Markov Decision Processes

Guillaume Barnier

Academic Year 2020-2021

Contents

1 Markov Process **1**
1.1 Definitions . . . 1

2 Markov Reward Process **2**
2.1 Definitions . . . 2
2.2 Bellman Equations . . . 4

3 Markov Decision Process **6**
3.1 Definitions . . . 6
3.2 Bellman Equations . . . 7

4 MDP Planning by Dynamic Programming **9**
4.1 Policy evaluation (prediction) . . . 9
4.2 Optimal value function . . . 9
4.3 Optimal policy . . . 10
4.4 Bellman optimality equations . . . 10
4.5 Control . . . 11
4.5.1 Policy iteration . . . 11
4.5.2 Policy improvement . . . 11
4.5.3 Value iteration . . . 12
4.5.4 Contraction mapping . . . 13

5 Exercises **13**

1 Markov Process

1.1 Definitions

Definition 1.1 (Markov Property)**.** Let $S_t = (s_0, s_1, ...)$ be a stochastic process evolving according to a transition dynamic P. This stochastic process satisfies the Markov property if

$$p(s_t|s_0, s_1, ..., s_{t-1}) = p(s_t|s_{t-1}),\ \forall t \in \mathbb{N} \tag{1}$$

Definition 1.2 (Markov Process)**.** A Markov Process (MP) is a stochastic process that satisfies the Markov property.

In a RL setting, we often make two additional assumptions:

- **Finite state space**. The state space of the Markov process is finite. This means that for the Markov process $(s_0, s_1, ...)$, there is a state space S with $|S| = n < \infty$, such that for all realizations of the Markov process, we have $s_t \in S$ for all t.
- **Stationary transition probability**. The transition probabilities are time independent:

$$p(s_p = s'|s_{p-1} = s) = p(s_q = s'|s_{q-1} = s),\ \forall (p, q) \tag{2}$$

A Markov process satisfying these assumptions is also sometimes called a Markov chain, although the precise definition of a Markov chain varies. With these assumptions, we can define characterize a Markov process with the following definition.

Definition 1.3 (Markov Process/Markov Chain)**.** A Markov Process is a tuple $\big(S, P\big)$, where

- S is the finite state-space of the Markov process, $|S| = n < \infty$
- P is the state transition probability model where $P_{ss'} = p(s_{t+1} = s' | s_t = s)$

Lemma 1.1. $p(s_{t+n}|s_t = s) = p(s_n|s_0 = s)$ for all t and n

Proof. I show this property by induction on n:

- For $n = 1$, $p(s_{t+1}|s_t = s) = p(s_1|s_0 = s)$ is true by the stationarity assumption
- I assume that $p(s_{t+n}|s_t = s) = p(s_n|s_0 = s)$ is true for n
- I show that $p(s_{t+n+1}|s_t = s) = p(s_{n+1}|s_0 = s)$:

$$
\begin{aligned}
p(s_{t+n+1}|s_t = s) &= \sum_{s'} p(s_{t+n+1}, s_{t+n} = s'|s_t = s) && (3)\\
&= \sum_{s'} p(s_{t+n+1}|s_t = s, s_{t+n} = s')p(s_{t+n} = s'|s_t = s) && (4)\\
&= \sum_{s'} p(s_{t+n+1}|s_{t+n} = s')p(s_n = s'|s_0 = s) && (5)\\
&= \sum_{s'} p(s_{n+1}|s_n = s')p(s_n = s'|s_0 = s) && (6)\\
&= \sum_{s'} p(s_{n+1}, s_n = s'|s_0 = s) && (7)\\
&= p(s_{n+1}|s_0 = s) && (8)
\end{aligned}
$$

Remark:

- Equation 3 is obtained by using the fact that for X, Y, Z random variables,

$$p(X|Y) = \sum_z z\, p(X, Z = z|Y) \tag{9}$$

- Equation 5 is obtained by using the Markov property and the induction assumption
- Equation 7 is obtained by using the Markov property again followed by Bayes rule

$$
\begin{aligned}
p(s_{n+1}|s_n = s')p(s_n = s'|s_0 = s) &= p(s_{n+1}|s_n = s', s_0 = s)p(s_n = s'|s_0 = s) && (10)\\
&= p(s_{n+1}, s_n = s'|s_0 = s) && (11)
\end{aligned}
$$

