+1}|S_{t+1} = s']|S_t = s, A_t = a] && (3.2.5)\\
&= \mathcal{R}^a_s + \gamma \mathop{\mathbb{E}}_{s'\sim P(\cdot|S_t)} [V^\pi(s')|S_t = s, A_t = a] && (3.2.6)\\
&= \mathcal{R}^a_s + \gamma \sum_{s'} \mathcal{P}^a_{ss'} V^\pi(s') && (3.2.7)\\
& && (3.2.8)
\end{aligned}$$

With the above relations, we can write the *Bellman expectation equations* for Q^π and V^π as

$$\boxed{\begin{aligned}
Q^\pi(s, a) &= \mathcal{R}^a_s + \gamma \sum_{s'} \mathcal{P}^a_{ss'} \sum_{a'} \pi(a'|s') Q^\pi(s', a') && (3.2.9)\\
V^\pi(s) &= \sum_a \pi(a|s) \left(\mathcal{R}^a_s + \gamma \sum_{s'} \mathcal{P}^a_{ss'} V^\pi(s') \right) && (3.2.10)
\end{aligned}}$$

Note that the value functions of state s do not include the reward from arriving to s, but only the expected return from starting from state s. For that reason, terminal states have value function 0.

Usually the reward function is written in terms of the initial and final state of the transition with $\mathcal{R}^a_{ss'} = R(s, a, s')$, so that with $\mathcal{R}^a_s = \sum_{s'} \mathcal{P}^a_{ss'} \mathcal{R}^a_{ss'}$. The Bellman equations can be explicitly written in terms of $\mathcal{R}^a_{ss'}$ as

$$V^\pi(s) = \sum_{a,s'} \pi(a|s) \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma\, V^\pi(s')\right] \tag{3.2.11}$$

and

$$Q^\pi(s, a) = \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma\, V^\pi(s')\right] \tag{3.2.12}$$

With this formulation it is easy to see the value in an absorbing state s_*, where $\mathcal{P}^a_{s_*s} = \mathbb{1}(s = s_*)$, getting $V(s_*) = \mathcal{R}^\pi_{s_*s_*}/(1-\gamma)$. If we consider it as terminal, we should set $\mathcal{R}^\pi_{s_*s_*} = 0 = V(s_*)$.

Also, note that Sutton & Barto use $P(s', r|s, a)$, which can be treated as equivalent to our formalism.

For a deterministic policy, $a = \pi(s)$, and $V^\pi(s) = Q^\pi(s, \pi(s))$.

3.3 k-step transition probability

Analogously to Sec. 1.3, we can define the k-step transition probability is the probability of going from s, a to s', a' in k steps:

$$P_k(s', a'|s, a) = P(S_{t+k} = s', A_{t+k} = a'|S_t = s, A_t = a) \tag{3.3.1}$$

Using the Markov property, it can be written in a concise form as

$$P^\pi_k(s_k, a_k|s_0, a_0) = \sum_{s_1, a_1 \dots s_{k-1}, a_{k-1}} \prod_{i=0}^{k-1} P^\pi(s_{i+1}, a_{i+1}|s_i, a_i) = \sum_{s_1, a_1 \dots s_{k-1}, a_{k-1}} \prod_{i=0}^{k-1} \mathcal{P}^{a_i}_{s_i s_{i+1}} \pi(a_{i+1}|s_{i+1}) \tag{3.3.2}$$

With $P^\pi_0(s', a'|s, a) = \mathbb{1}(s' = s)\mathbb{1}(a' = a)$. We can also define the k-step transition probability from s, a to s' in k steps from the relation:

$$P^\pi_k(s', a'|s, a) = P^\pi_k(s'|s, a)\pi(a'|s') \tag{3.3.3}$$

Note that we can integrate over the intermediate actions

$$P^\pi_k(s_k|s_0, a_0) = \sum_{s_1, \dots s_{k-1}} \mathcal{P}^{a_0}_{s_0 s_1} \prod_{i=1}^{k-1} \mathcal{P}^\pi_{s_i s_{i+1}} \tag{3.3.4}$$

if we also integrate over the initial action:

$$P^\pi_k(s_k|s_0) = \sum_{a_0} \pi(a_0|s_0) P^\pi_k(s_k|s_0, a_0) = \sum_{s_1, \dots s_{k-1}} \prod_{i=0}^{k-1} \mathcal{P}^\pi_{s_i s_{i+1}} \tag{3.3.5}$$

Unfolding the Bellman equation for Q^π:

$$Q^\pi(s, a) = \mathcal{R}^a_s + \gamma \sum_{s'} \sum_{a'} \pi(a'|s') \mathcal{P}^a_{ss'} \mathcal{R}^{a'}_{s'} + \gamma^2 \sum_{s', s''} \sum_{a', a''} \pi(a'|s') \pi(a''|s'') \mathcal{P}^a_{ss'} \mathcal{P}^{a''}_{s's''} \mathcal{R}^{a''}_{s''} + \dots \tag{3.3.6}$$

$$\tag{3.3.7}$$

we can find the a concise equivalent expression for the action-value function, and integrating it, for the value

$$\boxed{\begin{aligned} Q^\pi(s, a) &= \sum_{k=0}^{\infty} \gamma^k \sum_{s', a'} P^\pi_k(s', a'|s, a) \mathcal{R}^{a'}_{s'} \qquad (3.3.8) \\ V(s) &= \sum_{k=0}^{\infty} \gamma^k \sum_{s'} P^\pi_k(s'|s) \mathcal{R}^\pi_{s'} \qquad (3.3.9) \end{aligned}}$$

which are analogous to Eq. 2.3.5 in plain MRPs. Finally, it is worth mentioning a last expression for Q obtained directly from the expectation and using the probability of a trajectory τ:

$$Q^\pi(s_0, a_0) = \sum_{\tau_{0:}} \sum_{t=0}^{\infty} \gamma^t P(\tau_{0:}|s_0, a_0) \mathcal{R}^{a_t}_{s_t s_{t+1}} \tag{3.3.10}$$

where $\tau_{0:} = (s_1, a_1, \dots)$. This expression can lead to Eq. 3.3.8 after rearranging.

3.4 Vector form and equivalent MRPs

MDPs can be interpreted as instances of MRPs in different ways. In this section we illustrate that they can be viewed as MRPs through two distinct perspectives. Firstly, note that the Bellman equationn for the value function can be recast in a more compact form using Eqs. 3.1.4 and 3.1.5 as

$$V^\pi(s) = \mathcal{R}^\pi_s + \gamma \sum_{s'} \mathcal{P}^\pi_{ss'} V^\pi(s') \tag{3.4.1}$$

or in vector form

$$\boldsymbol{V}^\pi = \boldsymbol{\mathcal{R}}^\pi + \gamma \mathcal{P}^\pi \boldsymbol{V}^\pi \tag{3.4.2}$$

which is analogous to the vector Bellman equation Eq. 2.2.5 for a MRP defined by $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi \rangle$. This equivalent MRP always follows the policy π.

Analogously, the Bellman equation for the state-action value can be written as

$$Q^\pi(s,a) = \mathcal{R}^a_s + \gamma \sum_{s',a'} \mathcal{P}^{aa'}_{ss'} Q^\pi(s',a') \tag{3.4.3}$$

with $\mathcal{P}^{aa'}_{ss'} = P(s',a'|s,a) = \mathcal{P}^a_{ss'}\pi(a'|s')$. This illustrates the fact that a MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$ is just a MRP of the form $\langle \mathcal{S} \times \mathcal{A}, \mathcal{R}, \mathcal{P}\,\Pi \rangle$ with state space $\mathcal{S} \times \mathcal{A}$, so that the states are pairs $(s,a) \in \mathcal{S} \times \mathcal{A}$ and the transition probability can be denoted by $\mathcal{P}\,\Pi$ (more on this in the following). The "state" value function is thus $Q(s,a)$ and the transition probability given by $\mathcal{P}^{aa'}_{ss'}$. Also, note that the expressions in terms of k-transition probabilities, Eqs. 3.3.8 and 3.3.9, are analogous to Eq. 2.3.5.

One can also represent Q in vector form, with each pair (s_i, a_i) corresponding to an element i, with $\boldsymbol{Q}_i = Q^\pi(s_i, a_i)$ and $\boldsymbol{\mathcal{R}}_i = \mathcal{R}^{a_i}_{s_i}$. Define the operator Π such that

$$\Pi(s', (s,a)) = \pi(a|s)\delta(s'-s) \tag{3.4.4}$$

Then we can write $\boldsymbol{V}^\pi = \Pi \boldsymbol{Q}^\pi$, since $[\Pi \boldsymbol{Q}^\pi](s) = \sum_{s',a'} \pi(a'|s')\delta(s-s')Q(s',a') = \sum_a \pi(a|s)Q^\pi(s,a) = V^\pi(s)$. Also, $[\mathcal{P}\Pi]((s',a'),(s,a)) = \mathcal{P}^{aa'}_{ss'} = \mathcal{P}^a_{ss'}\pi(a'|s')$ (or equivalently $[\mathcal{P}\Pi]_{ij} = P^\pi(s_j, a_j|s_i, a_i)$), $\boldsymbol{\mathcal{R}}^\pi = \Pi\boldsymbol{\mathcal{R}}$ and $\mathcal{P}^\pi = \Pi\mathcal{P}$, with the operator $\mathcal{P}$ having components $\mathcal{P}^a_{ss'}$. With all the above, we can write the Bellman equation for Q in vector form

$$\boldsymbol{Q}^\pi = \boldsymbol{\mathcal{R}} + \gamma \mathcal{P}\,\Pi\,\boldsymbol{Q}^\pi = \boldsymbol{\mathcal{R}} + \gamma \mathcal{P}\,\boldsymbol{V}^\pi \tag{3.4.5}$$

Note that this has the same form than Eq. 2.2.5, with $\boldsymbol{Q}^\pi$ playing the role of the value function vector, with transition probability operator $\mathcal{P}\,\Pi$. Also, acting Π on the left we recover the Bellman equation for $\boldsymbol{V}^\pi$. The above identities show how the two MRPs, $\langle \mathcal{S} \times \mathcal{A}, \mathcal{R}, \mathcal{P}\,\Pi \rangle$ and $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi \rangle$, are related, with Π connecting one MRP space to the other.

Finally, the above vector equations have the following formal solutions

$$\boldsymbol{V}^\pi = (I - \gamma \mathcal{P}^\pi)^{-1}\,\boldsymbol{\mathcal{R}}^\pi \tag{3.4.6}$$

$$\boldsymbol{Q}^\pi = (I - \gamma \mathcal{P}\,\Pi)^{-1}\,\boldsymbol{\mathcal{R}} \tag{3.4.7}$$

3.5 Expectation values

Expectations in an MDP must condition on both the policy π and the transition dynamics. The key quantities are the expected value of a function of a future state $f(S_{t+k})$, and more generally of a transition $(S_{t+k}, A_{t+k}, S_{t+k+1})$, starting from a given state or state-action pair. These generalize the MRP expectations of Eq. 1.4.5 by averaging over both the stochastic actions and the stochastic transitions. Specifically, for MDPs we have

$$\mathbb{E}[f(S_{t+k})|S_t = s] = \sum_{s'} P^\pi_k(s'|s) f(s') \tag{3.5.1}$$

$$\mathbb{E}[f(S_{t+k}, A_{t+k}, S_{t+k+1})|S_t = s, A_t = a] = \sum_{s',a's''} \mathcal{P}^{\pi}_{s's''} P^{\pi}_k(s', a'|s, a) f(s', a', s'') \tag{3.5.2}$$

we can apply the above equation to the reward $R_{t+k+1} = \mathcal{R}^{A_{t+k}}_{S_{t+k}S_{t+k+1}}$:

$$\mathbb{E}[R_{t+k+1}|S_t = s, A_t = a] = \mathbb{E}[\mathcal{R}^{A_{t+k}}_{S_{t+k}S_{t+k+1}}|S_t = s, A_t = a] = \sum_{s',a'} P^{\pi}_k(s', a'|s, a)\mathcal{R}^{a'}_{s'} = \sum_{s'} P^{\pi}_k(s'|s, a)\mathcal{R}^{\pi}_{s'} \tag{3.5.3}$$

where the last equality only stands for $k > 0$. From the above we can recover the expressions for Q and V, Eqs. 3.3.8 and 3.3.9. Also, integrating Eq. 3.5.2 we can recover Eq. 3.5.1.

We can compute the expectation of the TD error δ_t. Applying Eq. 3.5.2

$$\mathbb{E}[\delta_{t+k}|S_t = s, A_t = a] = \mathbb{E}[\mathcal{R}^{A_{t+k}}_{S_{t+k}S_{t+k+1}} + \gamma V^{\pi}(S_{t+k+1}) - V^{\pi}(S_{t+k})|S_t = s, A_t = a] \tag{3.5.4}$$

$$= \sum_{s',a's''} \mathcal{P}^{\pi}_{s's''} P^{\pi}_k(s', a'|s, a) \left[\mathcal{R}^{a'}_{s's''} + \gamma V^{\pi}(s'') - V^{\pi}(s')\right] \tag{3.5.5}$$

$$= \sum_{s'a'} P^{\pi}_k(s', a'|s, a) \left[\mathcal{R}^{a'}_{s'} + \gamma \sum_{s''} \mathcal{P}^{a'}_{s's''} V(s'') - V^{\pi}(s')\right] \tag{3.5.6}$$

Here we have to distinguish between $k = 0$ and $k > 0$. When evaluated at $k = 0$, we get $\mathbb{E}[\delta_t|s_t, a_t] = \mathcal{R}^a_s + \gamma \sum_{s''} \mathcal{P}^a_{ss''} V(s'') - V^{\pi}(s) = Q^{\pi}(s, a) - V^{\pi}(s) = A^{\pi}(s, a)$, the advantage. However, for $k > 0$, note that the term between square brackets vanishes due to the Bellman equation for the quality, and thus $\mathbb{E}[\delta_{t+k}|s_t, a_t] = 0$ for $k > 0$.

$$Var[\delta_t|s_t, a_t] = \mathbb{E}[\delta^2_t|s_t, a_t] - \mathbb{E}[\delta_t|s_t, a_t]^2 = \mathbb{E}[(R_{t+1} + \gamma V_{t+1})^2|s_t, a_t] - Q^2_t \tag{3.5.7}$$

3.6 Examples of policies

In the following we show some typical policy forms that are used accross multiple RL algorithms.

- **Random uniform policy**: selects an action randomly from the set of available actions, sampled from an uniform distribution.

$$\pi(a|s) = \frac{1}{|\mathcal{A}|} \tag{3.6.1}$$

  where $|\mathcal{A}|$ is the number of available actions.

- **Greedy policy**: always selects the action with highest estimated value.

$$\pi(a|s) = \mathbb{1}\left[a = \arg\max_a Q(s, a)\right] \tag{3.6.2}$$

  where $\mathbb{1}[x]$ is 1 when the expression x holds, being 0 otherwise.

- **ϵ-greedy policy**: with probability ϵ selects a random action (exploration), otherwise acts greedily (exploitation).

$$\pi(a|s) = \frac{\epsilon}{|\mathcal{A}|} + (1 - \epsilon)\, \pi_{greedy}(a|s) \tag{3.6.3}$$

  where $|\mathcal{A}|$ is the number of available actions and $\epsilon \in [0, 1]$ controls the exploration rate. it is a weighted average of the random uniform and the greedy policy: with $\epsilon = 0$ it reduces to the previous greedy policy, while $\epsilon = 1$ leads to the uniform one.

- **Deterministic policy**: the policy is not stochastic and only depends on the state s by a function $\mu(s)$.

$$\pi(a|s) = \mathbb{1}[a = \mu(s)] \tag{3.6.4}$$

- **Boltzmann (softmax) policy**: assigns probabilities proportional to exponentiated action values, with inverse temperature $\beta > 0$ controlling the sharpness of the distribution.

$$\pi(a|s) = \frac{e^{\beta Q(s,a)}}{Z_Q(s,\beta)} = \frac{e^{\beta A(s,a)}}{Z_A(s,\beta)} \tag{3.6.5}$$

with $Z_X(s,\beta) = \sum_a \exp[\beta X(s,a)]$. As $\beta \to \infty$ the policy becomes greedy; as $\beta \to 0$ it becomes uniform random. As we shall see in Sec. 7.7, this form of the policy or similar shapes naturally arise in an optimization problem with entropy or KL divergence regularization.

The gradient of this policy is given by

$$\nabla_\theta \log \pi(a|s) = \beta \left(\nabla_\theta Q(s,a) - \sum_{a'} \pi(a'|s) \nabla_\theta Q(s,a') \right) = \beta \left(\nabla_\theta Q(s,a) - \mathbb{E}_{a' \sim \pi}[\nabla_\theta Q(s,a')] \right) \tag{3.6.6}$$

or equivalently with $A(s,a)$ instead of $Q(s,a)$. Assuming that Q is independent of β, the value function can be written as

$$V(s) = \sum_a \pi(a|s) Q(s,a) = \sum_a \frac{1}{Z_Q(s,\beta)} \frac{\partial}{\partial_\beta} e^{\beta Q(s,a)} = \partial_\beta \log Z_Q(s,\beta) \tag{3.6.7}$$

Then one can show that its derivative gives the variance of Q: $\partial_\beta V(s) = \mathbb{E}_{a\sim\pi}[Q(s,a)^2] - V(s)^2$. If we further assume that V is independent of β, integrating, $Z_Q(s,\beta) = \exp(\beta V(s))$, $Z_A = 1$ and $\pi(a|s) = \exp[\beta A(s,a)]$. Also, $\nabla_\theta \log \pi(a|s) = \beta \nabla_\theta A(s,a)$, and thus the advantage is compatible to the policy, with the same parameters.

- **Gaussian policy**: used for continuous action spaces. The policy is parameterized by a mean $\mu_\theta(s)$ and standard deviation $\sigma_\theta(s)$, both outputs of a neural network.

$$\pi_\theta(a|s) = \mathcal{N}\big(a \,\big|\, \mu_\theta(s),\, \sigma_\theta^2(s)\big) = \frac{1}{\sqrt{2\pi\sigma_\theta^2(s)}} \exp\left(-\frac{(a - \mu_\theta(s))^2}{2\,\sigma_\theta^2(s)} \right) \tag{3.6.8}$$

Actions are sampled as $a = \mu_\theta(s) + \sigma_\theta(s)\,\xi$, with $\xi \sim \mathcal{N}(0,1)$ (reparameterization trick). Similarly for multi action space with a multivariate gaussian.