2 Markov Reward Process

2.1 Definitions

Definition 2.1 (Markov Reward Process)**.** A Markov Reward Process (MRP) is a tuple $\big(S, P, R, \gamma\big)$, where

- S is the finite state-space of the Markov process (assume $|S| = n < \infty$)
- P is the state transition probability model where $P_{ss'} = p(s_{t+1} = s'|s_t = s)$
- $R : S \mapsto \mathbb{R}$ is a reward function that maps states to rewards, $R(s) = E[r_t|s_t = s]$
- $\gamma \in [0, 1]$ is a discount factor

In a Markov reward process, whenever a transition happens from a current state s to a successor state s', a reward is obtained depending on the current state s. Thus for the Markov process $(s_0, s_1, ...)$, each transition $s_t \to s_{t+1}$ is accompanied by a reward r_t for all $i = 0, 1, ...$, and so a particular episode of the Markov reward process is represented as $(s_0, r_0, s_1, r_1, s_2, r_2, ...)$. We should note that these rewards can be either deterministic or stochastic.

Definition 2.2 (Expected reward)**.** For a state $s \in S$, we define the expected reward $R(s)$ by

$$R(s) = E[r_0|s = s_0] \tag{12}$$

Just like the assumption of stationary transition probabilities, going forward we will also assume *stationarity of the rewards*. In the deterministic case, this implies that $r_i = r_j$ wherever $s_i = s_j$. In the stochastic case, we require that the cumulative distribution functions (CDF) of the rewards conditioned on the current state be time independent:

$$F(r_i|s_i = s) = F(r_j|s_j = s), \tag{13}$$

where F denotes the cumulative distribution function of r_i conditioned on s_i. Therefore, the reward function $R(s)$ is independent of t and we have the following properties:

$$p(r_{t+p}|s_{t+p} = s) = p(r_t|s_t = s) \tag{14}$$

$$R(s) = E(r_t|s_t = s) \tag{15}$$

Definition 2.3 (**Horizon**)**.** The horizon H of a Markov reward process is defined as the number of time steps in each episode (realization) of the process. The horizon can be finite or infinite. If the horizon is finite, then the process is also called a finite Markov reward process.

Definition 2.4 (**Return**)**.** The return G_t of a Markov reward process is defined as the discounted sum of rewards starting at time t up to the horizon H, and is given by

$$G_t = \sum_{k=t}^{H-1} \gamma^{k-t} r_k, \ for\ t \in [0, H-1] \tag{16}$$

For example, $G_0 = r_0 + \gamma r_1 + \gamma^2 r_2 + ... + \gamma^{H-1} r_{H-1}$

Definition 2.5 (**State value function**)**.** The state value function $V_t(s)$ for a Markov reward process and a state $s \in S$ is defined as the expected return starting from state s at time t, and is given by the following expression:

$$V_t(s) = E[G_t|s_t = s], \tag{17}$$

and can be interpreted as the long-term value of state s.

Lemma 2.1. Let us assume that

- Transition probability is stationary
- Rewards are stationary
- H is infinite.

Then $V_t(s)$ is independent of t. That is,

$$V_t(s) = V(s) \tag{18}$$

Proof. Even though this property seems obvious and intuitive, the proof is not totally straightforward (at least to me). I conduct the demonstration in two steps:

(1) I prove that $E[r_{t+n}|s_t = s] = E[r_n|s_0 = s]$ by recursion on n:

- For $n = 0$, I use the result the rewards' stationarity assumption to show that

$$E(r_t|s_t = s) = \sum_r r\ p(r_t = r|s_t = s) \tag{19}$$

$$= \sum_r r\ p(r_0 = r|s_0 = s) \tag{20}$$

$$= E(r_0|s_0 = s) \tag{21}$$

- I assume $E[r_{t+n}|s_t = s] = E[r_n|s_0 = s]$ for all n. Then, I show that $E[r_{t+n+1}|s_t = s] = E[r_{n+1}|s_0 = s]$.

$$
\begin{aligned}
E\big[r_{t+n+1}|s_t = s\big] &= E\Big[E\big[r_{t+n+1}|s_t = s, s_{t+n+1} = s'\big]|s_t = s\Big] && (22)\\
&= E\Big[E\big[r_{t+n+1}|s_{t+n+1} = s'\big]|s_t = s\Big] && (23)\\
&= \sum_{s'} E\big[r_{t+n+1}|s_{t+n+1} = s'\big]P(s_{t+n+1} = s'|s_t = s) && (24)\\
&= \sum_{s'} E\big[r_{t+n+1}|s_{t+n+1} = s'\big]P(s_{n+1} = s'|s_0 = s) && (25)\\
&= \sum_{s'} E\big[r_{n+1}|s_{n+1} = s'\big]P(s_{n+1} = s'|s_0 = s) && (26)\\
&= E\big[r_{n+1}|s_0 = s\big] && (27)
\end{aligned}
$$

Equation 22 does not come from the law of iterated expectation (as most of the papers/proofs I have seen), but rather from one form of the **tower property** for random variables, which states that

$$E\Big[E\big[X|Y, Z\big]|Y\Big] = E\big[X|Y\big], \tag{28}$$

whereas the law of iterated expectation is

$$E\Big[E\big[X|Z\big]\Big] = E\big[X\big]. \tag{29}$$

(2) Finally, I conclude that $V_{t+n}(s) = V_t(s)$

$$
\begin{aligned}
V_{t+n}(s) &= E\Big[G_{t+n}|s_{t+n} = s\Big] && (30)\\
&= E\Big[\sum_{k=0}^{\infty}\gamma^k r_{t+n+k}\Big|s_{t+n} = s\Big] && (31)\\
&= \sum_{k=0}^{\infty}\gamma^k E\big[r_{t+n+k}\Big|s_{t+n} = s\big] && (32)\\
&= \sum_{k=0}^{\infty}\gamma^k E\big[r_{t+k}\Big|s_t = s\big] && (33)\\
&= V_t(s) && (34)
\end{aligned}
$$

2.2 Bellman Equations

For an **infinite horizon** MRP, the value function $V_t(s)$ can be decomposed into two parts: (1) an immediate reward r_t, and (2) a discounted value of successor state $\gamma V_{t+1}(s')$:

$$V(s) = R(s) + \gamma\sum_{s'} V(s')P_{ss'} \tag{35}$$

Proof.

$$
\begin{aligned}
V_t(s) &= E[G_t|s_t = s] && (36)\\
&= E[r_t + \gamma G_{t+1}|s_t = s] && (37)\\
&= R(s) + \gamma E[G_{t+1}|s_t = s] && (38)\\
&= R(s) + \gamma E\Big[E\big[G_{t+1}|s_t = s, s_{t+1} = s'\big]|s_t = s\Big] && (39)\\
&= R(s) + \gamma E\Big[E\big[G_{t+1}|s_{t+1} = s'\big]|s_t = s\Big] && (40)\\
&= R(s) + E\Big[\gamma V_{t+1}(s')|s_t = s\Big] && (41)\\
&= R(s) + \gamma \sum_{s'} V_{t+1}(s')P(s_{t+1} = s'|s_t = s). && (42)
\end{aligned}
$$

If we assume the transition probability and the rewards to be stationary, and if H is infinite, we then have

$$V(s) = R(s) + \gamma \sum_{s'} V(s')P_{ss'} \tag{43}$$

Remark on the proof:

- We used the fact that $G_t = r_t + \gamma G_{t+1}$
- Equation 39 is obtained by using the tower property (equation 28)
- We used the fact that $V_t(s) = V_{t+1}(s) = V(s)$ (Lemma 2.1)