- **Visit count policy**: used in Monte Carlo Tree Search (MCTS) in AlphaGo Zero [4] and similar algorithms to improve the predicted policy. Let $N(s,a)$ be the number of visits to a state s and taking action s during some simulation. Highly frequently visited state-action pairs are more likely to be chosen, representing an average strategy from state-action counts

$$\pi(a|s) = \frac{N(s,a)^{1/\tau}}{\sum_{a'} N(s,a')^{1/\tau}} \tag{3.6.9}$$

where τ acts as a temperature: higher values lead to more random actions.

3.7 Optimal policy

A policy π is *optimal* if it achieves the highest possible expected return from every state simultaneously. It can be shown that for any finite MDP there always exists at least one deterministic optimal policy. We define the optimal value functions as the supremum over all policies:

$$V^*(s) = \max_\pi V^\pi(s), \qquad Q^*(s,a) = \max_\pi Q^\pi(s,a) \tag{3.7.1}$$

and the policy that attains these is π^*, so $V^*(s) = V^{\pi^*}(s)$, $Q^*(s,a) = Q^{\pi^*}(s,a)$.

The two optimal value functions are related by

$$V^*(s) = \max_a Q^*(s,a) \tag{3.7.2}$$

since the best we can do from state s is to pick the best action and then act optimally thereafter. Given Q^*, the optimal (deterministic) policy simply acts greedily:

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_a Q^*(s,a) \\ 0 & \text{otherwise} \end{cases} \qquad \Longrightarrow \qquad a^* = \pi^*(s) = \arg\max_a \, Q^*(s,a) \tag{3.7.3}$$

The optimal value functions satisfy a nonlinear version of the Bellman equations obtained by replacing the policy average with a max. These are known as the Bellman optimality equations:

$$V^*(s) = \max_a \left[\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a \, V^*(s') \right] \tag{3.7.4}$$

$$Q^*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a \, \max_{a'} Q^*(s',a') \tag{3.7.5}$$

Unlike the linear Bellman expectation equations for a fixed π, these are nonlinear due to the max operator, and generally have no closed-form solution. For finite MDPs they have a unique solution (V^*, Q^*), which can be found by dynamic programming methods (value iteration, policy iteration) or by model-free RL algorithms such as Q-learning, as shall be discussed in the following sections.

4 Classical RL Methods

4.1 Dynamic programming

Dynamic programming (DP) assumes full knowledge of the MDP (transition model $p(s'|s,a)$ and reward $r(s,a,s')$). It solves the Bellman equations iteratively to compute optimal policies.

- **Policy evaluation**: given a fixed policy π, compute its value function V^π by iterating the Bellman expectation backup until convergence:

$$V_{k+1}(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \left[\mathcal{R}_{ss'}^a + \gamma \, V_k(s') \right] \tag{4.1.1}$$

  As $k \to \infty$, it can be shown that it converges to the on-policy value function, $V_k \to V^\pi$, which satisfies the Bellman expectation equation:

$$V_\infty^\pi(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \left[\mathcal{R}_{ss'}^a + \gamma \, V_\infty^\pi(s') \right] \tag{4.1.2}$$

- **Policy improvement**: given V^π, construct a better policy by acting greedily with respect to the action-value function Q^π:

$$Q^\pi(s,a) = \sum_{s'} p(s'|s,a) \left[r(s,a,s') + \gamma \, V^\pi(s') \right] \tag{4.1.3}$$

$$\pi'(s) = \arg\max_a Q^\pi(s,a) \tag{4.1.4}$$

The *policy improvement theorem* guarantees $V^{\pi'}(s) \geq V^\pi(s)$ for all s.

- **Policy iteration**: alternate between policy evaluation and policy improvement until the policy stabilises:

$$\pi_0 \xrightarrow{\text{eval}} V^{\pi_0} \xrightarrow{\text{improve}} \pi_1 \xrightarrow{\text{eval}} V^{\pi_1} \xrightarrow{\text{improve}} \cdots \longrightarrow \pi^*, V^* \tag{4.1.5}$$

  Convergence to the optimal policy π^* is guaranteed in a finite number of iterations for finite MDPs.

- **Value iteration**: instead of waiting for full convergence of policy evaluation, apply a single Bellman *optimality* backup at each step:

$$V_{k+1}(s) = \max_a \sum_{s'} p(s'|s,a)\left[r(s,a,s') + \gamma\, V_k(s')\right] \tag{4.1.6}$$

  As $k \to \infty$, $V_k \to V^*$. The optimal policy is then recovered greedily:

$$\pi^*(s) = \arg\max_a \sum_{s'} p(s'|s,a)\left[r(s,a,s') + \gamma\, V^*(s')\right] \tag{4.1.7}$$

  Value iteration can be seen as policy iteration with policy evaluation truncated to a single sweep.

4.2 Monte Carlo and Temporal Difference

Monte Carlo (MC) and Temporal Difference (TD) methods estimate value functions directly from sampled experience, without requiring a model of the MDP. They differ in what they use as the learning target.

4.2.1 Monte Carlo prediction

In MC methods, the agent runs a full episode to completion and uses the actual discounted return G_t as the update target:

$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big) \tag{4.2.1}$$

This is equivalent to maintaining a running sample average: after $N(s)$ visits to state s, $V(s) \approx \frac{1}{N(s)} \sum_{i=1}^{N(s)} G_t^{(i)}$. Because G_t is the true return, the MC target is an unbiased estimator of $V^\pi(S_t)$. However, variance is high since $G_t = \sum_{k=0}^{T-t} \gamma^k R_{t+k+1}$ accumulates noise from every future step, and MC requires complete episodes.

4.2.2 Temporal Difference prediction

TD methods *bootstrap*: rather than waiting for G_t, they substitute the unknown future with the current value estimate. The TD(0) update after transition $(S_t, A_t, R_{t+1}, S_{t+1})$ is:

$$V(S_t) \leftarrow V(S_t) + \alpha\, \delta_t \tag{4.2.2}$$

where δ_t is the *TD error*:

$$\delta_t = R_{t+1} + \gamma\, V(S_{t+1}) - V(S_t) \tag{4.2.3}$$

The TD target $R_{t+1} + \gamma V(S_{t+1})$ replaces the full return with a one-step look-ahead. This introduces bias (since $V \neq V^\pi$ during learning) but greatly reduces variance. TD can also learn online, without waiting for the episode to end.

The TD error is the Bellman residual: $\mathbb{E}_\pi[\delta_t \mid S_t] = V^\pi(S_t) - V(S_t)$, which is zero when $V = V^\pi$. Conditioned on (S_t, A_t), its expectation equals the advantage:

$$\begin{aligned}
\mathbb{E}_\pi[\delta_t^\pi \mid S_t = s, A_t = a] &= \mathbb{E}_\pi[R_{t+1} + \gamma\, V^\pi(S_{t+1}) \mid S_t = s, A_t = a] - V^\pi(s) \\
&= \mathcal{R}_s^a + \gamma\, \mathbb{E}_\pi[V^\pi(S_{t+1}) \mid S_t = s, A_t = a] - V^\pi(s) \\
&= Q^\pi(s,a) - V^\pi(s) = \mathcal{A}^\pi(s,a)
\end{aligned} \tag{4.2.4}$$

	Monte Carlo ($n \to \infty$)	**TD(0)** ($n = 1$)
Target	G_t	$R_{t+1} + \gamma V(S_{t+1})$
Bias	None	Yes (bootstrapping)
Variance	High	Low
Requires complete episode	Yes	No
Online learning	No	Yes

Table 1: Comparison between Monte Carlo and TD(0) prediction.

4.2.3 n-step returns

MC and TD(0) are the two extremes of a family parameterised by the horizon n. The n-step return uses n actual rewards before bootstrapping:

$$G_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k R_{t+k+1} + \gamma^n V(S_{t+n}) \tag{4.2.5}$$

For $n = 1$ this is the TD(0) target; as $n \to \infty$ the bootstrap term vanishes and $G_t^{(\infty)} = G_t$ (MC). The update is $V(S_t) \leftarrow V(S_t) + \alpha(G_t^{(n)} - V(S_t))$. Increasing n reduces bias at the cost of higher variance.

4.2.4 TD(λ) and eligibility traces

Instead of committing to a fixed horizon n, TD(λ) forms the λ-*return*, a geometric weighted average over all n-step returns:

$$G_t^\lambda = (1-\lambda) \sum_{n=0}^{\infty} \lambda^n \, G_t^{(n+1)}, \qquad \lambda \in [0, 1]. \tag{4.2.6}$$

The factor $(1-\lambda)$ normalises the weights, which decay exponentially with n. At the extremes: $\lambda = 0$ recovers TD(0) (all weight on the one-step return), and $\lambda = 1$ recovers MC (all weight on the full return G_t). The update is

$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t^\lambda - V(S_t)\big). \tag{4.2.7}$$

This *forward view* requires the full future trajectory before updating, which is impractical online. The equivalent *backward view* implements the same updates incrementally using *eligibility traces* $e_t(s)$, which accumulate credit for recently visited states:

$$e_t(s) = \gamma\lambda \, e_{t-1}(s) + \mathbf{1}[S_t = s]. \tag{4.2.8}$$

At each step, every state is updated in proportion to its eligibility:

$$V(s) \leftarrow V(s) + \alpha \, \delta_t \, e_t(s), \qquad \forall s. \tag{4.2.9}$$

States visited frequently or recently carry higher traces, so the TD error δ_t is broadcast back in time appropriately. The two views are equivalent for the offline (end-of-episode) case and approximately equivalent online.

Note that TD(λ) is closely related to the Generalized Advantage Estimator discussed in Section 4.4.

4.3 Temporal difference algorithms

Temporal difference (TD) methods learn directly from experience without a model of the MDP, combining the sampling advantage of Monte Carlo with the bootstrapping of dynamic programming. Rather than waiting for an episode to end to compute a return, TD methods update value estimates after every step using the Bellman equation as a learning target. The key algorithmic question is whether to bootstrap from the current policy's own value estimate (on-policy, SARSA) or from the greedy value (off-policy, Q-learning).

The TD target for a transition (s, a, r, s') combines the immediate reward with a discounted estimate of the future:

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{4.3.1}$$

where δ_t is the *TD error*.

There are two important TD methods to cover: SARSA and Q-learning.

- **SARSA** (on-policy TD control): updates Q using the action a' actually taken by the current policy π in s'. Uses the quintuple $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \tag{4.3.2}$$

  Since $a_{t+1} \sim \pi(\cdot|s_{t+1})$, SARSA evaluates and improves the same policy that generates the data (on-policy).

- **Q-learning** (off-policy TD control): updates Q using the *greedy* action in s', regardless of what the behaviour policy actually does:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right] \tag{4.3.3}$$

  Q-learning directly approximates Q^* independently of the policy being followed (off-policy), provided all state-action pairs are visited sufficiently often.

	SARSA	**Q-learning**
Policy	On-policy	Off-policy
TD target	$r + \gamma Q(s', a'),\ a' \sim \pi$	$r + \gamma \max_{a'} Q(s', a')$
Convergence	Q^π	Q^*

Table 2: Comparison between SARSA and Q-learning.

4.4 Generalized Advantage Estimator

As shall be seen in Sec. 5.1, the advantage is extensively used in policy gradient methods. Therefore, it is important to fing robust estimators with low variance. Similarly to what we did in Sec. 4.2 with the n-step return, we can estimate the advantage using k steps

$$A_t^{(k)} = \sum_{l=0}^{k-1} \gamma^l \delta_{t+l} \tag{4.4.1}$$

For example $A_t^{(1)} = \delta_t$, $A_t^{(2)} = \delta_t + \gamma\delta_{t+1}$, $A_t^{(3)} = \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2}$. Note that there is a telescopic sum in the $A_t^{(k)}$, which can be casted in terms of the n-step returns:

$$A_t^{(k)} = \sum_{l=0}^{k-1} \gamma^l \delta_{t+l} = \sum_{l=0}^{k-1} \gamma^l R_{t+l+1} + \gamma^n V(S_{t+n}) - V(S_t) = G_t^{(k)} - V(S_t) \tag{4.4.2}$$

The Generalized Advantage Estimator (GAE), proposed by Schulman et al. 2016 [5], generalizes this by summing all the steps weighted by a parameter λ:

$$A_t^{GAE} = \sum_{k=0}^{\infty} (\gamma\lambda)^k \delta_{t+k} = (1-\lambda) \sum_{k=0}^{\infty} \lambda^k A_t^{(k+1)} \tag{4.4.3}$$

The advantages thus fulfill the following recurrence relation:

$$A_t^{GAE} = \delta_t + \gamma\lambda A_{t+1}^{GAE} \tag{4.4.4}$$

The GAE is then a geometric mixture of all n-step advantages which bootstraps(uses the value function) at every intermediate step, not just at the end. Note that this is pretty related to TD(λ). First, we have that $A_t^{GAE} = G_t^\lambda - V(S_t)$. When $\lambda = 0$, we get $A_t^{GAE} = \delta_t$, TD(0) (lowest variance, highest bias), while with $\lambda = 1$ we get after a telescopic sum $A_t^{GAE} = G_t - V(S_t)$, MC (lowest bias, highest variance).

If the value function is known, the GAE is unbiased. To show that, note that the following expectation vanishes:

$$\mathbb{E}_\pi[\delta_{t+1}|S_t = s, A_t = a] = \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^\pi_{s'} + \gamma \sum_{s''} \mathcal{P}^\pi_{s's''} V^\pi(s'') - V^\pi(s') \right] = 0 \tag{4.4.5}$$

This cancellation happens for all $t + k$ times with $k > 0$, as has been demonstrated from Eq. 3.5.6. Then,

$$\mathbb{E}_\pi[A_t^{GAE} \mid S_t = s, A_t = a] = \sum_{k=0}^{\infty} (\gamma\lambda)^k \mathbb{E}_\pi[\delta_{t+k} \mid S_t = s, A_t = a] \tag{4.4.6}$$

$$= \mathbb{E}_\pi[\delta_t \mid S_t = s, A_t = a] \tag{4.4.7}$$

$$= \mathcal{A}(S_t, A_t) \tag{4.4.8}$$

where the last equality uses Eq. (4.2.4), showing that the GAE is an unbiased estimator of the advantage. Notice however that when using value function approximation (with an NN e.g.) there is a bias introduced if $V^\phi \neq V^\pi$.

We can estimate the variance of the GAE following a similar argument to Sec. 2.5. For the sake of estimating the scaling, lets assume a constant variance of the TD error $\mathrm{Var}[\delta_t] = \sigma_\delta^2$ and a covariance between different times given by $\mathrm{Cov}[\delta_t, \delta_{t+k}] = \rho_\delta^k \sigma_\delta^2$, so that closer times are more correlated. Then:

$$\mathrm{Var}[A_t^{GAE}] = \sum_{k=0}^{\infty} (\gamma\lambda)^{2k} \sigma_\delta^2 + 2 \sum_{k=0, k<l}^{\infty} (\gamma\lambda)^{k+l} \rho_\delta^{l-k} \sigma_\delta^2 = \frac{\sigma_\delta^2}{1 - (\gamma\lambda)^2} \frac{1 + \gamma\lambda\rho_\delta}{1 - \gamma\lambda\rho_\delta} \tag{4.4.9}$$

where we have followed the steps from Sec. 2.5. Note that the higher λ (closer to MC), the higher the factors $1/(1 - (\gamma\lambda)^2)$ and $(1 + \gamma\lambda\rho)/(1 - \gamma\lambda\rho)$ separately, and thus higher the variance of the GAE.

The GAE can be interpreted as potential based reward shaping, which is discussed in detail in Sec. 9.2. If we define an alternative MPD with reward $\tilde{R}_{t+1} = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ with potential V and a discount factor given by $\gamma\lambda$, then the new return reads $\tilde{G}_t = \sum_{k=0}^\infty (\gamma\lambda)^k \tilde{R}_{t+k+1} = \sum_{k=0}^\infty (\gamma\lambda)^k \delta_{t+k} = A_t^{GAE}$, so the GAE can be interpreted as the reward-shaping return with potential V.

Practical implementations of the GAE such as the `skrl` [6] and `rsl_rl` [7] libraries compute the advantage backwards from $A_t^{GAE} = \delta_t + \gamma\lambda A_{t+1}^{GAE}$, using the boundary condition $A_n^{GAE} = 0$ being n the rollout length, since the bootstrap is already included through $V(S_n)$ in the TD errors (equivalent to $G_n^\lambda = V(S_n)$). Then one gets the returns from the identity $G_t^\lambda = A_t^{GAE} + V(S_t)$, which are used as the target in the MSE loss for the value function approximation.

5 Stochastic Policy Gradient Methods

Instead of learning the value functions V or Q, we can directly approximate the policy π with a neural network or other method. Policy gradient methods directly optimize the parameters θ of a parameterized policy π_θ by ascending the gradient of the expected return $J(\theta) = \mathbb{E}_{\pi_\theta}[G_0]$. Unlike value-based methods, which derive a policy implicitly from a value function, policy gradient methods work in policy space and can handle continuous action spaces and stochastic policies naturally. The key challenge is that $J(\theta)$ depends on θ both through the policy and through the distribution over states and trajectories induced by that policy, making a direct gradient computation seemingly intractable. The Policy Gradient Theorem resolves this by providing a tractable expression for $\nabla_\theta J(\theta)$ that can be estimated from sampled trajectories.

5.1 Policy Gradient Theorem

For proving the theorem, here we mostly follow the Spinning Up [3] derivation. We describe a trajectory $\tau = (s_0, a_0, s_1, a_1, ..., s_T, a_T)$, and $\tau_{t:} = (s_{t+1}, a_{t+1}, ..., s_T, a_T)$. The trajectory probability under policy π_θ is

$$P_\theta(\tau) = \rho_0(s_0) \prod_{t=0}^{\infty} \pi_\theta(a_t|s_t) P(s_{t+1}|s_t, a_t) \tag{5.1.1}$$

The total return of a trajectory is given by:

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t \mathcal{R}^{a_t}_{s_t s_{t+1}} \tag{5.1.2}$$

In policy gradient methods we want to maximize the expected value of the total return over trajectories, which is given by the following objective function:

$$J(\theta) = \mathbb{E}_{\tau \sim P_\theta}[R(\tau)] = \int P_\theta(\tau) R(\tau) d\tau \tag{5.1.3}$$

Note that the objective function can be related to the usual value functions as $J(\theta) = \sum_s \rho_0(s) V^\pi(s) = \sum_{s,a} \rho_0(s) \pi(a|s) Q^\pi(s,a)$.