Additionally, for $n = |S| < \infty$, equation 35 can be written as linear system of equations,

$$\mathbf{V} = \mathbf{R} + \gamma \mathbf{V}\mathbf{P}, \tag{44}$$

where $\mathbf{V} \in \mathbb{R}^n$ and $\mathbf{R} \in \mathbb{R}^n$ are the value-function and expected rewards vectors, respectively. $\mathbf{P} \in \mathbb{R}^{n\times n}$ is the transition probability matrix, where $(\mathbf{P})_{i,j} = P_{ij} = p(s_{t+1} = s_j|s_t = s_i)$. Equation can also be written as
which can also be written as

$$\begin{bmatrix} v(s_1) \\ \vdots \\ v(s_n) \end{bmatrix} = \begin{bmatrix} R_{s_1} \\ \vdots \\ R_{s_n} \end{bmatrix} + \gamma \begin{bmatrix} P_{s_1s_1} & \dots & P_{s_1s_n} \\ \vdots & \ddots & \vdots \\ P_{s_ns_1} & \dots & P_{s_ns_n} \end{bmatrix} \begin{bmatrix} v(s_1) \\ \vdots \\ v(s_n) \end{bmatrix}. \tag{45}$$

For $\gamma < 1$, $(\mathbf{I} - \gamma\mathbf{P})$ is invertible and Equation 44 yields the following analytical solution

$$\mathbf{V} = (\mathbf{I} - \gamma\mathbf{P})^{-1}\mathbf{R}. \tag{46}$$

Proof. We show that for $0 \leq \gamma < 1$, $(\mathbf{I} - \gamma\mathbf{P})$ is invertible.

1. We first prove that $\mathbf{P}$ has an eigenvalue equal to 1
2. We show that any eigenvalue λ of $\mathbf{P}$ is such that $|\lambda| < 1$
3. We conclude that $\mathbf{I} - \gamma\mathbf{P}$ is invertible

1. $\mathbf{P}$ is a row stochastic matrix: $\sum_{j=1}^{|S|} P_{ij} = 1$, and $\lambda = 1$ is an eigenvalue of $\mathbf{P}$ with a corresponding eigenvector $\mathbf{v} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$

$$\mathbf{P}\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \tag{47}$$

2. Let λ be an eigenvalue of $\mathbf{P}$ with a corresponding eigenvector $\mathbf{v}$. Then, $\mathbf{Pv} = \lambda\mathbf{v}$. By examining the i^{th} row, we can write

$$\sum_{j=1}^{|S|} P_{ij} v_j = \lambda v_i \tag{48}$$

Let $|v_k| = \max_q |v_q|$. Since $\mathbf{v}$ is an eigenvector, we have $|v_k| > 0$. Therefore

$$\begin{aligned} |\lambda||v_k| &= |\sum_{j=1}^{|S|} P_{ij} v_j| && (49)\\ &\leq \sum_{j=1}^{|S|} P_{ij}|v_j| && (50)\\ &\leq |v_k| \sum_{j=1}^{|S|} P_{ij} && (51)\\ &= |v_k| && (52) \end{aligned}$$

Therefore, $|\lambda| \leq 1$ 3. Let us assume that $\lambda = 0$ is an eigenvalue of $\mathbf{I} - \gamma\mathbf{P}$. Then, there exists $\mathbf{v} \neq \mathbf{0}$ such that

$$\begin{aligned} (\mathbf{I} - \gamma\mathbf{P})\mathbf{v} &= \mathbf{0} && (53)\\ \gamma\mathbf{P}\mathbf{v} &= \mathbf{v} && (54) \end{aligned}$$

and thus $\frac{1}{\gamma} > 1$ is an eigenvalue of $\mathbf{P}$, which contradicts our previous result.

3 Markov Decision Process

3.1 Definitions

Definition 3.1 (**Markov Decision Process**). A Markov Decision Process (MDP) is a tuple $\big(S, A, P, R, \gamma\big)$, where

- S is the finite state-space of the Markov process (assume $|S| < \infty$)
- A is the finite action-space available from each state s
- P is the state transition probability model where $P^a_{ss'} = P(s_{t+1} = s' | s_t = s, a_t = a)$
- $R : S \times A \mapsto \mathbb{R}$ is a reward function that maps states to rewards, $R(s, a) = E[r_t | s_t = s, a_t = a]$
- $\gamma \in [0, 1]$ is a discount factor

The basic model of the dynamics is that there is a state space S, and an action space A, both of which we will consider to be finite. The agent starts from a state s_t at time t, chooses an action a_t from the action space, obtains a reward r_t and then reaches a successor state s_{t+1}. An episode of a MDP is thus represented as $(s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, ...)$.