We start by differentiating the objective

$$\nabla_\theta J(\theta) = \int \nabla_\theta P_\theta(\tau) R(\tau) d\tau \tag{5.1.4}$$

We can use the log-derivative trick $\nabla_\theta P_\theta(\tau) = P_\theta(\tau) \nabla_\theta \log P_\theta(\tau)$, so:

$$\nabla_\theta J(\theta) = \int P_\theta(\tau) \nabla_\theta \log P_\theta(\tau) R(\tau) d\tau = \mathbb{E}_{\tau \sim P_\theta} \left[\nabla_\theta \log P_\theta(\tau) R(\tau) \right] \tag{5.1.5}$$

Lets expand trajectory log-probability

$$\log P_\theta(\tau) = \log \rho_0(s_0) + \sum_{t=0}^{\infty} \left(\log \pi_\theta(a_t|s_t) + \log P(s_{t+1}|s_t, a_t) \right) \tag{5.1.6}$$

Taking the gradient, Given that the environment dynamics and initial state do not depend on θ, we can remove those terms independent of θ:

$$\nabla_\theta \log P_\theta(\tau) = \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t) \tag{5.1.7}$$

Substitute back and exchange sum and expectation

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim P_\theta} \left[\left(\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t) \right) R(\tau) \right] = \sum_{t=0}^{\infty} \mathbb{E}_{\tau \sim P_\theta} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau) \right] \tag{5.1.8}$$

The above result is what vanilla REINFORCE algorithm use. But we can improve upon it. Substitute $R(\tau) = \sum_{t=0}^{\infty} \gamma^t \mathcal{R}^{a_t}_{s_t s_{t+1}}$

$$\nabla_\theta J(\theta) = \sum_{t=0}^{\infty} \sum_{k=0}^{\infty} \gamma^k \mathbb{E}_{\tau \sim P_\theta} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \mathcal{R}^{a_k}_{s_k s_{k+1}} \right] \tag{5.1.9}$$

The previous expression includes terms for k lower than t which should not be relevant for the objective at t (only future rewards should count, no matter how it ended there). We will show that these terms vanish.[1] Note that the expression in the expectation only depends on $s_t, a_t, s_k, a_k, s_{k+1}$. Thus, the sampling $\tau \sim P_\theta$ is sampling over these variables only.

[1] This part is extracted from here.

Using the law of total expectation: $\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]]$ (Appendix A.1),

$$\mathop{\mathbb{E}}_{s_t,a_t,s_k,a_k,s_{k+1}}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\mathcal{R}^{a_k}_{s_k s_{k+1}}\right] = \mathop{\mathbb{E}}_{s_k,a_k,s_{k+1}}\left[\mathop{\mathbb{E}}_{s_t,a_t}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\mathcal{R}^{a_k}_{s_k s_{k+1}}|s_k,a_k,s_{k+1}\right]\right] \tag{5.1.10}$$

$$= \mathop{\mathbb{E}}_{s_k,a_k,s_{k+1}}\left[\mathcal{R}^{a_k}_{s_k s_{k+1}}\mathop{\mathbb{E}}_{s_t,a_t}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)|s_k,a_k,s_{k+1}\right]\right] \tag{5.1.11}$$

If $k < t$, the innermost expectation is 0 by the Expected Grad-Log-Prob (EGPL) Lemma $\mathbb{E}_{x\sim P_\theta}[\nabla_\theta \log P_\theta(x)] = 0$ (Appendix A.2). Thus, we can start the sum in k from t, and then replace full return $R(\tau)$ with the reward-to-go or future return G_t:

$$\nabla_\theta J(\theta) = \sum_{t=0}^{\infty}\mathbb{E}_{\tau\sim P_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\sum_{k=t}^{\infty}\gamma^k\mathcal{R}^{a_k}_{s_k s_{k+1}}\right] = \sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)G_t\right] \tag{5.1.12}$$

where we get the factor γ^t from the change of the dummy index. Using the law of total expectation again: $\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]]$ (Appendix A.1),

$$\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)G_t\right] = \mathbb{E}_{\pi_\theta}\left[\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)G_t|s_t,a_t\right]\right] \tag{5.1.13}$$

$$= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\mathbb{E}_{\pi_\theta}\left[G_t|s_t,a_t\right]\right] \tag{5.1.14}$$

$$= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)Q^{\pi_\theta}(s_t,a_t)\right] \tag{5.1.15}$$

And hence we can convert the gradient of the objective function to use the state-action expectation

$$\nabla_\theta J(\theta) = \sum_{t=0}^{\infty}\gamma^t\,\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)Q^{\pi}(s_t,a_t)\right] \tag{5.1.16}$$

Note that, due the EGPL lemma (Appendix A.2), we can subtract from $Q^\pi(s_t,a_t)$ a baseline term $b(s)$ which only depends on s, and leaves the policy gradient unchanged:

$$\mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\,b(s)] = \mathbb{E}_s\big[b(s)\,\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}[\nabla_\theta \log \pi_\theta(a|s)]\big] = 0 \tag{5.1.17}$$

where the inner expectation vanishes by the lemma applied to the distribution $\pi_\theta(\cdot|s)$ over actions.

As shown in Sec. 5.2, adding a baseline can reduce the variance of the policy gradient estimator, and the value function $V^\pi(s)$ is a near optimal choice for that. Thus, it is convenient to substract the value function $V^\pi(s)$, so that the final expression appears as function of the advantage $A^\pi(s,a)$. This is our final result for the Policy Gradient Theorem:

$$\boxed{\nabla_\theta J(\theta) = \sum_{t=0}^{\infty}\gamma^t\,\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)A^{\pi}(s_t,a_t)\right]} \tag{5.1.18}$$

Vanilla policy gradient methods like REINFORCE estimate the above gradient of the objective to update the parameters of the parameterized policy, either with estimated Monte Carlo returns or with advantages. One usually computes the advantage with some estimator $\hat{A}$, commonly the GAE discussed in Sec. 4.4.

Finally, note that the above result involves an expectation over states and actions at time t. We can write the probability to be in state s and action a at time t as $P(S_t = s, A_t = a) = \sum_{s'} P_t^\pi(s|s')\rho_0(s')$. Thus, the policy gradient theorem can also be expressed as

$$\nabla_\theta J(\theta) = \sum_{t=0}^{\infty}\gamma^t \mathop{\mathbb{E}}_{s,a\sim P_t^\pi}\left[\nabla_\theta \log \pi_\theta(a|s)A^{\pi}(s,a)\right] \tag{5.1.19}$$

$$= \sum_{t=0}^{\infty}\gamma^t \sum_{s,a}\nabla_\theta \log \pi_\theta(a|s)A^{\pi}(s,a)P^\pi(S_t = s, A_t = a) \tag{5.1.20}$$

$$= \sum_{s,a}\nabla_\theta \log \pi_\theta(a|s)A^{\pi}(s,a)\sum_{t=0}^{\infty}\gamma^t \sum_{s'}P_t^\pi(s|s')\rho_0(s') \tag{5.1.21}$$

In this section we have proven the theorem through a classic demonstration based on the probability of the trajectory. In follow up sections we will provide equivalent and more concise proofs of this theorem.

5.2 Variance reduction with baselines

Although subtracting any state-dependent baseline $b(s)$ leaves the gradient unbiased, it does change the variance. Write the single-sample estimator as

$$\hat{g}_b = \nabla_\theta \log \pi_\theta(a|s)\,(Q^\pi(s,a) - b(s)), \tag{5.2.1}$$

and let $h = \nabla_\theta \log \pi_\theta(a|s)$. Because $\mathbb{E}_{a\sim\pi}[h] = 0$ by the EGLP lemma, the mean of $\hat{g}_b$ equals the mean of $\hat{g}_0 = hQ^\pi$:

$$\mathbb{E}_{a\sim\pi}[\hat{g}_b] = \mathbb{E}_{a\sim\pi}[hQ^\pi] - b(s)\underbrace{\mathbb{E}_{a\sim\pi}[h]}_{=0} = \nabla_\theta J(\theta). \tag{5.2.2}$$

The second moment of the estimator is

$$\mathbb{E}_{a\sim\pi}\left[\|\hat{g}_b\|^2\right] = \mathbb{E}_{a\sim\pi}\left[\|h\|^2(Q^\pi - b)^2\right] = \mathbb{E}[\|h\|^2 Q^{\pi\,2}] - 2b\,\mathbb{E}[\|h\|^2 Q^\pi] + b^2\,\mathbb{E}[\|h\|^2]. \tag{5.2.3}$$

The variance is given by $Var(\hat{g}_b) = \mathbb{E}[\|\hat{g}_b\|^2] - \mathbb{E}[\|\hat{g}_b\|]^2$. Since $\mathbb{E}[\|\hat{g}_b\|]^2 = \|\nabla_\theta J\|^2$ does not depend on b, minimising $\mathbb{E}[\|\hat{g}_b\|^2]$ over b is equivalent to minimising the variance. The expression above is a quadratic in b, so we can differentiate with respect to b and equal to 0 to find the minimum b^*:

$$\boxed{b^*(s) = \frac{\mathbb{E}_{a\sim\pi}\left[\|\nabla_\theta \log \pi_\theta(a|s)\|^2\, Q^\pi(s,a)\right]}{\mathbb{E}_{a\sim\pi}[\|\nabla_\theta \log \pi_\theta(a|s)\|^2]}.} \tag{5.2.4}$$

This is precisely the solution to a weighted least squares regression of $Q^\pi(s,a)$ onto a constant with weights $\|\nabla_\theta \log \pi_\theta(a|s)\|^2$. The optimal baseline is a weighted average of Q^π across actions, with actions that produce large policy gradient norms receiving more weight. The value function $V^\pi(s) = \mathbb{E}_{a\sim\pi}[Q^\pi(s,a)]$ corresponds to the unweighted case and is the standard practical choice: it is easy to learn and reduces variance substantially, even if it is not exactly optimal.

5.3 Policy gradient through functional derivatives

We can get the same expressions for the policy gradient theorem by a more direct method, using the closed forms solutions for the value functions, occupancy measures and functional derivatives (see Appendix A.4 for a summary of the relevant rules). First, the gradient of the objective wrt the parameters of the policy can be written in terms of the functional derivative of the policy as

$$\nabla_\theta J^\pi = \sum_{s,a} \frac{\delta J^\pi}{\delta \pi(a|s)} \nabla_\theta \pi(a|s) \tag{5.3.1}$$

The expression $J^\pi = \sum_s \rho_0(s) V^\pi(s)$ can be written in vector form as the dot product $J^\pi = \boldsymbol{\rho}_0 \cdot \boldsymbol{V}^\pi$ we can apply the functional derivative, getting $\frac{\delta J^\pi}{\delta \pi(a|s)} = \boldsymbol{\rho}_0 \cdot \frac{\delta \boldsymbol{V}^\pi}{\delta \pi(a|s)}$.

Then we can apply the functional derivative to the closed form through the operator equation, Eq. 3.4.2. Using the derivative for operators $d(A^{-1}) = -A^{-1}\,dA\,A^{-1}$,[2] we get

$$\frac{\delta \boldsymbol{V}^\pi}{\delta \pi(a|s)} = (I - \gamma\mathcal{P}^\pi)^{-1}\,\gamma \frac{\delta \mathcal{P}^\pi}{\delta \pi(a|s)}\,(I - \gamma\mathcal{P}^\pi)^{-1}\,\boldsymbol{\mathcal{R}}^\pi + (I - \gamma\mathcal{P}^\pi)^{-1}\,\frac{\delta \boldsymbol{\mathcal{R}}^\pi}{\delta \pi(a|s)} \tag{5.3.2}$$

$$= (I - \gamma\mathcal{P}^\pi)^{-1}\left[\gamma \frac{\delta \mathcal{P}^\pi}{\delta \pi(a|s)} \boldsymbol{V}^\pi + \frac{\delta \boldsymbol{\mathcal{R}}^\pi}{\delta \pi(a|s)}\right] \tag{5.3.3}$$

Note then that, taking into account the identity $\delta f(x)/\delta f(y) = \delta(x-y)$ (Appendix A.4), the functional derivative of $\mathcal{P}^\pi_{s's''} = \sum_{a'} \pi(a'|s')\mathcal{P}^{a'}_{s's''}$ and $\mathcal{R}^\pi_{s'} = \sum_{a'} \pi(a'|s')\mathcal{R}^{a'}_{s'}$ are

[2] Which is proven by deriving $A^{-1}A = I$

$$\frac{\delta \mathcal{P}^{\pi}_{s's''}}{\delta \pi(a|s)} = \mathcal{P}^{a'}_{s's''}\delta(s-s') \;\text{ and }\; \frac{\delta \mathcal{R}^{\pi}_{s'}}{\delta \pi(a|s)} = \mathcal{R}^{a}_{s'}\delta(s-s') \tag{5.3.4}$$

and thus

$$\frac{\delta V^{\pi}(s')}{\delta \pi(a|s)} = \sum_{s''}\left[(I-\gamma\mathcal{P}^{\pi})^{-1}\right]_{s's''}\left[\mathcal{R}^{a}_{s''} + \gamma\sum_{s'''}\mathcal{P}^{a}_{s''s'''}V^{\pi}(s''')\right]\delta(s-s'') \tag{5.3.5}$$

$$= \left[(I-\gamma\mathcal{P}^{\pi})^{-1}\right]_{s's} Q^{\pi}(s,a) \tag{5.3.6}$$

$$= \sum_{t=0}^{\infty}\gamma^{t}P_t(s|s')Q^{\pi}(s,a) \tag{5.3.7}$$

And putting this result into Eq. 5.3.1 we finally recover the policy gradient theorem

$$\nabla_{\theta}J^{\pi} = \sum_{s,a}\nabla_{\theta}\pi(a|s)Q^{\pi}(s,a)\sum_{s'}\sum_{t=0}^{\infty}\gamma^{t}P_t(s|s')\rho_0(s') = \sum_{t=0}^{\infty}\gamma^{t}\mathop{\mathbb{E}}_{s,a\sim P^{\pi}_t}\left[\nabla_{\theta}\log\pi(a|s)Q^{\pi}(s,a)\right] \tag{5.3.8}$$

Note that this derivation is much shorter than the one presented in Sec. 5.1 and does not involve recursion relations in the previous section.

5.4 Actor-Critic

Actor-critic methods combine a parameterized policy (the *actor*, π_θ) with a learned value function (the *critic*, V_ϕ). The actor is updated by ascending the policy gradient using advantage estimates provided by the critic; the critic is updated by minimizing a regression loss to bootstrapped return targets. This separates the roles of *how to act* (actor) and *how good is a state* (critic), and is strictly more sample-efficient than pure Monte Carlo policy gradient because the critic's bootstrapped targets reduce variance without waiting for episode termination.

The update rules at each step are:

$$\phi \;\leftarrow\; \phi - \alpha_c\,\nabla_{\phi}\big(G^{\lambda}_t - V_{\phi}(S_t)\big)^2 \tag{5.4.1}$$

$$\theta \;\leftarrow\; \theta + \alpha_a\,A^{\mathrm{GAE}}_t\,\nabla_{\theta}\log\pi_{\theta}(A_t|S_t) \tag{5.4.2}$$

where G^{λ}_t are the λ-returns used as regression targets for the critic (Section 4.4), and $A^{\mathrm{GAE}}_t = G^{\lambda}_t - V_\phi(S_t)$ is the advantage estimate fed to the actor. The GAE parameter λ interpolates between a high-bias/low-variance critic-only baseline ($\lambda = 0$) and a low-bias/high-variance Monte Carlo advantage ($\lambda = 1$).

In practice, actor and critic share a common trunk network whose features are split into a policy head (outputting action distribution parameters) and a value head (outputting a scalar $V_\phi(s)$). Training the two heads jointly with a combined loss

$$\mathcal{L}(\theta,\phi) = -\mathcal{L}_{\mathrm{actor}}(\theta) + c_v\,\mathcal{L}_{\mathrm{critic}}(\phi) - c_e\,H[\pi_\theta] \tag{5.4.3}$$

where $H[\pi_\theta]$ is a policy entropy bonus that encourages exploration, is the standard recipe used in A2C, PPO, and most modern on-policy algorithms.

6 Occupancy measures formulation

An equivalent way to formulate the RL problem is in terms of occupancy measures, which allows for very clean derivations. In this section we detail this approach and prove the policy gradient theorem in an alternative procedure.

6.1 Occupancy measure

The *state-action occupancy measure* of a policy π is the $(1-\gamma)$-normalised discounted frequency with which π visits each state-action pair:

$$\rho^\pi(s,a) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P^\pi(s_t = s,\, a_t = a). \tag{6.1.1}$$

where $P^\pi(s_t = s,\, a_t = a) = \sum_{s_0 \ldots s_{t-1}, a_0 \ldots a_{t-1}} P^\pi(s_0, a_0, s_1, a_1, ..., s_t = s,\, s_t = a)$. the $1-\gamma$ term is a normalisation factor that ensures ρ^π sums to 1 across all state-action pairs, so it can be interpreted as a probability distribution over $\mathcal{S}\times\mathcal{A}$. In the undiscounted case, this tends to the stationary distribution, if such exists.

The expected return J is proven to be a linear functional of ρ^π. Using the law of total expectation and exchanging the expectation and the time sum gives

$$J^\pi = \mathbb{E}_\pi[G_0] = \sum_{t=0}^{\infty}\gamma^t \mathbb{E}_\pi\left[R_{t+1}\right] = \sum_{t=0}^{\infty}\gamma^t \mathop{\mathbb{E}}_{s,a\sim P_t^\pi}\left[\mathbb{E}_\pi[R_{t+1}|S_t = s, A_t = a]\right] \tag{6.1.2}$$

$$= \sum_{t=0}^{\infty}\gamma^t \mathop{\mathbb{E}}_{s,a\sim P_t^\pi}[\mathcal{R}_s^a] = \sum_{t=0}^{\infty}\gamma^t \sum_{s,a} P^\pi(S_t = s,\, A_t = a)\,\mathcal{R}_s^a \tag{6.1.3}$$

$$= \frac{1}{(1-\gamma)}\sum_{s,a}\rho^\pi(s,a)\,\mathcal{R}_s^a = \frac{1}{(1-\gamma)}\mathop{\mathbb{E}}_{s,a\sim\rho^\pi}[\mathcal{R}_s^a] \tag{6.1.4}$$

Because J is linear in ρ^π for fixed rewards, optimising over policies is equivalent to optimising over occupancy measures.

The occupancy measure fulfills a recurrence relation due to probability conservation. Unfolding the occupancy measure definition, using $\rho_0(s,a) = P(S_0 = s, A_0 = a) = \rho_0(s)\pi(a|s)$ and the Chapman-Kolmogorov equation and reindexing gives

$$\rho^\pi(s,a) = (1-\gamma)P(S_0 = s, A_0 = a) + \sum_{t=1}^{\infty}\gamma^t P(S_t = s, A_t = a) \tag{6.1.5}$$

$$= (1-\gamma)\rho_0(s,a) + \sum_{t=1}^{\infty}\gamma^t \sum_{s',a'} P(s,a|s',a')P(S_{t-1} = s', A_{t-1} = a') \tag{6.1.6}$$

$$= (1-\gamma)\rho_0(s,a) + \gamma\sum_{s',a'}\sum_{t=0}^{\infty}\gamma^t P(s,a|s',a')P(S_t = s', A_t = a') \tag{6.1.7}$$

$$= (1-\gamma)\rho_0(s,a) + \gamma\sum_{s',a'} P(s,a|s',a')\rho(s',a') \tag{6.1.8}$$

$$\tag{6.1.9}$$

This is the *Bellman flow constraint*, and shows that probability mass must be conserved at every state: the discounted frequency of visiting s' must equal the frequency of having transitioned into it from any (s,a) plus the probability of starting there under ρ_0.