Unlike in the case of a Markov Process or a Markov Reward Process where the transition probability was only a function of the successor state and the current state, the transition probabilities for a MDP at time t are a function of the successor state s_{t+1} along with both the current state s_t and the action a_t, written as $P^a_{ss'} = P(s_{t+1} = s' | s_t = s, a_t = a)$. We still assume the principle of stationary transition probabilities which in the context of a MDP is written mathematically as

$$P^a_{ss'} = P(s_{i+1} = s' | s_i = s, a_i = a) = P(s_{j+1} = s' | s_j = s, a_j = a) \tag{55}$$

Additionally, the reward r_t at time t depends on both s_t and s_t, in contrast to a Markov Reward Process where it depended only on the current state. These rewards can be stochastic or deterministic, but just like in the case of a Markov reward process, we will assume that the rewards are stationary and the only relevant quantity will be the expected reward which we will denote by $R(s,a)$ for a fixed state s and action a, and defined below:

$$R(s,a) = E[r_t|s_t = s, a_t = a] \tag{56}$$

Definition 3.2 (**Policy for MDPs**). A policy specifies what action to take in each state of a MDP and fully defines the behavior of an agent. Policies can either be deterministic or stochastic. To cover both these cases, we will consider a policy to be a probability distribution over actions given the current state:

$$\pi_t(a|s) = P(a_t = a|s_t = s) \tag{57}$$

The policy may be varying with time, which is especially true in the case of finite horizon MDPs. We will denote a generic policy by π, defined as the infinite dimensional tuple $\pi = (\pi_0, \pi_1, ...)$, where π_t refers to the policy at time t. We will call policies that do not vary with time "stationary policies", and indicate them as π, i.e. in this case $\pi = (\pi, \pi, ...)$.

Remark: Given a MDP $M = \big(S, A, P, R, \gamma\big)$ and a policy π:

- The state sequence $(s_0, s_1, ...)$ is a Markov Process (S, P^π)
- The state/reward sequence $(s_0, r_0, s_1, r_1, ...)$ is a Markov reward process $(S, R^\pi, P^\pi, \gamma)$

Additionally, the transition probability matrix and the reward functions are given by

$$P^\pi_{ss'} = \sum_{a\in A} P(s_{t+1} = s'|s_t = s, a_t = a)\pi(a_t = a|s_t = s) \tag{58}$$

$$R^\pi(s) = \sum_{a\in A} R(s_t = s, a_t = a)\pi(a_t = a|s_t = s) \tag{59}$$

Definition 3.3 (**State value function for a MDP**). The state-value function $V^\pi(s)$ of a MDP is the expected return starting from state s, and then following policy π

$$V^\pi_t(s) = E_\pi[G_t|s_t = s], \tag{60}$$

where E_π denotes the expected value of a random variable given that the agent follows policy π. The value of the terminal state (if any) is always zero.

Definition 3.4 (**State-action value function for a MDP**). The action-value function Q^π of an MDP is the expected return starting from state s, taking action a, and then following policy π

$$Q^\pi_t(s) = E_\pi[G_t|s_t = s, a_t = a], \tag{61}$$

In a similar fashion as for the value-function, we can show using the stationarity and finite-horizon assumptions that $Q^\pi_i(s) = Q^\pi_j(s)$.

3.2 Bellman Equations

For an infinite horizon MDP, the Bellman recursive equations for the **state-value** functions are given by

$$V^\pi(s) = E_\pi[r_t + \gamma V^\pi(s_{t+1})|s_t = s] \tag{62}$$

$$V^\pi(s) = R^\pi(s) + \gamma \sum_{s'} V^\pi(s')P^\pi_{ss'} \tag{63}$$

$$V^\pi(s) = \sum_a \pi(a|s)Q^\pi(s,a) \tag{64}$$

$$V^\pi(s) = \sum_a \pi(a|s)\big[R(s,a) + \gamma \sum_{s'} P^a_{ss'}V^\pi(s')\big] \tag{65}$$