We can also define the state occupancy measure, or marginal state visitation, $\rho^\pi(s) = \sum_a \rho^\pi(s,a)$. Conversely, any $\rho^\pi \geq 0$ satisfying (6.1.9) corresponds to a valid policy, recovered by

$$\pi(a|s) = \frac{\rho^\pi(s,a)}{\rho^\pi(s)} \tag{6.1.10}$$

And thus we can factor $\rho^\pi(s,a) = \rho^\pi(s)\,\pi(a|s)$. The marginal state visitation satisfies the marginalised flow constraint

$$\rho^\pi(s) = (1-\gamma)\,\rho_0(s) + \gamma\sum_{s',a'}\mathcal{P}_{s's}^{a'}\,\pi(a'|s')\,\rho^\pi(s') \tag{6.1.11}$$

or in vector form

$$\boldsymbol{\rho}^\pi = (1-\gamma)\boldsymbol{\rho}_0 + \gamma \boldsymbol{\rho}^\pi \mathcal{P}^\pi \tag{6.1.12}$$

which can be formally solved to give in vector form

$$\boldsymbol{\rho}^\pi = (1-\gamma)\boldsymbol{\rho}_0 \left(I - \gamma \mathcal{P}^\pi\right)^{-1} \tag{6.1.13}$$

similar to Eq. 3.4.2. Note that the $\mathcal{P}^\pi$ operator acts on the right, as opposite to acting on the left with the value function $\boldsymbol{V}^\pi$.

For any function $f(s)$, we can take the dot product of Eq. 6.1.12 with $\boldsymbol{f}$, getting the identity [8]

$$0 = (1-\gamma)\boldsymbol{\rho}_0 \cdot \boldsymbol{f} + \boldsymbol{\rho}^\pi \cdot (\gamma \mathcal{P}^\pi \boldsymbol{f} - \boldsymbol{f}) \tag{6.1.14}$$

For instance, for $V^\pi(s)$ and using Eq. 3.4.2 and $J^\pi = \boldsymbol{\rho}_0 \cdot \boldsymbol{V}^\pi$ we get $(1-\gamma)\, J^\pi = \boldsymbol{\rho}^\pi \cdot (\boldsymbol{V}^\pi - \gamma \mathcal{P}^\pi \boldsymbol{V}^\pi) = \boldsymbol{\rho}^\pi \cdot \boldsymbol{\mathcal{R}}^\pi$, which is Eq. 6.1.4 in vector form.

6.2 Value functions in terms of occupancy measures

We can also express the state and state-action value functions using a occupancy measure formulation. First, note that we can write $P^\pi(s_0 = s, a_0 = a, s_t = s', a_t = a') = P_t^\pi(s', a'|s, a)\rho_0(s, a)$. This allows us to define the *state-action occupancy measure* of a policy π in s', a' starting from s, a as

$$\rho(s, a, s', a') = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t P^\pi(s_0 = s, a_0 = a, s_t = s', a_t = a') = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t P_t^\pi(s', a'|s, a)\rho_0(s, a) \tag{6.2.1}$$

The marginal probabilities are just $\rho(s, s', a') = \sum_a \rho(s, a, s', a')$ and $\rho(s', a') = \sum_{s,a} \rho(s, a, s', a')$, reducing to the occupancy measure. We can then write the occupancy measure in terms of the k-step transitions.

$$\rho^\pi(s, a) = (1-\gamma)\sum_{k=0}^{\infty} \gamma^k \sum_{s',a'} P_k^\pi(s, a|s', a')\pi(a'|s')\rho_0(s') \tag{6.2.2}$$

and

$$\rho^\pi(s) = (1-\gamma)\sum_{k=0}^{\infty} \gamma^k \sum_{s'} P_k^\pi(s|s')\rho_0(s') \tag{6.2.3}$$

Then,

$$Q^\pi(s, a) = \frac{1}{(1-\gamma)} \sum_{s',a'} \frac{\rho(s, a, s', a')}{\rho_0(s)\pi(a|s)} \mathcal{R}_{s'}^{a'} \tag{6.2.4}$$

and

$$V^\pi(s) = \frac{1}{(1-\gamma)} \sum_{s',a'} \frac{\rho(s, s', a')}{\rho_0(s)} \mathcal{R}_{s'}^{a'} \tag{6.2.5}$$

Note that writing the occupancy measure distributions in terms of the k-step transitions, we recover our well known forms Eqs. 3.3.8 and 3.3.9. Integrating again we recover $J = \frac{1}{(1-\gamma)} \sum_{s',a'} \rho(s', a')\mathcal{R}_{s'}^{a'}$.

6.3 Policy gradient in terms of occupancy

We can derive the policy gradient theorem in terms of occupancy measures. First, note that the derivative of J will be expressed in terms of the gradient of the occupancy measure, which can be computed using the marginalized flow constraint (Eq. 6.1.11). We get a recursion relation that we can unfold, obtaining:

$$\nabla\rho^\pi(s,a) = \rho^\pi(s)\nabla\pi(a|s) + \pi(a|s)\,\gamma\sum_{s',a'}\mathcal{P}^{a'}_{s's}[\rho(s')\nabla\pi(a'|s') + \pi(a'|s')\nabla\rho(s')] \tag{6.3.1}$$

$$= \rho^\pi(s)\nabla\pi(a|s) + \gamma\sum_{s',a'}\rho(s')\nabla\pi(a'|s')\mathcal{P}^{a'}_{s's}\pi(a|s) \tag{6.3.2}$$

$$+\ \gamma^2\sum_{s'',a''}\rho(s'')\nabla\pi(a''|s'')\sum_{s',a'}\mathcal{P}^{a''}_{s''s'}\mathcal{P}^{a'}_{s's}\pi(a'|s')\pi(a|s) + ... \tag{6.3.3}$$

$$= \sum_{k=0}^{\infty}\gamma^k\sum_{s',a'}P_k(s,a|s',a')\rho^\pi(s')\nabla\pi(a'|s') \tag{6.3.4}$$

which is a expression in terms of the k-step transition probabilities $P_k(s|s',a')$. Then, the gradient of the objective J can be computed using the previous gradient, rearranging terms and applying the $Q(s,a)$ expression from Eq. 3.3.8:

$$(1-\gamma)\,\nabla_\theta J(\theta) = \sum_{s,a}\mathcal{R}^a_s\,\nabla\rho^\pi(s,a) \tag{6.3.5}$$

$$= \sum_{s,a}\mathcal{R}^a_s\sum_{k=0}^{\infty}\gamma^k\sum_{s',a'}P_k(s,a|s',a')\rho^\pi(s')\nabla\pi(a'|s') \tag{6.3.6}$$

$$= \sum_{s',a'}\rho^\pi(s')\nabla\pi(a'|s')\sum_{k=0}^{\infty}\gamma^k\sum_{s}P_k(s,a|s',a')\mathcal{R}^a_s \tag{6.3.7}$$

$$= \sum_{s',a'}\rho^\pi(s')\nabla\pi(a'|s')\,Q^\pi(s',a') \tag{6.3.8}$$

which we can further rearrange to get the policy gradient formula:

$$\boxed{\nabla_\theta J(\theta) = \frac{1}{(1-\gamma)}\sum_{s,a}\rho^{\pi_\theta}(s,a)\,\nabla_\theta\log\pi_\theta(a|s)\,Q^{\pi_\theta}(s,a) = \frac{1}{(1-\gamma)}\mathop{\mathbb{E}}_{s,a\sim\rho^{\pi_\theta}}\left[\nabla_\theta\log\pi_\theta(a|s)\,Q^{\pi_\theta}(s,a)\right]} \tag{6.3.9}$$

The occupancy-based form has a clear interpretation: $Q^{\pi_\theta}(s,a)$ acts as the functional derivative of J with respect to the policy density at (s,a), weighted by how often state s is visited. Subtracting a state-dependent baseline $b(s)$ from Q^{π_θ} still works here since it leaves the gradient unchanged. Finally, note that substituting ρ by its definition allows us to recover the expression derived in Sec. 5.1.

$$\nabla_\theta J(\theta) = \sum_{t=0}^{\infty}\gamma^t\mathop{\mathbb{E}}_{s,a\sim P^\pi_t}\left[\nabla_\theta\log\pi_\theta(a|s)\,Q^{\pi_\theta}(s,a)\right] \tag{6.3.10}$$

6.4 Policy gradient with occupancy through functional derivatives

In the same line that Sec. 5.3, we can prove the policy gradient theorem from closed vector forms for the value functions, occupancy measures and functional derivatives.

From the expression $(1-\gamma)J^\pi = \sum_{s,a}\rho^\pi(s)\pi(a|s)\,\mathcal{R}^a_s$ and using $\delta f(x)/\delta f(y) = \delta(x-y)$, we can apply the functional derivative, getting

$$(1-\gamma)\frac{\delta J^\pi}{\delta\pi(a|s)} = \sum_{s',a'}\left(\frac{\delta\pi(a'|s')}{\delta\pi(a|s)}\rho^\pi(s') + \frac{\delta\rho^\pi(s')}{\delta\pi(a|s)}\pi(a'|s')\right)\mathcal{R}^{a'}_{s'} = \rho^\pi(s)\mathcal{R}^a_s + \sum_{s'}\frac{\delta\rho^\pi(s')}{\delta\pi(a|s)}\mathcal{R}^\pi_{s'} \tag{6.4.1}$$

The last term can be written in vector form as the dot product $\sum_{s'} \frac{\delta \rho^\pi(s')}{\delta \pi(a|s)} \mathcal{R}^\pi_{s'} = \frac{\delta \boldsymbol{\rho}^\pi}{\delta \pi(a|s)} \cdot \boldsymbol{\mathcal{R}}^\pi$. Then we can apply the functional derivative to the closed form through the operator equation, Eq. 6.1.13. Using the derivative for operators $d[(I-A)^{-1}] = (I-A)^{-1}\, dA\, (I-A)^{-1}$, we get

$$\frac{\delta \boldsymbol{\rho}^\pi}{\delta \pi(a|s)} = (1-\gamma)\boldsymbol{\rho}_0 \left(I - \gamma \mathcal{P}^\pi\right)^{-1} \gamma \frac{\delta \mathcal{P}^\pi}{\delta \pi(a|s)} \left(I - \gamma \mathcal{P}^\pi\right)^{-1} = \gamma \boldsymbol{\rho}^\pi \frac{\delta \mathcal{P}^\pi}{\delta \pi(a|s)} \left(I - \gamma \mathcal{P}^\pi\right)^{-1} \tag{6.4.2}$$

Then using the vector form of V, Eq. 3.4.2, the dot product reads

$$\frac{\delta \boldsymbol{\rho}^\pi}{\delta \pi(a|s)} \cdot \boldsymbol{\mathcal{R}}^\pi = \gamma \boldsymbol{\rho}^\pi \frac{\delta \mathcal{P}^\pi}{\delta \pi(a|s)} \left(I - \gamma \mathcal{P}^\pi\right)^{-1} \boldsymbol{\mathcal{R}}^\pi = \gamma \boldsymbol{\rho}^\pi \frac{\delta \mathcal{P}^\pi}{\delta \pi(a|s)} \boldsymbol{V}^\pi \tag{6.4.3}$$

Note then that, taking into account the property $\delta f(x)/\delta f(y) = \delta(x-y)$, the functional derivative of $\mathcal{P}^\pi_{s's''} = \sum_{a'} \pi(a'|s') \mathcal{P}^{a'}_{s's''}$ is

$$\frac{\delta \mathcal{P}^\pi_{s's''}}{\delta \pi(a|s)} = \mathcal{P}^{a'}_{s's''} \delta(s - s') \tag{6.4.4}$$

Thus, the functional derivative of the objective function reads

$$(1-\gamma)\frac{\delta J^\pi}{\delta \pi(a|s)} = \rho^\pi(s)\mathcal{R}^a_s + \gamma \sum_{s',s''} \rho^\pi(s') \frac{\delta \mathcal{P}^\pi_{s's''}}{\delta \pi(a|s)} V^\pi(s'') \tag{6.4.5}$$

$$= \rho^\pi(s) \left(\mathcal{R} + \gamma \sum_{s''} \mathcal{P}^{a'}_{ss''} V^\pi(s'') \right) \tag{6.4.6}$$

$$= \rho^\pi(s) Q^\pi(s,a) \tag{6.4.7}$$

And putting this result into Eq. 5.3.1 we finally recover the policy gradient theorem

$$(1-\gamma)\nabla_\theta J^\pi = \sum_{s,a} \rho^\pi(s) Q^\pi(s,a) \nabla_\theta \pi(a|s) = \sum_{s,a} \rho^\pi(s,a)\, \nabla_\theta \log \pi(a|s)\, Q^\pi(s,a) \tag{6.4.8}$$

Note that this derivation is much shorter than the one presented in Sec. 5.1 and does not involve recursion relations in the previous section.

6.5 Occupancy measure Lagrangian

There is an elegant way to derive the Bellman equation and the policy gradient theorem from a occupancy measure lagrangian with constraints, which connects to KKT conditions.

Essentially, we can write the RL problem as a constrained optimization over occupancy measures, aiming to maximize:

$$\max_\rho \sum_{s,a} \rho(s,a)\, \mathcal{R}^a_s \tag{6.5.1}$$

subject to the constraints given by the flow probability conservation (Eq. 6.1.9) and the condition $\rho(s,a) \geq 0$.

We introduce the dual variable $Q(s,a)$, which act as Lagrange multipliers for the flow constraints and for the non-negativity constraints. Thus, we can write the Lagrangian as

$$\mathcal{L}(\rho, Q) = \rho(s,a)\mathcal{R}^a_s + Q(s,a) \left[(1-\gamma)\rho_0(s,a) + \gamma \sum_{s',a'} P(s,a|s',a')\rho(s',a') - \rho(s,a) \right] \tag{6.5.2}$$

with the action given by $\mathcal{S} = \sum_{s,a} \mathcal{L}(\rho(s,a), Q(s,a))$.

Thus, if we apply the derivative with respect to $Q(s,a)$ and make it equal to 0, $\partial \mathcal{L}(\rho,Q)/\partial Q(s,a) = 0$, we recover the flow constraint, Eq. 6.1.9. On the other hand, the Euler-Lagrange equation with respect to $\rho(s,a)$, $\partial \mathcal{L}(\rho,Q)/\partial \rho(s,a) = 0$, gives

$$\mathcal{R}_s^a + \gamma \sum_{s',a'} P(s',a'|s,a)Q(s',a') - Q(s,a) = 0 \implies Q(s,a) = \mathcal{R}_s^a + \gamma \sum_{s',a'} \pi(a'|s')\mathcal{P}_{ss'}^a Q(s',a') \tag{6.5.3}$$

where we have used $P(s',a'|s,a) = \pi(a'|s')P(s'|s,a) = \pi(a'|s')\mathcal{P}_{ss'}^a$. We thus recover the Bellman equation for Q.

7 Advanced policy gradient methods

Policy gradient methods estimate the gradient using on-policy trajectories and discard each batch after a single update. This is highly sample-inefficient. Both TRPO and PPO address this by reusing data collected under an older policy π_{old} via importance sampling, while controlling how far the updated policy π_θ may deviate from π_{old}.

7.1 Importance-sampled objective

Let π_{old} be the behaviour policy used to collect data. Define the importance sampling ratio

$$r_\theta(s,a) = \frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)}. \tag{7.1.1}$$

The importance-sampled surrogate objective rewrites the expected advantage under π_θ as an expectation under π_{old}:

$$\mathcal{L}_{\text{IS}}(\theta) = \mathbb{E}_{\pi_{\text{old}}}[r_\theta(s,a)\, A(s,a)] = \mathbb{E}_{\pi_\theta}[A(s,a)]\,. \tag{7.1.2}$$

The equality follows from the change-of-measure identity $\mathbb{E}_{\pi_{\text{old}}}[r_\theta f] = \mathbb{E}_{\pi_\theta}[f]$, an is an example of importance sampling. Since the gradient of r_θ satisfies

$$\nabla_\theta r_\theta(s,a) = r_\theta(s,a)\, \nabla_\theta \log \pi_\theta(a|s), \tag{7.1.3}$$

provided that the parameters of the old policy remain fixed, so $\mathcal{L}_{\text{IS}}$ and $J(\theta)$ share the same gradient:

$$\nabla_\theta \mathcal{L}_{\text{IS}}(\theta) = \mathbb{E}_{\pi_{\text{old}}}[r_\theta(s,a)\nabla_\theta \log \pi_\theta(a|s)\, A(s,a)] = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\, A(s,a)] = \nabla_\theta J(\theta). \tag{7.1.4}$$

This justifies using $\mathcal{L}_{\text{IS}}$ as a surrogate for $J(\theta)$. This surrogate loss is similar to using the Minorize-Maximization MM algorithm, which is an iterative algorithm which allows to use a lower bound function for J, approximating J at the current policy, and easier to optimize. In the following sections, we will see that this surrogate loss is not an heuristic choice, showing how it naturally arises from first principles.

7.2 Performance Difference Lemma

Here we proof a useful result known as the Performance Difference Lemma (PDL) [9], which states that the difference between performance of two policies can be casted as the expected value of the advantage of one policy evaluated at states and actions sampled from the another one. To exemplify the variety of approaches to RL, we provide 3 different proofs of the lemma: a standard one, a proof through occupancy measures and a proof through functional derivatives.

7.2.1 Standard proof

We start by the difference between the value functions $V^{\pi'}(s) - V^\pi(s)$. To prove the performance difference lemma, note that this identity is fulfilled: $\sum_{t=0}^\infty \gamma^t V^\pi(S_t) = V^\pi(S_0) + \sum_{t=1}^\infty \gamma^t V^\pi(S_t) = V^\pi(S_0) + \gamma \sum_{t=0}^\infty \gamma^t V^\pi(S_{t+1})$. Then, we first add and subtract the same quantity $\mathbb{E}_{\pi'}\left[\sum_{t=0}^\infty \gamma^t (V^\pi(S_t)\right]$, use the previously mentioned identity and use the definition of the TD error δ_t and its average given by the advantage, Eq. 4.2.4:

$$V^{\pi'}(s) - V^{\pi}(s) = V^{\pi'}(s) + \mathbb{E}_{\pi'}\left[\sum_{t=0}^{\infty}\gamma^t(V^{\pi}(S_t) - V^{\pi}(S_t))|S_t = s\right] - V^{\pi}(s) \tag{7.2.1}$$

$$= V^{\pi'}(s) + \mathbb{E}_{\pi'}\left[\sum_{t=0}^{\infty}\gamma^t(\gamma V^{\pi}(S_{t+1}) - V^{\pi}(S_t))|S_t = s\right] \tag{7.2.2}$$

$$= \sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{\pi'}\left[R_{t+1} + \gamma V^{\pi}(S_{t+1}) - V^{\pi}(S_t)|S_t = s\right] \tag{7.2.3}$$

$$= \sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{\pi'}\left[\delta_t^{\pi}|S_t = s\right] = \sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{\pi'}\left[\mathbb{E}_{\pi'}\left[\delta_t^{\pi} \mid S_t = s, A_t = a\right] \mid S_t = s\right] \tag{7.2.4}$$

$$= \sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{a\sim\pi'}\left[A^{\pi}(s,a) \mid S_t = s\right] \tag{7.2.5}$$

Integrating over $\rho_0(s)$ to write it down in terms of the objective $J^{\pi} = \sum_s \rho_0(s)V^{\pi}(s)$ we get:

$$\boxed{J^{\pi'} - J^{\pi} = \sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{\pi'}\left[A^{\pi}(s_t, a_t)\right]} \tag{7.2.6}$$

7.2.2 Using occupancy measure and recurrence relations

In this proof we also start by the difference between the value functions

$$V^{\pi'}(s) - V^{\pi}(s) = \sum_a \pi'(a|s)\left(V^{\pi'}(s) - V^{\pi}(s)\right) \tag{7.2.7}$$

$$= \sum_a \pi'(a|s)\left(V^{\pi'}(s) - Q^{\pi}(s,a) + Q^{\pi}(s,a) - V^{\pi}(s)\right) \tag{7.2.8}$$

$$= \sum_a \pi'(a|s)A^{\pi}(s,a) + \sum_a \pi'(a|s)\left(V^{\pi'}(s) - Q^{\pi}(s,a)\right) \tag{7.2.9}$$

$$= \sum_a \pi'(a|s)A^{\pi}(s,a) + \sum_{s'} \mathcal{P}_{ss'}^{\pi'}\left(V^{\pi'}(s') - V^{\pi'}(s')\right) \tag{7.2.10}$$

where we have used the Bellman equations. Note that we have found a recurrence relation for $V^{\pi'}(s) - V^{\pi}(s)$, which can be written in concise form identically to Eq. 3.3.9:

$$V^{\pi'}(s) - V^{\pi}(s) = \sum_{k=0}^{\infty}\gamma^k \sum_{s'} P_k^{\pi'}(s'|s) \sum_{a'} \pi'(a'|s')A^{\pi}(s',a') \tag{7.2.11}$$

Thus, using Eq. 6.2.3, given that $J^{\pi} = \sum_s \rho_0(s)V^{\pi}(s)$, we can integrate over $\rho_0(s)$ to get the difference between objectives J:

$$\boxed{J^{\pi'} - J^{\pi} = \frac{1}{(1-\gamma)}\sum_{s,a}\rho^{\pi'}(s)\pi'(a|s)A^{\pi}(s,a) = \frac{1}{(1-\gamma)}\mathop{\mathbb{E}}_{s,a\sim\rho^{\pi'}}[A^{\pi}(s,a)]} \tag{7.2.12}$$

7.2.3 Using vector forms

There is an alternative, more concise way to prove this result using vector forms. Consider the objective of π' as $(1-\gamma)J^{\pi'} = \boldsymbol{\rho}^{\pi'} \cdot \boldsymbol{\mathcal{R}}^{\pi'}$ and add the identity from Eq. 6.1.14 for π' with $\boldsymbol{f} = \boldsymbol{V}^{\pi}$, the value for the policy π:

$$(1-\gamma)J^{\pi'} = \boldsymbol{\rho}^{\pi'} \cdot \boldsymbol{\mathcal{R}}^{\pi'} = \boldsymbol{\rho}^{\pi'} \cdot \boldsymbol{\mathcal{R}}^{\pi'} + (1-\gamma)\boldsymbol{\rho}_0 \cdot \boldsymbol{V}^{\pi} + \boldsymbol{\rho}^{\pi'} \cdot \left(\gamma \mathcal{P}^{\pi'} \boldsymbol{V}^{\pi} - \boldsymbol{V}^{\pi}\right) \tag{7.2.13}$$

$$= (1-\gamma)J^{\pi} + \boldsymbol{\rho}^{\pi'} \cdot \left(\boldsymbol{\mathcal{R}}^{\pi'} + \gamma \mathcal{P}^{\pi'} \boldsymbol{V}^{\pi} - \boldsymbol{V}^{\pi}\right) \tag{7.2.14}$$

$$= (1-\gamma)J^{\pi} + \sum_{s,a} \rho^{\pi'}(s)\pi'(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V^{\pi}(s') - V^{\pi}(s)\right) \tag{7.2.15}$$

$$= (1-\gamma)J^{\pi} + \sum_{s,a} \rho^{\pi'}(s)\pi'(a|s)\, A^{\pi}(s,a) \tag{7.2.16}$$

Neat!

7.3 Surrogate loss and upper bound

The PDL allows us to compare the performance of two policies. In the practice, if π' represents a slightly modified policy with respect to the original one π, one can approximate the sampling distribution $\rho^{\pi'}$ by ρ^{π}, and thus we can write the performance metric of a modified policy π'

$$J^{\pi'} \simeq J^{\pi} + \frac{1}{(1-\gamma)} \mathop{\mathbb{E}}_{s\sim\rho^{\pi}, a\sim\pi'}[A^{\pi}(s,a)] = J^{\pi} + \frac{1}{(1-\gamma)} \sum_{s,a} \rho^{\pi}(s)\pi'(a|s)A^{\pi}(s,a) \tag{7.3.1}$$

This result has great practical importance. We can use it to gather data using policy π, sample states from $\rho^{\pi}(s)$ and compute the advantage $A^{\pi}(s,a)$, evaluating with the new policy π'. Thus, given the policy ratio $r(s,a) = \pi'(a|s)/\pi(a|s)$, we can define the loss wrt policy π'

$$L^{\pi'} = \sum_{s,a} \rho^{\pi}(s)\pi'(a|s)A^{\pi}(s,a) = \mathop{\mathbb{E}}_{s,a\sim\rho^{\pi}}[r(s,a)A^{\pi}(s,a)] \tag{7.3.2}$$

which is equivalent to the importance sampling loss given by Eq. 7.1.2, used as a basis for TRPO and PPO. Note that this surrogate is zero for $\pi' = \pi$ (meaning $J^{\pi'} = J^{\pi}$) and its gradient wrt θ equals the real gradient of $(1-\gamma)\nabla_{\theta} J^{\pi'}$ (for π weights fixed).

How good is the above approximation? Can we just substitute $\rho^{\pi'}$ by ρ^{π}? Papers such as [8] find lower and upper bounds on the objective difference. Here we provide a proof on an upper bound on that approximation. Write $\bar{A}^{\pi,\pi'} = \sum_a \pi'(a|s)A^{\pi}(s,a)$.

Consider now $Z^{\pi} = (I - \gamma P^{\pi})^{-1}$ and $\Delta P = P^{\pi'} - P^{\pi}$. Then, $(Z^{\pi})^{-1} - (Z^{\pi'})^{-1} = \gamma \Delta P$ and multiplying by $Z^{\pi'}$ and Z^{π} we get the relation

$$Z^{\pi'} = Z^{\pi} + \gamma Z^{\pi} \Delta P Z^{\pi'} \tag{7.3.3}$$

and solving we get

$$Z^{\pi'} = Z^{\pi} \left(I - \gamma \Delta P Z^{\pi}\right)^{-1} \tag{7.3.4}$$

Thus

$$\boldsymbol{\rho}^{\pi'} - \boldsymbol{\rho}^{\pi} = (1-\gamma)\boldsymbol{\rho}_0 (Z^{\pi'} - Z^{\pi}) \tag{7.3.5}$$

$$= (1-\gamma)\boldsymbol{\rho}_0 \gamma Z^{\pi} \Delta P Z^{\pi'} \tag{7.3.6}$$

$$= \gamma \boldsymbol{\rho}^{\pi} \Delta P Z^{\pi'} \tag{7.3.7}$$

$$= \gamma \boldsymbol{\rho}^{\pi} \Delta P Z^{\pi} \left(I - \gamma \Delta P Z^{\pi}\right)^{-1} \tag{7.3.8}$$

We can write in vector form the PDL as $(1-\gamma)(J^{\pi'} - J^{\pi}) = \boldsymbol{\rho}^{\pi'} \cdot \bar{\boldsymbol{A}}^{\pi,\pi'}$.

$$(1-\gamma)(J^{\pi'} - J^{\pi}) = \boldsymbol{\rho}^{\pi'} \cdot \bar{\boldsymbol{A}}^{\pi,\pi'} \tag{7.3.9}$$

$$= \boldsymbol{\rho}^{\pi} \cdot \bar{\boldsymbol{A}}^{\pi,\pi'} + \gamma \boldsymbol{\rho}^{\pi} \Delta P Z^{\pi} \left(I - \gamma \Delta P Z^{\pi}\right)^{-1} \bar{\boldsymbol{A}}^{\pi,\pi'} \tag{7.3.10}$$

$$= L^{\pi'} + \gamma \boldsymbol{\rho}^{\pi} \Delta P Z^{\pi} \left(I - \gamma \Delta P Z^{\pi}\right)^{-1} \bar{\boldsymbol{A}}^{\pi,\pi'} \tag{7.3.11}$$

This is an exact result which can be expanded in powers of $\gamma \Delta P Z^{\pi}$. We can compute an upper bound on the above expression. Using the Hölder's inequality $\mid \boldsymbol{f} \cdot \boldsymbol{g} \mid \leq \|\boldsymbol{f}\|_p \|\boldsymbol{g}\|_q$ with $1/p + 1/q = 1$. Then

$$\mid (1-\gamma)(J^{\pi'} - J^{\pi}) - L^{\pi'} \mid_1 \leq \gamma \|\boldsymbol{\rho}^{\pi}\|_1 \left\| \Delta P Z^{\pi} \left(I - \gamma \Delta P Z^{\pi}\right)^{-1} \bar{\boldsymbol{A}}^{\pi,\pi'} \right\|_{\infty} \tag{7.3.12}$$

We have $\|\boldsymbol{\rho}^{\pi}\|_1 = 1$. On the other hand,

$$\left\| \Delta P Z^{\pi} \left(I - \gamma \Delta P Z^{\pi}\right)^{-1} \bar{\boldsymbol{A}}^{\pi,\pi'} \right\|_{\infty} \leq \|\Delta P\|_{\infty} \|Z^{\pi}\|_{\infty} \left\| (I - \gamma \Delta P Z^{\pi})^{-1} \right\|_{\infty} \left\| \bar{\boldsymbol{A}}^{\pi,\pi'} \right\|_{\infty} \tag{7.3.13}$$

Now

$$\|Z^{\pi}\|_{\infty} = \left\| \sum_{k=0} \gamma^k (\mathcal{P}^{\pi})^k \right\|_{\infty} \leq \sum_{k=0} \gamma^k \left\| (\mathcal{P}^{\pi})^k \right\|_{\infty} \leq \sum_{k=0} \gamma^k = \frac{1}{1-\gamma} \tag{7.3.14}$$

Having $\Delta P_{ss'} = \sum_a \mathcal{P}^a_{ss'} (\pi'(a|s) - \pi(a|s))$, then

$$\|\Delta P\|_{\infty} \leq \sum_a \|\mathcal{P}\|_{\infty} \|\pi'(a|s) - \pi(a|s)\|_{\infty} \leq \sum_a \|\pi'(a|s) - \pi(a|s)\|_{\infty} = 2 D_{TV}(\pi, \pi') \tag{7.3.15}$$

where $D_{TV}(\pi, \pi') = \frac{1}{2} \sum_a \|\pi'(a|s) - \pi(a|s)\|_{\infty}$ is the total variation distance. On the other hand

$$\left\| (I - \gamma \Delta P Z^{\pi})^{-1} \right\|_{\infty} = \left\| \sum_{k=0}^{\infty} (\gamma \Delta P Z^{\pi})^k \right\|_{\infty} \leq \sum_{k=0}^{\infty} \|\gamma \Delta P Z^{\pi}\|_{\infty}^k \tag{7.3.16}$$

$$\leq \sum_{k=0}^{\infty} \gamma^k \|\Delta P\|_{\infty}^k \|Z^{\pi}\|_{\infty}^k = \sum_{k=0}^{\infty} \left(\frac{\gamma 2 D_{TV}(\pi, \pi')}{1-\gamma} \right)^k \tag{7.3.17}$$

$$= \frac{1}{1 - \left(\frac{\gamma 2 D_{TV}(\pi, \pi')}{1-\gamma} \right)} \tag{7.3.18}$$

And finally

$$\mid (1-\gamma)(J^{\pi'} - J^{\pi}) - L^{\pi'} \mid_1 \leq \frac{\gamma}{1-\gamma} \frac{2 D_{TV}(\pi, \pi')}{1 - \left(\frac{\gamma 2 D_{TV}(\pi, \pi')}{1-\gamma} \right)} \left\| \bar{\boldsymbol{A}}^{\pi,\pi'} \right\|_{\infty} \tag{7.3.19}$$

Thus, the above result bounds from above the error one can get when using the surrogate loss as an approximation. A lower bound can also be found using a similar procedure [8]. Through through Pinsker's inequality, one can relate the total variation distance with the KL divergence, which has better and smoother properties for differentiating:

$$D_{TV}(\pi, \pi') \leq \sqrt{\frac{1}{2} D_{KL}(\pi' || \pi)} \tag{7.3.20}$$

7.4 Natural Policy Gradient

Standard gradient ascent on $J(\theta)$ moves in the direction of steepest ascent per unit Euclidean step in parameter space. This choice is arbitrary: the Euclidean metric on θ has no intrinsic meaning, and small parameter changes can correspond to large or small changes in the policy distribution depending on the parameterization. The natural policy gradient [10] instead measures progress in the space of distributions, using the Fisher Information Matrix (FIM) as the metric.

We seek the step $\delta\theta$ that maximises the first-order improvement in J while keeping the KL divergence between the old and updated policies within a trust region of size ε:

$$\max_{\delta\theta} \nabla_\theta J(\theta)^\top \delta\theta \qquad \text{subject to} \qquad D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\theta+\delta\theta}) \le \varepsilon. \tag{7.4.1}$$

The Lagrangian of this constrained problem is

$$\mathcal{L}(\delta\theta, \lambda) = J(\theta + \delta\theta) - \lambda\Big(D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\theta+\delta\theta}) - \varepsilon\Big). \tag{7.4.2}$$

By the KL–FIM relation (Appendix A.3), for small $\delta\theta$ the KL divergence is approximated to second order as

$$D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\theta+\delta\theta}) \approx \tfrac{1}{2}\, \delta\theta^\top F(\theta)\, \delta\theta, \tag{7.4.3}$$

so the Lagrangian becomes

$$\mathcal{L}(\delta\theta, \lambda) \approx J(\theta) + \nabla_\theta J(\theta)^\top \delta\theta - \frac{\lambda}{2}\, \delta\theta^\top F(\theta)\, \delta\theta + \lambda\varepsilon. \tag{7.4.4}$$

Setting $\nabla_{\delta\theta}\mathcal{L} = 0$ gives the optimality condition

$$\nabla_\theta J(\theta) - \lambda\, F(\theta)\, \delta\theta^* = 0, \tag{7.4.5}$$

which yields

$$\delta\theta^* = \frac{1}{\lambda}\, F(\theta)^{-1} \nabla_\theta J(\theta). \tag{7.4.6}$$

Absorbing $1/\lambda$ into a learning rate α, the natural policy gradient update is

$$\boxed{\theta_{t+1} = \theta_t + \alpha\, F(\theta_t)^{-1} \nabla_\theta J(\theta_t).} \tag{7.4.7}$$

For a policy π_θ, the FIM is

$$F(\theta) = \mathbb{E}_{s\sim d^{\pi_\theta},\, a\sim\pi_\theta(\cdot|s)} \left[\nabla_\theta \log \pi_\theta(a|s)\, \nabla_\theta \log \pi_\theta(a|s)^\top\right], \tag{7.4.8}$$

where d^{π_θ} is the (discounted) state visitation distribution.

7.5 TRPO

Trust Region Policy Optimization (TRPO) [11] keeps the importance weights reliable by constraining the KL divergence between the old and new policies. The trust-region idea is grounded in the theoretical framework of [9], who showed that bounding the KL deviation between successive policies guarantees monotonic improvement. The KL divergence is defined as

$$D_{\mathrm{KL}}(P \,\|\, Q) = \mathbb{E}_{x\sim P}\left[\log \frac{P(x)}{Q(x)}\right] = \sum_x P(x) \log \frac{P(x)}{Q(x)} \tag{7.5.1}$$

and is a non-symmetric measure of how different are two distributions.

In TRPO, the optimisation problem is

$$\max_\theta \ \mathbb{E}_{\pi_{\mathrm{old}}}[r_\theta(s,a)\, A(s,a)] \quad \text{subject to} \quad \mathbb{E}_s[D_{\mathrm{KL}}(\pi_{\mathrm{old}}(\cdot|s) \,\|\, \pi_\theta(\cdot|s))] \le \delta. \tag{7.5.2}$$

The constraint δ defines a trust region in which the linear approximation to $J(\theta)$ provided by $\mathcal{L}_{\text{IS}}$ remains valid. TRPO solves this constrained problem using the conjugate gradient method to compute the natural gradient direction, followed by a backtracking line search to satisfy the constraint. While theoretically well-motivated and guaranteed to produce monotonic policy improvement, TRPO is a second-order method and is computationally expensive to implement at scale.

The natural gradient and TRPO solve the same constrained problem at each step, but differ in how they handle the feasibility of the computation. The natural gradient computes $F(\theta)^{-1}\nabla J$ exactly (or via compatible critics) and then takes a fixed step, without verifying that the KL constraint is satisfied globally. TRPO [11] replaces the exact inverse with a conjugate-gradient solve (a matrix-free $F^{-1}v$ product) and adds a backtracking line search to enforce the KL bound, making it applicable to large neural network policies where F cannot be formed or inverted.

There are extensions of this idea, such as the Constrained Policy Optimization (CPO) [8], which extends the TRPO framework to safe reinforcement learning by adding constraints on auxiliary cost signals alongside the trust-region constraint. See also elegant derivations of bounds in that paper.