The Bellman recursive equations for the **action-value** functions are given by

$$
\begin{aligned}
Q^{\pi}(s,a) &= E_{\pi}[r_t + \gamma Q^{\pi}(s_{t+1}, a_{t+1})|s_t = s, a_t = a] && (66)\\
Q^{\pi}(s,a) &= R(s,a) + \gamma \sum_{s'} P^{a}_{ss'} \sum_{a'} \pi(a'|s') Q^{\pi}(s',a') && (67)\\
Q^{\pi}(s,a) &= R(s,a) + \gamma \sum_{s'} P^{a}_{ss'} V^{\pi}(s') && (68)
\end{aligned}
$$

Proof

- Proof of equation 62

$$
\begin{aligned}
V^{\pi}_t(s) &= E[G_t|s_t = s] && (69)\\
&= E[r_t + \gamma G_{t+1}|s_t = s] && (70)\\
&= E[r_t + \gamma E[G_{t+1}|s_{t+1}]|s_t = s] && (71)\\
&= E[r_t + \gamma V^{\pi}(s_{t+1})|s_t = s] && (72)\\
V^{\pi}(s) &= E[r_t + \gamma V^{\pi}(s_{t+1})|s_t = s] && (73)
\end{aligned}
$$

- Equation 63 can be derived in a similar way as equation 35.
- Proof of equation 64:

$$
\begin{aligned}
V^{\pi}_t(s,a) &= E[G_t|s_t = s] && (74)\\
&= \sum_g g\, p(G_t = g|s_t = s) && (75)\\
&= \sum_g g \sum_a p(G_t = g, a_t = a|s_t = s) && (76)\\
&= \sum_g g \sum_a p(G_t = g|s_t = s, a_t = a)\pi(a_t = a|s_t = s) && (77)\\
&= \sum_a \pi(a_t = a|s_t = s) \sum_g g\, p(G_t = g|s_t = s, a_t = a) && (78)\\
&= \sum_a \pi(a|s) E[G_t|s_t = s, a_t = a] && (79)\\
&= \sum_a \pi(a|s) Q^{\pi}(s,a) && (80)
\end{aligned}
$$

- Equation 66 can be obtained with a similar proof as equation 62.
- We now prove equation 67.

$$
\begin{aligned}
Q^{\pi}_t(s,a) &= E[G_t|s_t = s, a_t = a] && (81)\\
&= R(s,a) + \gamma E[G_{t+1}|s_t = s, a_t = a] && (82)\\
&= R(s,a) + \gamma E\Big[E\big[G_{t+1}|s_{t+1} = s', a_{t+1} = a'\big]|s_t = s, a_t = a\Big] && (83)\\
&= R(s,a) + \gamma E\Big[Q^{\pi}_{t+1}(s',a')|s_t = s, a_t = a\Big] && (84)\\
&= R(s,a) + \gamma \sum_{s',a'} Q^{\pi}_{t+1}(s',a') p(s_{t+1} = s', a_{t+1} = a'|s_t = s, a_t = a) && (85)\\
&= R(s,a) + \gamma \sum_{s',a'} Q^{\pi}_{t+1}(s',a') p(a_{t+1} = a'|s_{t+1} = s') p(a_{t+1} = s'|s_t = s, a_t = a). && (86)
\end{aligned}
$$

We can then conclude that

$$
Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s'} P^{a}_{ss'} \sum_a Q^{\pi}(s',a')\pi(a'|s') \quad (87)
$$

- Equation 65 is obtained by replacing the expression of $V^\pi(s)$ from equation 68 into equation 64
- Finally, equation 65 is obtained by replacing the expression of $Q^\pi(s,a)$ found in equation 68 into equation 64

4 MDP Planning by Dynamic Programming

Important assumption. In this section, we assume that the dynamics of the world P and R are known and given to us. We will use these functions to perform two tasks,

- **Prediction** where the input is (S, A, P, R, γ) and π, and the output is the state-value function V^π
- **Control** where the input is (S, A, P, R, γ) and the output is the optimal state-value function V^* and an optimal policy π^*

4.1 Policy evaluation (prediction)

Policy evaluation is the process of computing the value of $V^\pi(s)$ for all $s \in S$ given a fixed policy π.