7.6 PPO

Proximal Policy Optimization (PPO) [12] achieves a similar effect to TRPO with a much simpler first-order algorithm. Instead of a hard KL constraint, PPO clips the importance ratio to the interval $[1-\epsilon,\, 1+\epsilon]$:

$$\mathcal{L}_{\text{PPO}}(\theta) = \mathbb{E}_{\pi_{\text{old}}}\left[\min\big(r_\theta(s,a)\,A(s,a),\ \text{clip}\big(r_\theta(s,a),\,1-\epsilon,\,1+\epsilon\big)\,A(s,a)\big)\right]. \tag{7.6.1}$$

The min makes the objective a pessimistic lower bound: the clip removes the incentive to push r_θ outside $[1-\epsilon,\, 1+\epsilon]$, and taking the minimum ensures the clipped value only ever reduces the objective relative to the unclipped one. The effect is asymmetric depending on the sign of the advantage:

- If $A(s,a) > 0$ (beneficial action): r_θ is clipped above at $1+\epsilon$, preventing the policy from making the action arbitrarily more likely:

$$\mathcal{L}_{\text{PPO}}(\theta)\big|_{A>0} = \mathbb{E}_{\pi_{\text{old}}}\left[\min\big(r_\theta(s,a),\ 1+\epsilon\big)\,A(s,a)\right]. \tag{7.6.2}$$

- If $A(s,a) < 0$ (detrimental action): r_θ is clipped below at $1-\epsilon$, preventing the policy from suppressing the action too aggressively:

$$\mathcal{L}_{\text{PPO}}(\theta)\big|_{A<0} = \mathbb{E}_{\pi_{\text{old}}}\left[\max\big(r_\theta(s,a),\ 1-\epsilon\big)\,A(s,a)\right]. \tag{7.6.3}$$

In practice, the PPO objective is augmented with an entropy bonus and a value-function loss term, and several gradient steps are taken over the same batch before refreshing π_{old}.

7.7 Boltzmann policies from regularized optimization

Both entropy regularization and KL-constrained optimization (TRPO) lead to Boltzmann-type optimal policies via the same Lagrangian argument. The two cases differ only in the regularizer and its reference distribution.

7.7.1 Entropy regularization

Maximize expected return with an entropy bonus (temperature $\tau > 0$) subject to normalization:

$$\max_\pi \sum_a \pi(a|s)Q(s,a) + \tau\mathcal{H}(\pi(\cdot|s)), \qquad \mathcal{H}(\pi(\cdot|s)) = -\sum_a \pi(a|s)\log\pi(a|s) \tag{7.7.1}$$

The same procedure can also be performed with the advantage rather than Q. The Lagrangian with multiplier λ enforcing $\sum_a \pi(a|s) = 1$ is:

$$\mathcal{L} = \sum_a \pi(a|s)Q(s,a) - \tau\sum_a \pi(a|s)\log\pi(a|s) - \lambda\left(1 - \sum_a \pi(a|s)\right) \tag{7.7.2}$$

Setting $\partial\mathcal{L}/\partial\pi(a|s) = 0$ gives $Q(s,a) - \tau(1 + \log\pi(a|s)) - \lambda = 0$, and solving:

$$\boxed{\pi(a|s) = \frac{\exp(Q(s,a)/\tau)}{\sum_b \exp(Q(s,b)/\tau)}} \tag{7.7.3}$$

7.7.2 KL-constrained optimization

Maximize expected advantage subject to a KL trust region, as done in TRPO, enforced via a Lagrange multiplier $1/\tau$:

$$\mathcal{L} = \sum_a \pi(a|s)A(s,a) - \frac{1}{\tau}\sum_a \pi(a|s)\log\frac{\pi(a|s)}{\pi_{\text{old}}(a|s)} - \lambda\left(1 - \sum_a \pi(a|s)\right) \tag{7.7.4}$$

Setting $\partial\mathcal{L}/\partial\pi(a|s) = 0$ gives $A(s,a) - \frac{1}{\tau}(\log\pi(a|s) - \log\pi_{\text{old}}(a|s) + 1) - \lambda = 0$, and solving:

$$\boxed{\pi_{\text{new}}(a|s) = \frac{\pi_{\text{old}}(a|s)\exp(A(s,a)/\tau)}{\sum_b \pi_{\text{old}}(b|s)\exp(A(s,b)/\tau)}} \tag{7.7.5}$$

The entropy regularization result is recovered from here when π_{old} is the uniform distribution.

8 Deterministic Policy Gradient

Policy gradient methods such as PPO and TRPO parameterize stochastic policies and must integrate the policy gradient over the action distribution. For continuous action spaces one can instead learn a deterministic policy $\mu_\theta : \mathcal{S} \to \mathcal{A}$, which maps each state directly to an action. Silver et al. 2014 [13] set the theoretical foundations for this proving the Deterministic Policy Gradient (DPG) theorem.

8.1 Continuous formulation

Deterministic policies are specially relevant in continuous action spaces. We can write down all the relevant equations in continuous spaces by basically replacing the sums by integrals. For instance, the Bellman expectation equations take the form

$$Q^\pi(s,a) = \mathcal{R}_s^a + \gamma\int_{\mathcal{S}} ds'\,\mathcal{P}_{ss'}^a\,V^\pi(s') = \mathcal{R}_s^a + \gamma\int_{\mathcal{S},\mathcal{A}} ds'\,da'\,\mathcal{P}_{ss'}^a\,\pi(a'|s')Q^\pi(s',a') \tag{8.1.1}$$

$$V^\pi(s) = \int_{\mathcal{A}} da\,\pi(a|s)Q^\pi(s,a) = \int_{\mathcal{A}} da\,\pi(a|s)\left[\mathcal{R}_s^a + \gamma\int_{\mathcal{S}} ds'\,\mathcal{P}_{ss'}^a\,V^\pi(s')\right] \tag{8.1.2}$$

We can apply the same logic to the Policy Gradient Theorem. Using the occupancy measure version of the theorem, Eq. 6.3.9,, we write the aforementioned theorem in continuous form as

$$(1-\gamma)\nabla_\theta J(\theta) = \int ds\,da\,\rho^{\pi_\theta}(s,a)\,\nabla_\theta\log\pi_\theta(a|s)\,Q^{\pi_\theta}(s,a) \tag{8.1.3}$$

8.2 Deterministic Policy Gradient Theorem

The Deterministic Policy Gradient Theorem can be derived following an analogous proof to Sec. 6.3. But it can also be derived directly from the Stochastic Policy Gradient Theorem, as a special case where the policy is deterministic. For this proof, we start from the continuous occupancy measure version of the theorem , Eq. 8.1.3.

For a deterministic policy $\mu_\theta(s)$, we have $\pi(a|s) = \mathbb{1}(a = \mu_\theta(s))$, which in continuous case can be written as a Dirac delta, $\pi(a|s) = \delta_D(a - \mu_\theta(s))$. Then, the gradient of the policy can be written as

$$\nabla_\theta \pi(a|s) = \nabla_\theta \delta_D(a - \mu_\theta(s)) = \nabla_\theta \mu_\theta(s) \nabla_{\mu_\theta(s)} \delta_D(a - \mu_\theta(s)) = -\nabla_\theta \mu_\theta(s) \nabla_a \delta_D(a - \mu_\theta(s)) \tag{8.2.1}$$

where we have used the Dirac delta property $\partial_x \delta_D(x - y) = -\partial_y \delta_D(x - y)$ (which can be proven by integrating it).[3] Putting the above result into Eq. 8.1.3 gives

$$(1-\gamma)\nabla_\theta J(\theta) = \int ds\, da\, \rho^{\pi_\theta}(s)\, \nabla_\theta \pi_\theta(a|s)\, Q^{\pi_\theta}(s,a) \tag{8.2.2}$$

$$= \int ds\, \rho^{\pi_\theta}(s) \nabla_\theta \mu_\theta(s) \int da\, (-\nabla_a \delta_D(a - \mu_\theta(s))) Q(s,a) \tag{8.2.3}$$

$$= \int ds\, \rho^{\pi_\theta}(s) \nabla_\theta \mu_\theta(s) \int da\, \left[\nabla_a \left(-\delta_D(a - \mu_\theta(s)) Q(s,a)\right) + \delta_D(a - \mu_\theta(s)) \nabla_a Q(s,a)\right] \tag{8.2.4}$$

where we have integrated by parts. The first term vanishes, while after integrating the second we get the Deterministic Policy Gradient:

$$\boxed{(1-\gamma)\nabla_\theta J(\theta) = \int ds\, \rho^{\pi_\theta}(s) \nabla_\theta \mu_\theta(s) \nabla_a Q(s,a)\Big|_{a=\mu_\theta(s)} = \mathop{\mathbb{E}}_{s\sim\rho^\mu}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)\big|_{a=\mu_\theta(s)}\right]} \tag{8.2.5}$$

where we write ρ^μ instead of ρ^π to indicate that they are ruled by the deterministic policy μ.

8.3 DDPG

DDPG (Deep Deterministic Policy Gradient) [14] is an off-policy actor-critic algorithm that learns a deterministic policy. The idea is that we replace the computation expensive max operation of the optimal policy $\max_a Q(s,a)$ by $Q(s, \mu_\theta(s))$, where μ_θ is a deterministic policy parameterized by θ. Note that maximizing J with gradient ascent is thus equivalent to search for the μ_θ that maximizes $\max_a Q^\mu(s,a) = Q^\mu(s, \mu_\theta(s))$ for all s.

DDPG maintains an actor μ_θ and a critic Q_ϕ, together with slowly-updated target copies $\mu_{\theta'}$ and $Q_{\phi'}$. The critic is trained to minimize the mean-squared Bellman error over transitions (s, a, r, s') sampled from a replay buffer $\mathcal{D}$:

$$\mathcal{L}(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\big(Q_\phi(s,a) - y\big)^2\right], \qquad y = r + \gamma\, Q_{\phi'}(s',\, \mu_{\theta'}(s')). \tag{8.3.1}$$

Similarly to Q-learning and DQN, the target correspond to the optimal $Q(s,a)$, being thus off-policy, replacing $r + \max_a Q(s,a)$ by $r + Q(s, \mu(s))$. The target y is computed with the frozen target networks to prevent the moving-target instability that arises when the same network is used for both prediction and bootstrap. The actor is then updated by ascending the DPG gradient through the critic:

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{s\sim\mathcal{D}}\left[\nabla_\theta \mu_\theta(s)\, \nabla_a Q_\phi(s,a)\big|_{a=\mu_\theta(s)}\right]. \tag{8.3.2}$$

Storing transitions in a replay buffer $\mathcal{D}$ and sampling mini-batches breaks the temporal correlations between consecutive updates and allows each transition to be reused many times, giving DDPG its off-policy character, similarly to DQN.

Because μ_θ is deterministic, it would never explore if used directly. DDPG adds temporally-correlated noise ε (typically Ornstein–Uhlenbeck or simple Gaussian) to the actor's output at rollout time, $a = \mu_\theta(s) + \varepsilon$, while keeping the policy update noise-free. Target networks are refreshed via Polyak averaging, $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$ with $\tau \ll 1$ (typically $\tau \approx 0.005$), so they lag behind the learned networks and provide stable regression targets.

[3] $\int dx f(x) \partial_x \delta_D(x-y) = -\int dx \partial_x f(x) \delta_D(x-y) = -\partial_y f(y)$, but also $-\partial_y f(y) = -\partial_y \int dx f(x) \delta_D(x-y)$.

9 Rewards Engineering

Designing a good reward signal is often as important as the choice of algorithm. Sparse rewards are theoretically clean but slow to learn from; dense rewards accelerate learning but risk *reward hacking*. The techniques below address this from different angles: reward shaping adds auxiliary terms that guide exploration without changing the optimal policy, Lyapunov-based rewards construct dense signals from control-theoretic stability certificates, and constrained RL separates the primary objective from safety constraints rather than folding everything into a single scalar. We also show how different rewards definition can drastically change the variance of the return and thus impact the stability of the training.

9.1 Action penalty

Optimal policies usually require equlibrated actions to be successful, since too large actions can deviate from the optimal path. To enforce efficiency, effort penalties are commonly added to the reward to punish too large actions. A L2 penalty in the actions reads

$$\mathcal{R}_s^a = -\alpha \left\|a\right\|^2 \tag{9.1.1}$$

Assume a policy $\pi(a|s)$ with mean $\mu(s)$ and covariance matrix $\Sigma(s)$. From $\Sigma_{ij} = \mathbb{E}[(a_i - \mu_i(s))(a_j - \mu_j(s))] = \mathbb{E}[a_i a_j] - \mu_i(s)\mu_j(s)$ we have

$$\mathcal{R}_s^\pi = -\alpha \int da\, \pi(a|s) \left\|a\right\|^2 = -\alpha \left(\left\|\mu(s)\right\|^2 + \mathrm{Tr}[\Sigma(s)] \right) \tag{9.1.2}$$

where $\mathrm{Tr}[\Sigma(s)] = \sum_i \sigma_i^2$ with σ_i^2 the variance of each action i. For a L1 penalty one cannot write such a general closed analytical form, although it is possible for e.g. gaussian policies.

From Eq. 3.3.9, the value function can be written as

$$V^\pi(s) = \sum_{k=0}^{\infty} \gamma^k \sum_{s'} P_k^\pi(s'|s) \mathcal{R}_{s'}^\pi \tag{9.1.3}$$

$$= \mathcal{R}_s^\pi + \sum_{k=1}^{\infty} \gamma^k \sum_{s'} P_k^\pi(s'|s) \mathcal{R}_{s'}^\pi \tag{9.1.4}$$

$$= \mathcal{R}_s^\pi + \gamma \sum_{k=0}^{\infty} \gamma^k \sum_{s'} P_{k+1}^\pi(s'|s) \mathcal{R}_{s'}^\pi \tag{9.1.5}$$

$$= \mathcal{R}_s^\pi + \frac{\gamma}{1-\gamma} \sum_{s'} \mathcal{T}(s'|s) \mathcal{R}_{s'}^\pi \tag{9.1.6}$$

Thus, the value function correspondent to an action penalty term can be written as

$$V^\pi(s) = -\alpha \left(\left\|\mu(s)\right\|^2 + \mathrm{Tr}[\Sigma(s)] + \frac{\gamma}{1-\gamma} \int ds' \mathcal{T}(s'|s) \left(\left\|\mu(s')\right\|^2 + \mathrm{Tr}[\Sigma(s')] \right) \right) \tag{9.1.7}$$

If one can estimate the kernel $\mathcal{T}(s'|s)$ then the value function could be computed from the above formula.

9.2 Reward shaping

Reward shaping augments the environment reward with an auxiliary term to guide exploration and accelerate learning, without changing the optimal policy. Potential-based reward shaping was extensively studied by Ng et al. 1999 [15], and it is described as follows: given any *potential function* $\Phi : \mathcal{S} \to \mathbb{R}$, the shaped reward is

$$\tilde{R}(s, a, s') = R(s, a, s') + \gamma \Phi(s') - \Phi(s) \tag{9.2.1}$$

The shaping term $\gamma\Phi(s') - \Phi(s)$ is a difference of potentials, so when summing over a trajectory the terms telescope:

$$\tilde{G}_t = \sum_{k=0}^{T} \gamma^k \tilde{R}(s_{t+k}, a_{t+k}, s_{t+k+1}) \tag{9.2.2}$$

$$= \sum_{k=0}^{T} \gamma^k R(s_{t+k}, a_{t+k}, s_{t+k+1}) + \gamma\Phi(s_{t+1}) - \Phi(s_t) + \gamma^2\Phi(s_{t+2}) - \gamma\Phi(s_{t+1}) + \gamma^3\Phi(s_{t+3}) - \gamma^2\Phi(s_{t+2}) + ...$$

$$= \sum_{k=0}^{T} \gamma^k R(s_{t+k}, a_{t+k}, s_{t+k+1}) + \gamma^{T+1}\Phi(s_{t+T+1}) - \Phi(s_t)$$

$$\simeq \sum_{k=0}^{T} \gamma^k R(s_{t+k}, a_{t+k}, s_{t+k+1}) - \Phi(s_t) \tag{9.2.3}$$

where the terminal term $\gamma^{T+1}\Phi(s_{t+T+1}) \simeq 0$ for $\gamma < 1$ and $T \to \infty$ or terminal states with $\Phi = 0$. The approximation is exact for $T \to \infty$, as is the case of the discounted return. Consequently, the shaped value functions shift by Φ:

$$\tilde{Q}^\pi(s, a) = Q^\pi(s, a) - \Phi(s), \qquad \tilde{V}^\pi(s) = V^\pi(s) - \Phi(s) \tag{9.2.4}$$

while the advantage remains invariant:

$$\tilde{A}^\pi(s, a) = \tilde{Q}^\pi(s, a) - \tilde{V}^\pi(s) = Q^\pi(s, a) - V^\pi(s) = A^\pi(s, a) \tag{9.2.5}$$

and therefore the greedy policy is preserved:

$$a^*(s) = \arg\max_a \tilde{Q}(s, a) = \arg\max_a \ [Q(s, a) - \Phi(s)] = \arg\max_a Q(s, a) \tag{9.2.6}$$

Similarly, one can show that the optimal policy is preserved. To prove that, consider the objective $J^\pi = \sum_s \rho_0(s) V^\pi(s)$. The optimal policy reads $\pi^* = \arg\max_\pi J^\pi$. Then, for the modified objective $\tilde{J}^\pi = \sum_s \rho_0(s)\tilde{V}^\pi(s)$:

$$\tilde{\pi}^* = \arg\max_\pi \tilde{J}^\pi = \arg\max_\pi \sum_s \rho_0(s) \left(V^\pi(s) - \Phi(s)\right) = \arg\max_\pi \sum_s \rho_0(s) V^\pi(s) = \pi^* \tag{9.2.7}$$

since the term $\rho_0(s)\Phi(s)$ does not depend on the policy. For keeping the convenient property that the value function vanishes at terminal states s_*, the potential should also vanish: $\Phi(s_*) = 0$.

Note that reward shaping acts similarly to a gauge invariance in physics: a shift in the reward keeps the underlying optimal policy invariant. Thus, we have freedom to choose a reward function which may be beneficial for being more dense without affecting the optimality of actions. Note also that a Boltzmann policy $\pi(a|s) \propto \exp(A(s, a))$ would remain invariant too, even not being optimal.

There is a connection to GAE [5], see Sec. 4.4. GAE can be interpreted as reward shaping with $\Phi(s) = V(s)$ and an effective discount $\gamma\lambda$. In this case $\tilde{V}(s) = V(s) - \Phi(s) = 0$ at every state, so the shaped value function is identically zero and only the advantage signal remains, controlling the bias–variance trade-off via $\lambda \in [0, 1]$.