Algorithm 1: Policy evaluation (PE)

---

1. Initialize $V(s) = 0$
2. For $k = 1$ until convergence,
   - For all $s \in S$,

$$V_k^\pi(s) = R^\pi(s) + \gamma \sum_{s'} P_{ss'}^\pi V_{k-1}^\pi(s') \tag{88}$$

Remarks:

- The computational cost of algorithm 1 is $O(|S|^2)$
- By defining the Bellman backup operator B^π for a policy π, we can show that B^π is a contraction mapping (for any norm since we are considering finite-dimensional vector spaces) for $\gamma \leq 1$. That is,

$$||B^\pi V - B^\pi V'|| \leq ||V - V'||. \tag{89}$$

  This result implies that B^π has a unique fixed point, and Policy evaluation amounts to computing the fixed point of B^π. To do policy evaluation, we repeatedly apply operator until V^π stops changing.

4.2 Optimal value function

Definition 4.1 (**Optimal state-value function**)**.** The optimal state-value function $V^*(s)$ is the maximum value function over all policies

$$V^*(s) = \max_\pi V^\pi(s). \tag{90}$$

Definition 4.2 (**Optimal action-value function**)**.** The optimal action-value function $q_*(s,a)$ is the maximum action-value function over all policies

$$Q^*(s) = \max_\pi Q^\pi(s,a). \tag{91}$$

The optimal value-functions specify the best possible performance in the MDP. In addition, an MDP is "solved" when we know the optimal value function.

4.3 Optimal policy

We can define partial ordering over policies as follows

$$\pi \geq \pi' \text{ if } \mathrm{V}^{\pi}(\mathrm{s}) \geq \mathrm{V}^{\pi'}(\mathrm{s}),\ \forall \mathrm{s} \in \mathrm{S} \tag{92}$$

Theorem.

- For any MDP, there exists an optimal policy π^* that is better or equal to all other policies, $\pi^* \geq \pi$ for all π

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, V^{\pi}(s) \tag{93}$$

- All optimal policies achieve the optimal state-value function, $V^{\pi^*}(s) = V^*(s)$ for all s
- All optimal policies achieve the optimal action-value function, $Q^{\pi^*}(s,a) = Q^*(s,a)$ for all s and a

Additionally, an optimal policy can be found by maximizing over $Q^*(s,a)$,

$$\pi^*(a|s) = 1 \text{ if } \mathrm{a} = \underset{\mathrm{a}}{\operatorname{argmax}}\, \mathrm{Q}^*(\mathrm{s},\mathrm{a}) \tag{94}$$

$$= 0 \text{ otherwise} \tag{95}$$

Therefore, once we know $Q^*(s,a)$, we immediately have the optimal policy.
Property:

- If H is infinite, $|S| < \infty$, then there is exist a deterministic stationary optimal policy, but not necessarily unique
- There exist a unique optimal value function $V^* = V^{\pi^*}$
- The number of deterministic policies is $|A|^{|S|}$

4.4 Bellman optimality equations

The four recursive equations are given by

$$V^*(s) = \max_a Q^*(s,a) \tag{96}$$

$$Q^*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V^*(s') \tag{97}$$

$$V^*(s) = \max_a \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V^*(s') \right) \tag{98}$$

$$Q^*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \max_{a'} Q^*(s',a') \tag{99}$$

Remark.

- From equation 98, we can define the Bellman optimality backup operator B :, which is applied to a value function and returns a new value function as follows,

$$BV(s) = \max_a \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V(s') \right). \tag{100}$$

  Similarly as for B^{π}, it can be shown that B is a contraction mapping for $\gamma \leq 1$, which implies that the solution of equation 98 exists and is unique.

- Consequently, if you find a policy π that satisfy any of the Bellman optimality equations, then π is the optimal policy. For instance, if we have π such that for all $s \in S$ and $a \in A$,

$$V^{\pi}(s) = \max_a Q^{\pi}(s,a), \tag{101}$$

  then the policy π is optimal, $\pi = \pi^*$.

4.5 Control

The goal is to find the optimal deterministic policy for a MDP in a finite-dimensional (tabular) space.

4.5.1 Policy iteration

The goal of this algorithm is to iteratively compute an optimal deterministic policy in an infinite horizon finite-state MDP. It is composed of two steps as described in algorithm 2. Step 1 of algorithm 2 is performed by applying algorithm 1. We describe step 2 (referred to as policy improvement) in more details in the next section.