9.3 Lyapunov function

A Lyapunov function $\Phi(S)$ is defined such that $\Phi(S) \geq 0 \ \forall s$, $\Delta\Phi_t = \Phi(S_{t+1}) - \Phi(S_t) \leq 0$, where the equality holds in the success termination. Typically, Lyapunov functions can be built as quadratic forms $\Phi = (1/2)s^T P s$, with $P > 0$. For instance, the state s may refer to distance to a target, or velocity error wrt some reference. Therefore, we can build our rewards as

$$R_{t+1} = -\Delta\Phi_t = \Phi(S_t) - \Phi(S_{t+1}) \tag{9.3.1}$$

or $R(s, a, s') = \Phi(s) - \Phi(s')$. Lets compute the return and value for the undiscounted and discounted cases separately.

9.3.1 Undiscounted case

Thus, for the undiscounted return

$$G_t = \sum_{k=0}^{T-1} R_{t+k+1} = \Phi(S_t) - \Phi(S_{t+1}) + \Phi(S_{t+1}) - \Phi(S_{t+2}) + ... = \Phi(S_t) - \Phi(S_{t+T}) \tag{9.3.2}$$

where most of the terms cancel in a telescopic sum, similarly to previous section with reward shaping. Hence, the value function is

$$V(s) = \mathbb{E}_\pi[G_t|S_t = s] = \Phi(s) - \mathbb{E}_\pi[\Phi(S_{t+T})|S_t = s] \tag{9.3.3}$$

If success, $\Phi(S_{t+T}) = 0$. Thus, under a successful policy, the value function shall get closer to the Lyapunov function of that state $\Phi(s)$. In general, $\mathbb{E}_\pi[\Phi(S_{t+T})|S_t = s] = \int ds'\, P_T(s'|s)\Phi(s')$.

Note that, the further a state it is from success, the larger is the Lyapunov function, and thus the larger the value function too. This may seem counterintuitive, since we may be tempted to think that better states have larger value function, but that is a misunderstanding. The value function $V(s)$ indicates the expected return *from the state* s. If s is far from the objective, it means one can have large gains moving towards the target. Once s is close to the target, the expected returns are lower.

Note that if the reward is only given by the Lyapunov function, the value is independent on the path. To ensure optimality and avoid unnecessary actions, one may need to introduce an effort penalty as in Sec. 9.1:

$$R(s, a, s') = \Phi(s) - \Phi(s') - \lambda||a||^2 \tag{9.3.4}$$

However, discounts can also break path independence.

9.3.2 Discounted case

Alternatively, we can use discounted rewards, getting:

$$\begin{aligned} G_t &= -\sum_{k=0}^{\infty} \gamma^k \Delta\Phi_{t+k} && (9.3.5)\\ &= \Phi(S_t) - \Phi(S_{t+1}) + \gamma\left(\Phi(S_{t+1}) - \Phi(S_{t+2})\right) + \gamma^2\left(\Phi(S_{t+2}) - \Phi(S_{t+3})\right)... && (9.3.6)\\ &= \Phi(S_t) - (1-\gamma)\Phi(S_{t+1}) - \gamma(1-\gamma)\Phi(S_{t+2}) + ... && (9.3.7)\\ &= \Phi(S_t) - (1-\gamma)\sum_{k=0}^{\infty} \gamma^k \Phi(S_{t+k+1}) && (9.3.8) \end{aligned}$$

Note that in this case we have avoided path independence, and the return depends on all the states visited by the agent. Lets compute then the value function:

$$V(s) = \mathbb{E}_\pi[G_t|S_t = s] = \Phi(s) - (1-\gamma)\sum_{k=0}^{\infty} \gamma^k \mathbb{E}_\pi[\Phi(S_{t+k+1})|S_t = s] \tag{9.3.9}$$

If we approximate the expected value of the Lyapunov as $\mathbb{E}_\pi[\Phi(S_{t+k+1})|S_t = s] \sim \phi(s)$, then $V(s) \simeq \Phi(s) - \phi(s)(1-\gamma)\sum_{k=0}^{\infty}\gamma^k = \Phi(s) - \phi(s)$. This can be interpreted as the value function at s is equal to the Lyapunov function at s minus the weighted average of the Lyapunov function in the future states. Note also that using Eq. 3.5.1 the second term can be written as $\sum_{k=0}^{\infty}\gamma^k\mathbb{E}_\pi[\Phi(S_{t+k+1})|S_t = s] = \sum_{k=0}^{\infty}\gamma^k \int ds'\, P_{k+1}(s'|s)\Phi(s')$.

The above result in vector form it reads $\boldsymbol{V}^\pi = \left[I - (1-\gamma)(I - \gamma\mathcal{P}^\pi)^{-1}\mathcal{P}^\pi\right]\boldsymbol{\Phi} = (I - \gamma\mathcal{P}^\pi)^{-1}\left[I - \mathcal{P}^\pi\right]\boldsymbol{\Phi}$. Note that we can derive the above value function using $\mathcal{R}^\pi_s = \int ds' da\, \pi(a|s)\mathcal{P}^a_{ss'}\mathcal{R}^a_{ss'} = \Phi(s) - \int ds'\, \mathcal{P}^\pi_{ss'}\Phi(s')$, or in vector form $\boldsymbol{\mathcal{R}}^\pi = \boldsymbol{\Phi} - \mathcal{P}^\pi\boldsymbol{\Phi}$. Then, $\boldsymbol{V}^\pi = (I - \gamma\mathcal{P}^\pi)^{-1}(I - \mathcal{P}^\pi)\boldsymbol{\Phi}$

It is worth noting that we can define the kernel $\mathcal{T}(s'|s) = (1-\gamma)\sum_{k=0}^{\infty}\gamma^k P_{k+1}(s'|s)$ which fulfills $\int ds'\, \mathcal{T}(s'|s) = 1$. This kernel can be interpreted as the discounted probability of going from s to s'. Thus, we can write

$$V(s) = \Phi(s) - \int ds'\, \mathcal{T}(s'|s)\Phi(s') \tag{9.3.10}$$

For value estimation and actor-critic methods, one could in principle, instead of directly estimating the value function, write the value function as the Lyapunov function plus the learnable residual, $V_\theta(s) = \Phi(s) + f_\theta(s)$, which may simplify the learning likely reducing the bias. In an optimal policy, $f_\theta \to 0$. Equivalently, one could try to estimate the kernel $\mathcal{T}(s'|s)$ with a neural network. This is something I have not seen in the literature but is worth checking.

9.3.3 Optimal action

Lyapunov functions are usually defined in the continuous case, with $\Phi(s) \geq 0\ \forall s$ and $\dot\Phi \leq 0$. In the undiscounted case $G(t) = -\int_t^{t+T} dt' \dot\Phi(t') = \Phi(t) - \Phi(T)$.

Consider that the dynamics of the system are ruled by $\dot s = f(s) + g(s)\, a$. Then,

$$\dot\Phi(s) = \nabla_s \Phi(s) \dot s = \nabla_s \Phi(s)(f(s) + g(s)\, a) \tag{9.3.11}$$

It simple to see that if Φ is a Lyapunov function for the system with no control, i.e., when $a = 0$, then $\nabla_s \Phi(s) f(s) \leq 0$. Thus, for the control system, to ensure that $\dot\Phi \leq 0$, it is sufficient to add to the former condition that $\nabla_s \Phi(s) g(s)\, a \leq 0$. A possibility to fulfill that is to choose $a = -\alpha g^{-1} \nabla_s \Phi(s)$, with $\alpha > 0$, so that $\nabla_s \Phi(s) g\, a = -\alpha |\nabla_s \Phi(s)|^2 \leq 0$. Therefore, an optimal policy would go in the direction of the decreasing gradient of the Lyapunov function, $a \propto -\nabla_s \Phi(s)$. If the Lyapunov function is quadratic, then $a = -g^{-1} P s$, and the optimal action is linear in the features.

9.4 Variance reduction

Variance of the return can be of high importance for value approximation problems, for instance when we try to approximate the value function by a neural network using regression to the sampled returns, minimizing a loss of the form $(G(S_t) - V(S_t))^2$. High variance returns would lead to harder and less stable training. We demonstrate here that the specific shape of the rewards can notably impact this variance.

Increment-like rewards can help notably reducing the variance of the returns. To show that, lets consider some measure of the error of a a feature x_t at time t wrt a target x^*: ϵ_t. This can be an absolute error, a MSE or a Lyapunov function, any function which is larger the further from the target value. Then lets consider two reward cases: a case A where we define our rewards based directly on the error as $R^{(A)}_{t+1} = \alpha\, \epsilon_{t+1}$, and a case B where the reward has a increment-like shape, such as would have using a Lyapunov function, $R^{(A)}_{t+1} = \beta\, (\epsilon_t - \epsilon_{t+1})$. The discounted truncated returns are respectively

$$G^{(A)}_t = \alpha \sum_{k=0}^{n-1} \gamma^k \epsilon_{t+k+1} \qquad \text{and} \qquad G^{(B)}_t = \beta \sum_{k=0}^{n-1} \gamma^k (\epsilon_{t+k} - \epsilon_{t+k+1}) \tag{9.4.1}$$

Note that we can write the case B return in terms of the sum appearing in case A plus some boundary terms:

$$G^{(B)}_t = \beta \left(\epsilon_t - \gamma^{n-1} \epsilon_{t+n} - (1-\gamma) \sum_{k=0}^{n-1} \gamma^k \epsilon_{t+k+1} \right) \tag{9.4.2}$$

Following the scaling argument from Sec. 2.5, we can estimate the variance of the quantity $\sum_{k=0}^{n-1} \epsilon_{t+k+1}$ as $\sigma^2_\epsilon/(1-\gamma^2)$, with σ^2_ϵ the constant variance of ϵ_t, and neglecting correlations for simplicity (similar conclusions follow including them). Then, for case A we have the variance

$$\mathrm{Var}[G^{(A)}_t] \simeq \alpha^2 \frac{\sigma^2_\epsilon}{1-\gamma^2} \tag{9.4.3}$$

For case B, neglecting the term γ^n and cross correlations, we get

$$\mathrm{Var}[G_t^{(B)}] \simeq \beta^2 \left(\sigma_\epsilon^2 + (1-\gamma)^2 \frac{\sigma_\epsilon^2}{1-\gamma^2} \right) = \beta^2 \frac{\sigma_\epsilon^2}{1+\gamma} \tag{9.4.4}$$

With typical values of the discount factor around $\gamma = 0.99$, $1/(1-\gamma^2) \simeq 50$ and $1/(1+\gamma) \simeq 0.5$, so the case A variance is 2 orders of magnitude larger than case B for similar order of α and β. Therefore, using increment-based rewards like case B, with some error or Lyapunov function can notably reduce variance of the returns and thus lead to a more stable training. Note that although one could reduce the absolute scale of the variance in case A by reducing the parameter α, that would not change the signal-to-noise ratio, not being an useful variance reduction method in practice.

Finally, note that the increment reward forms can also reduce the signal-to-noise ratio. To show that, define the mean reward $\bar{\epsilon} = \mathbb{E}[(1-\gamma)\sum_{k=0}^{n-1} \gamma^k \epsilon_{t+k+1}]$ and the mean discounted increment $\Delta\epsilon = \mathbb{E}[\epsilon_t - \gamma^{n-1}\epsilon_{t+n}]$. Then, $\mathbb{E}[G_t^{(A)}] = \alpha\,\bar{\epsilon}/(1-\gamma)$ and $\mathbb{E}[G_t^{(B)}] = \beta\,(\Delta\epsilon - \bar{\epsilon})$. Then, the signal-to-noise ratio $SNR = \mathbb{E}[G_t]^2/\,\mathrm{Var}[G_t]$ reads

$$SNR^{(A)} \simeq \left(\frac{1+\gamma}{1-\gamma} \right) \frac{\bar{\epsilon}^2}{\sigma_\epsilon^2} \qquad \text{and} \qquad SNR^{(B)} \simeq (1+\gamma) \frac{(\Delta\epsilon - \bar{\epsilon})^2}{\sigma_\epsilon^2} \tag{9.4.5}$$

The gamma factors are ~ 200 and ~ 2 respectively, so the prefactor in the SNR is suppressed by two orders of magnitude. We can rewrite case B in terms of case A as $SNR^{(B)} = (1-\gamma)SNR^{(A)} + (1+\gamma)(\Delta\epsilon^2 - 2\bar{\epsilon}\Delta\epsilon)/\sigma_\epsilon^2$ Thus, one has to have this in mind when building the reward function: there is a tradeoff between variance and SNR, and variance reduction can come at a cost of SNR reduction, which depending on the scenario could be harmful.

10 Continuous time reinforcement learning

So far we have treated time evolution with discrete steps. We devote this section to study reinforcement learning in a continuous time approach, extending the previous theory to continuous evolution.

10.1 Continuous time stochastic processes

Most of theory from discrete stochastic processes is directly extended in the continuous domain. Let $X(t)$ be a random process on time t. The joint probability of be at $X(t)$ and $X(t+\tau)$ can be decomposed with the conditional probability as

$$P(X(t), X(t+\tau)) = P(X(t+\tau)|X(t))P(X(t)) \tag{10.1.1}$$

Consider now the three joint probability with $\tau \geq \tau' \geq 0$

$$P(X(t), X(t+\tau'), X(t+\tau)) = P(X(t+\tau)|X(t+\tau'))P(X(t+\tau')|X(t))P(X(t)) \tag{10.1.2}$$

integrating $X(t+\tau')$ we must get the marginal distribution $P(X(t), X(t+\tau))$, and dividing by $P(X(t))$, getting the Chapman-Kolmogorov equation

$$P(X(t+\tau)|X(t)) = \int dX(t+\tau')\; P(X(t+\tau)|X(t+\tau'))P(X(t+\tau')|X(t)) \tag{10.1.3}$$

We can thus define $P(X(t+\tau) = x'|X(t) = x) = P_\tau(x''|x)$, which is the continuous time equivalent of the k-step probability transition for discrete evolution The Chapman-Kolmogorov equation can be put in a more compact form

$$P_{t+\tau}(x''|x) = \int dx' P_\tau(x''|x') P_t(x'|x) \tag{10.1.4}$$

analogous to Eq. 1.3.4 for discrete evolution. The above formulae can be applied to MDPs using X as the state-action pairs

$$P_\tau(s', a'|s, a) = P(S(t+\tau) = s', A(t+\tau) = a'|S(t) = s, A(t) = a) \tag{10.1.5}$$

and the Chapman-Kolmogorov equation takes the form

$$P_{t+\tau}(s', a'|s, a) = \int ds'' da'' \, P_\tau(s', a'|s'', a'') P_t(s'', a''|s, a) \tag{10.1.6}$$

Finally, the action of transition probability can be written as an operator P_t^a acting on a function $f(s)$, which also can be expressed in terms of the generator $\mathcal{L}^a$, so that

$$(P_t^a f)(s) = \int ds' P_t(s'|s, a) f(s') = e^{t\mathcal{L}^a} f(s) \tag{10.1.7}$$

Note that with this operator notation, the Chapman-Kolmogorov equation can be expressed as a product of operators: $P_{t+\tau}^a = e^{(t+\tau)\mathcal{L}^a} = e^{t\mathcal{L}^a} e^{\tau\mathcal{L}^a} = P_t^a P_\tau^a$.

10.2 Continuous time return

To extend the return defined in the discrete case, Eq. 2.1.4, we have to substitute $\gamma^t = \exp(\log \gamma^t)) = \exp(-\alpha t)$, with $\alpha = \log \gamma$ and replace the sum by an integral. Thus, the continuous time return reads

$$G(t) = \int_0^\infty d\tau \, e^{-\alpha\tau} R(t + \tau) = \int_t^\infty d\tau \, e^{-\alpha(\tau - t)} R(\tau) \tag{10.2.1}$$

Similarly to Eq. 2.1.5, we can show it fulfills the recurrence relation

$$G(t) = \int_0^{\Delta t} d\tau \, e^{-\alpha\tau} R(t + \tau) + e^{-\alpha\Delta t} G(t + \Delta t) \tag{10.2.2}$$

rearranging

$$\frac{G(t + \Delta t) - G(t)}{\Delta t} = \frac{\left(e^{\alpha\Delta t} - 1\right)}{\Delta t} G(t) - \frac{e^{\alpha\Delta t}}{\Delta t} \int_0^{\Delta t} d\tau \, e^{-\alpha\tau} R(t + \tau) \tag{10.2.3}$$

We can expand the right hand side by Taylor in powers of Δt and taking the limit $\Delta \to 0$, neglect order Δt^2 and larger, getting the differential equation

$$\frac{dG(t)}{dt} = \alpha G(t) - R(t) \tag{10.2.4}$$

which has as solution Eq. 10.2.1 when integrated taking the boundaries ∞ and t. The above equation can also be found deriving Eq. 10.2.1 from its second form.

10.3 Value functions

We define the state-action value function in the same fashion than in the discrete case, as

$$Q(s, a) = \mathbb{E}[G(t)|S(t) = s, A(t) = a] \tag{10.3.1}$$

while the reward is also given by $\mathbb{E}[R(t)|S(t) = s, A(t) = a] = \mathcal{R}_s^a$. Using the law of total expectation we get

$$\mathbb{E}[R(t + \tau)|S(t) = s, A(t) = a] = \mathop{\mathbb{E}}_{s', a' \sim P(t+\tau)} [\mathbb{E}[R(t + \tau)|S(t + \tau) = s', A(t + \tau) = a']|S(t) = s, A(t) = a] \tag{10.3.2}$$

$$= \mathop{\mathbb{E}}_{s', a' \sim P(t+\tau)} \left[\mathcal{R}_{s'}^{a'}|S(t) = s, A(t) = a\right] \tag{10.3.3}$$

$$= \int ds' da' P_\tau(s', a'|s, a) \mathcal{R}_{s'}^{a'} \tag{10.3.4}$$

and hence a concise expression for Q:

$$Q(s,a) = \int_0^\infty d\tau\, e^{-\alpha\tau} \int ds' da'\, P_\tau(s', a'|s, a)\, \mathcal{R}_{s'}^{a'} \tag{10.3.5}$$

equivalent to Eq. 3.3.8 in the discrete time case. It is worth introducing the notation

$$\langle f(s,a)\rangle_t = \int ds' da' P_t(s', a'|s, a) f(s', a') = \int ds' P_t(s'|s, a) \int da' \pi(a'|s') f(s', a') \tag{10.3.6}$$

We can interpret $\langle f(s,a)\rangle_t$ as the expectation of f after evolving a time t starting from s, a. Then, we can write

$$Q(s,a) = \int_0^\infty d\tau\, e^{-\alpha\tau} \langle \mathcal{R}_s^a \rangle_\tau \tag{10.3.7}$$

Using Eq. 10.2.2, one can get a Bellman expectation equation

$$Q(s,a) = \int_0^{\Delta t} d\tau e^{-\alpha\tau} \int ds' da' P_\tau(s', a'|s, a) \mathcal{R}_{s'}^{a'} + e^{-\alpha\Delta t} \int ds' da' P_{\Delta t}(s', a'|s, a) Q(s', a') \tag{10.3.8}$$

or equivalently

$$Q(s,a) = \int_0^t d\tau e^{-\alpha\tau} \langle \mathcal{R}_s^a \rangle_\tau + e^{-\alpha t} \langle Q(s,a)\rangle_t \tag{10.3.9}$$

which is similar to a Bellman expectation equation. Deriving wrt t we get a differential equation for $\langle Q(s,a)\rangle_t$:

$$\frac{d\langle Q(s,a)\rangle_t}{dt} = \alpha \langle Q(s,a)\rangle_t - \langle \mathcal{R}_s^a \rangle_t \tag{10.3.10}$$

which is analogous to Eq. 10.2.4 with averages. The state value function $V(s) = \int da\, \pi(a|s) Q(s,a)$ satifies similar relations:

$$V(s) = \int_0^\infty d\tau e^{-\alpha\tau} \langle \mathcal{R}_s^\pi \rangle_\tau = \int_0^t d\tau e^{-\alpha\tau} \langle \mathcal{R}_s^\pi \rangle_\tau + e^{-\alpha t} \langle V(s)\rangle_t \tag{10.3.11}$$

and

$$\frac{d\langle V(s)\rangle_t}{dt} = \alpha \langle V(s)\rangle_t - \langle \mathcal{R}_s^\pi \rangle_t \tag{10.3.12}$$

Also, if we define a value between times t_0 and t_1, $V(s, t_0, t_1) = \int_{t_0}^{t_1} d\tau e^{-\alpha\tau} \langle \mathcal{R}_s^\pi \rangle_\tau$, note that we have $V(s, t_0, t_2) = V(s, t_0, t_1) + V(s, t_1, t_2)$, which is sometimes called the Bellman's principle of optimality. This states that the value function can be decomposed in the value of different time segments and sum up.

10.4 Hamilton-Jacobi-Bellman equations

We can derive further related differential equations applying an infinitesimal time increment in the above equations. First, note that $\langle Q(s,a)\rangle_t$ can be written in terms of the generator for $t > 0$ as

$$\langle Q(s,a)\rangle_t = \int ds' P_t(s'|s, a) V(s') = e^{t\mathcal{L}^a} V(s) \tag{10.4.1}$$

Note that for $t = 0$ we cannot apply the above formula, since we have $\langle Q(s,a)\rangle_0 = Q(s,a)$. Then, expanding in powers of the time increment,

$$\langle Q(s,a)\rangle_{\Delta t} \simeq Q(s,a) + \Delta t \mathcal{L}^a V(s) + O(\Delta t^2) \tag{10.4.2}$$

Then, we get

$$Q(s,a) = \int_0^{\Delta t} d\tau e^{-\alpha\tau} \langle \mathcal{R}_s^a \rangle_\tau + e^{-\alpha\Delta t} \langle Q(s,a) \rangle_{\Delta t} \tag{10.4.3}$$

$$\simeq \mathcal{R}_s^a \Delta t + (1 - \alpha\Delta t) \left(Q(s,a) + \Delta t \mathcal{L}^a V(s) \right) + O(\Delta t^2) \tag{10.4.4}$$

Canceling terms, and following the same procedure for V, we derive a differential equation known as the *Hamilton-Jacobi-Bellman expectation equations*

$$\alpha Q(s,a) = \mathcal{R}_s^a + \mathcal{L}^a V(s) \tag{10.4.5}$$

$$\alpha V(s) = \mathcal{R}_s^\pi + \mathcal{L}^\pi V(s) \tag{10.4.6}$$

These equations apply on policy, when the value functions are computed based on the policy π. However, this formalism is usually applied to look for optimal policies, and thus used the optimality version, analogous to Sec. 3.7. What is commonly referred as HJB equation is the *Hamilton-Jacobi-Bellman optimality equation*

$$\boxed{\alpha V(s) = \max_a \left[\mathcal{R}_s^a + \mathcal{L}^a V(s) \right]} \tag{10.4.7}$$

What is the exact form of $\mathcal{L}^a V(s)$? To figure it out, note that $\exp(\Delta t \mathcal{L}^a V(s)) = \int ds' P_{\Delta t}(s'|s,a) V(s') = \mathbb{E}[V(S(t+\Delta t) \mid S(t) = s, A(t) = a]$. On the other hand, for infinitesimal time increments dt, we can expand V with its derivatives. Here we allow for explicit dependence on t:

$$V(S(t+dt), t+dt) \simeq V(S(t),t) + \partial_t V(S(t),t) dt + \nabla_s V(S(t)) \dot{s}\, dt + O(dt^2) \tag{10.4.8}$$

with the notation $\partial_t f = \frac{\partial f}{\partial t}$. If the evolution of the states fulfills a deterministic differential equation of the form $\dot{s} = f(s,a,t)$, and recalling that $\exp(dt\mathcal{L}^a) V(s) \simeq V(s) + dt \mathcal{L}^a V(s)$, then the generator acting on V can be written as[4]

$$\mathcal{L}^a V(s) = \partial_t V(s) + f(s,a,t) \nabla_s V(s) \tag{10.4.9}$$

Thus, the HJB equation for the above scenario reads

$$\boxed{\alpha V(s) = \partial_t V(s) + \max_a \left[\mathcal{R}_s^a + f(s,a,t) \nabla_s V(s) \right]} \tag{10.4.10}$$

10.5 Occupancy measure

We define the occupancy measure distribution in the continuous time case as

$$\rho(s,a) = \alpha \int_0^\infty d\tau e^{-\alpha\tau} P(S(\tau) = s, A(\tau) = a) \tag{10.5.1}$$

which can be casted in terms of the transition probability as

$$\rho(s,a) = \alpha \int_0^\infty d\tau e^{-\alpha\tau} \int ds' da' P_\tau(s,a|s',a') \rho_0(s',a') \tag{10.5.2}$$

From the Chapman-Kolmogorov equation, Eq. 10.1.6, we can derive a recurrence relation for the continuous time occupancy measure:

$$\rho(s,a) = e^{-\alpha t} \int ds' da'\, P_t(s,a|s',a') \rho(s',a') \tag{10.5.3}$$

Similarly, the marginalized distribution $\rho(s) = \int da\, \rho(s,a) = \rho(s,a)/\pi(s|a)$ also fulfills a recurrence relation (we are ommiting the superscript π)

[4] If the evolution equation is instead stochastic with a noise component η like $\dot{s} = f(s,a,t) + \eta$ there is an additional term given by the noise variance, coming from the Itô's lemma, omitted here for simplicity.

$$\rho(s) = e^{-\alpha t} \int ds' da' \, P_t(s|s')\rho(s') \tag{10.5.4}$$

Finally, the objective function can be written in terms of the occupancy following the same procedure as in Sec. 6

$$J = \mathbb{E}[G(0)] = \int_0^\infty d\tau \, e^{-\alpha\tau} \mathbb{E}[R(\tau)] \tag{10.5.5}$$

$$= \int_0^\infty d\tau \, e^{-\alpha\tau} \mathop{\mathbb{E}}_{s,a\sim P(\tau)} [\mathbb{E}[R(\tau)|S(\tau) = s, A(\tau) = a]] \tag{10.5.6}$$

$$= \int_0^\infty d\tau \, e^{-\alpha\tau} \int ds \, da \, P(S(\tau) = s, A(\tau) = a)\mathcal{R}_s^a \tag{10.5.7}$$

$$= \frac{1}{\alpha} \int ds \, da \, \rho(s, a)\mathcal{R}_s^a = \frac{1}{\alpha} \int ds \, \rho(s)\mathcal{R}_s^\pi \tag{10.5.8}$$

References

[1] David Silver. Lectures on reinforcement learning. URL: https://www.davidsilver.uk/teaching/, 2015.

[2] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.

[3] Joshua Achiam. Spinning Up in Deep Reinforcement Learning, 2018. URL: https://spinningup.openai.com/en/latest/.

[4] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017. doi:10.1038/nature24270.

[5] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016. URL: https://arxiv.org/abs/1506.02438.

[6] Antonio Serrano-Muñoz, Dimitrios Chrysostomou, Simon Bøgh, and Nestor Arana-Arexolaleiba. skrl: Modular and flexible library for reinforcement learning. *Journal of Machine Learning Research*, 24(254):1–9, 2023. URL: http://jmlr.org/papers/v24/23-0112.html.

[7] Clemens Schwarke, Mayank Mittal, Nikita Rudin, David Hoeller, and Marco Hutter. Rsl-rl: A learning library for robotics research. *arXiv preprint arXiv:2509.10771*, 2025.

[8] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 22–31, 2017. URL: https://arxiv.org/abs/1705.10528.

[9] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, pages 267–274, 2002. URL: https://icml.cc/2002/papers/0267.pdf.

[10] Sham Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 14, 2001. URL: https://papers.nips.cc/paper/2001/hash/4b86abe48d358ecf4528536e1cc3f908-Abstract.html.

[11] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015. URL: https://arxiv.org/abs/1502.05477.

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL: https://arxiv.org/abs/1707.06347.

[13] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 2014. URL: http://proceedings.mlr.press/v32/silver14.html.

[14] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016. URL: https://arxiv.org/abs/1509.02971.

[15] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, page 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

A Appendix

A.1 Law of Total Expectation

Theorem (Law of Total Expectation / Tower Property). Let X and Y be random variables defined on the same probability space. Then:

$$\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X \mid Y]] \tag{A.1.1}$$

Proof (discrete case). By definition of conditional expectation, $\mathbb{E}[X \mid Y = y] = \sum_x x\, P(X = x \mid Y = y)$. Therefore:

$$\mathbb{E}[\mathbb{E}[X \mid Y]] = \sum_y P(Y = y)\, \mathbb{E}[X \mid Y = y] \tag{A.1.2}$$

$$= \sum_y P(Y = y) \sum_x x\, P(X = x \mid Y = y) \tag{A.1.3}$$

$$= \sum_x x \sum_y P(Y = y)\, P(X = x \mid Y = y) \tag{A.1.4}$$

$$= \sum_x x \sum_y P(X = x, Y = y) \tag{A.1.5}$$

$$= \sum_x x\, P(X = x) \;=\; \mathbb{E}[X] \tag{A.1.6}$$

where the third line swaps the order of summation, the fourth uses the definition of conditional probability $P(X = x \mid Y = y) = P(X = x, Y = y)/P(Y = y)$, and the fifth marginalises over Y. The continuous case follows analogously by replacing sums with integrals. □

A.2 Expected Grad-Log-Prob Lemma

Lemma (EGLP). Let $P_\theta(x)$ be a probability distribution parameterized by θ, satisfying regularity conditions that permit interchange of differentiation and integration. Then:

$$\mathbb{E}_{x \sim P_\theta}[\nabla_\theta \log P_\theta(x)] = 0 \tag{A.2.1}$$

Proof. We start from the definition of the expectation and apply the log-derivative identity $\nabla_\theta \log P_\theta(x) =$

$\nabla_\theta P_\theta(x)/P_\theta(x)$:

$$\mathbb{E}_{x\sim P_\theta}[\nabla_\theta \log P_\theta(x)] = \int P_\theta(x)\, \nabla_\theta \log P_\theta(x)\, dx \tag{A.2.2}$$

$$= \int P_\theta(x)\, \frac{\nabla_\theta P_\theta(x)}{P_\theta(x)}\, dx \tag{A.2.3}$$

$$= \int \nabla_\theta P_\theta(x)\, dx \tag{A.2.4}$$

$$= \nabla_\theta \int P_\theta(x)\, dx \tag{A.2.5}$$

$$= \nabla_\theta\, 1 \;=\; 0 \tag{A.2.6}$$

where the interchange of ∇_θ and $\int$ in the fourth line is justified by the regularity assumptions (e.g. dominated convergence). The last step uses the fact that P_θ is a normalized probability distribution. □

The EGLP lemma is often used with the policy in the form $\mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\,|s] = 0$.

A.3 KL Divergence and Fisher Information Matrix

The Fisher Information Matrix (FIM) at parameter θ is defined as:

$$F(\theta) = \mathbb{E}_{x\sim p_\theta}\left[\nabla_\theta \log p_\theta(x)\, \nabla_\theta \log p_\theta(x)^\top\right] \tag{A.3.1}$$

Proposition. For small parameter variations $\delta\theta$, the KL divergence from p_θ to $p_{\theta+\delta\theta}$ satisfies:

$$D_{\mathrm{KL}}(p_\theta \parallel p_{\theta+\delta\theta}) \approx \tfrac{1}{2}\, \delta\theta^\top F(\theta)\, \delta\theta \tag{A.3.2}$$

to second order in $\|\delta\theta\|$.

Proof. We Taylor-expand $\log p_{\theta+\delta\theta}(x)$ to second order around θ:

$$\log p_{\theta+\delta\theta}(x) = \log p_\theta(x) + \delta\theta^\top \nabla_\theta \log p_\theta(x) + \tfrac{1}{2}\, \delta\theta^\top \nabla_\theta^2 \log p_\theta(x)\, \delta\theta + \mathcal{O}(\|\delta\theta\|^3) \tag{A.3.3}$$

Substituting into the definition $D_{\mathrm{KL}}(p_\theta \| p_{\theta+\delta\theta}) = \mathbb{E}_{x\sim p_\theta}[\log p_\theta(x) - \log p_{\theta+\delta\theta}(x)]$:

$$\begin{aligned} D_{\mathrm{KL}}(p_\theta \parallel p_{\theta+\delta\theta}) = {} & -\delta\theta^\top \mathbb{E}_{x\sim p_\theta}[\nabla_\theta \log p_\theta(x)] \\ & - \tfrac{1}{2}\, \delta\theta^\top \mathbb{E}_{x\sim p_\theta}\left[\nabla_\theta^2 \log p_\theta(x)\right] \delta\theta + \mathcal{O}(\|\delta\theta\|^3) \end{aligned} \tag{A.3.4}$$

The first term vanishes by the EGLP lemma (Appendix A.2). For the second term we establish the identity $\mathbb{E}_{x\sim p_\theta}[\nabla_\theta^2 \log p_\theta(x)] = -F(\theta)$.

Applying the log-derivative identity $\nabla_\theta \log p_\theta(x) = \nabla_\theta p_\theta(x)/p_\theta(x)$ to differentiate once more:

$$\nabla_\theta^2 \log p_\theta(x) = \frac{\nabla_\theta^2 p_\theta(x)}{p_\theta(x)} - \nabla_\theta \log p_\theta(x)\, \nabla_\theta \log p_\theta(x)^\top \tag{A.3.5}$$

Taking expectations and interchanging differentiation with integration (justified by the same regularity conditions as in the EGLP lemma):

$$\mathbb{E}_{x\sim p_\theta}\left[\nabla_\theta^2 \log p_\theta(x)\right] = \int \nabla_\theta^2 p_\theta(x)\, dx - F(\theta) \tag{A.3.6}$$

$$= \nabla_\theta^2 \int p_\theta(x)\, dx - F(\theta) \tag{A.3.7}$$

$$= -F(\theta) \tag{A.3.8}$$

where the last step uses normalization of p_θ. Substituting back yields:

$$D_{\mathrm{KL}}(p_\theta \parallel p_{\theta+\delta\theta}) = \tfrac{1}{2}\, \delta\theta^\top F(\theta)\, \delta\theta + \mathcal{O}(\|\delta\theta\|^3) \tag{A.3.9}$$

□

An immediate corollary is that the FIM equals the Hessian of the KL divergence with respect to $\delta\theta$, evaluated at $\delta\theta = 0$:

$$\nabla^2_{\delta\theta} D_{\mathrm{KL}}(p_\theta \parallel p_{\theta+\delta\theta}) \Big|_{\delta\theta=0} = F(\theta) \tag{A.3.10}$$

This follows directly from the quadratic approximation: the Hessian of $\frac{1}{2}\delta\theta^\top F(\theta)\,\delta\theta$ with respect to $\delta\theta$ is $F(\theta)$.

This result shows that the FIM acts as the local Riemannian metric on the statistical manifold of distributions: it quantifies how distinguishable two nearby distributions are, independently of the parameterization chosen. This geometric viewpoint underlies natural gradient descent and trust-region methods such as TRPO, where parameter updates are constrained in KL distance rather than Euclidean distance.

A.4 Functional Derivatives

A *functional* $F[f]$ maps a function π to a scalar. The *functional derivative* $\frac{\delta F}{\delta f(x)}$ is the continuous analogue of a partial derivative: it measures the infinitesimal change in F when f is perturbed at a single point x. Formally, it is defined by the Gâteaux derivative

$$\int \frac{\delta F[f]}{\delta f(x)}\,\eta(x)\,dx = \lim_{\varepsilon\to 0} \frac{F[f+\varepsilon\,\eta] - F[f]}{\varepsilon} \tag{A.4.1}$$

for any test perturbation η. In RL we apply functional derivatives to the policy, so $f \to \pi$ and $x \to s, a$.

The most basic functional derivative is that of the function itself:

$$\frac{\delta f(y)}{\delta f(x)} = \delta(x-y) \tag{A.4.2}$$

which is the continuous analogue of $\partial x_i/\partial x_j = \delta_{ij}$. In the discrete state–action setting used throughout these notes, the Dirac deltas become Kronecker deltas. Applying the above identity to a policy it reads:

$$\frac{\delta \pi(a'|s')}{\delta \pi(a|s)} = \delta(s-s')\,\delta(a-a') \tag{A.4.3}$$

An example of usage with a functional $F[f] = \int dx f(x)g(x)$:

$$\frac{\delta F[f]}{\delta f(y)} = g(y) \tag{A.4.4}$$

If $F[\pi] = \sum_{a'} \pi(a'|s_0)\,g(a', s_0)$ for some fixed function g, then applying the fundamental identity gives

$$\frac{\delta F[\pi]}{\delta \pi(a|s)} = g(a,s)\,\delta(s-s_0). \tag{A.4.5}$$

This is the rule used to differentiate the policy-weighted transition operator $\mathcal{P}^\pi_{s's''} = \sum_{a'} \pi(a'|s')\mathcal{P}^{a'}_{s's''}$ and reward vector $\mathcal{R}^\pi_{s'} = \sum_{a'} \pi(a'|s')\mathcal{R}^a_{s'}$. Finally, if the functional F depends on the function f_θ with some parameter θ, the gradient of the former wrt θ can be written in terms of the functional derivative of f:

$$\nabla_\theta F[f_\theta] = \int dx \frac{\delta F}{\delta f_\theta(x)} \nabla_\theta f_\theta(x) \tag{A.4.6}$$