Algorithm 2: Policy iteration (PI)

1. Set $i = 0$
2. $\pi_0(s)$ randomly initialized
3. While $i = 0$ or $||\pi_i - \pi_{i+1}||_1 > 0$
   - **Step 1**: Evaluate V^{π_i} with policy evaluation
   - **Step 2**: Perform policy improvement: $\pi_i \rightarrow \pi_{i+1}$
   - $i = i + 1$

4.5.2 Policy improvement

In step 2 of algorithm 2, we improve the policy by acting greedily,

$$\pi_{i+1}(s) = \max_a Q^{\pi_i}(s, a), \tag{102}$$

In that case, it can be shown that this way of choosing π_{i+1} from π_i ensures monotonic improvement in policy:

$$\pi_{i+1} \geq \pi_i \tag{103}$$

Remark.

- If the policy does not change at a given iteration, then it cannot not change again, and the algorithm has converged to the optimal policy π^*
- There is a maximum of $|A|^{|S|}$ iterations

Proof.
(1) We prove that step 2 of algorithm 2 ensures a monotonic policy improvement.

- We begin by showing that

$$\max_a Q^{\pi_i}(s, a) \geq V^{\pi_i}(s), \tag{104}$$

  which is true because

$$\max_a Q^{\pi_i}(s, a) \geq \max_a Q^{\pi_i}(s, \pi(s)) = V^{\pi_i}(s). \tag{105}$$

- We use the following recursive Bellman equations:

$$Q^{\pi_i}(s, a) = R(s, a) + \gamma \sum_{s'} P^a_{ss'} V^{\pi_i}(s') \tag{106}$$

$$Q^{\pi_i}(s, a) = E[r_t + \gamma Q^{\pi_i}(s_{t+1}, a_{t+1})|s_t, a_t] \tag{107}$$

$$Q^{\pi_i}(s, \pi_{i+1}(s)) = E[r_t + \gamma V^{\pi_i}(s_{t+1})|s_t, \pi_{i+1}(s_t)]. \tag{108}$$

We show that $V^{\pi_i}(s) \leq V^{\pi_{i+1}}(s)$:

$$\begin{aligned} V^{\pi_i}(s_t) &= Q^{\pi_i}(s_t, \pi_i(s_t)) && (109)\\ &\leq Q^{\pi_i}(s, \pi_{i+1}(s_t)) && (110)\\ &= E[r_t + \gamma Q^{\pi_i}(s_{t+1}, \pi_i(s_{t+1}))|s_t, \pi_{i+1}(s_t)] && (111)\\ &= E[r_t + \gamma V^{\pi_i}(s_{t+1})|s_t, \pi_{i+1}(s_t)] && (112)\\ &= E_{\pi_{i+1}}[r_t + \gamma V^{\pi_i}(s_{t+1})|s_t] && (113)\\ &= E_{\pi_{i+1}}[r_t|s_t] + E_{\pi_{i+1}}[\gamma V^{\pi_i}(s_{t+1})|s_t] && (114)\\ & && (115) \end{aligned}$$

We can now apply the analogous inequality to $V^{\pi_i}(s_{t+1})$,

$$V^{\pi_i}(s_{t+1}) \leq E_{\pi_{i+1}}[r_{t+1}|s_{t+1}] + E_{\pi_{i+1}}[\gamma V^{\pi_i}(s_{t+2})|s_{t+1}]. \quad (116)$$

Therefore,

$$\begin{aligned} V^{\pi_i}(s_t) &\leq E_{\pi_{i+1}}[r_t|s_t] + E_{\pi_{i+1}}[\gamma E_{\pi_{i+1}}[r_{t+1}|s_{t+1}] + E_{\pi_{i+1}}[\gamma V^{\pi_i}(s_{t+2})|s_{t+1}]|s_t] && (117)\\ &= E_{\pi_{i+1}}[r_t|s_t] + \gamma E_{\pi_{i+1}}[E_{\pi_{i+1}}[r_{t+1}|s_{t+1}]|s_t] + \gamma^2 E_{\pi_{i+1}}[E_{\pi_{i+1}}[V^{\pi_i}(s_{t+2})|s_{t+1}]|s_t] && (118)\\ &= E_{\pi_{i+1}}[r_t|s_t] + \gamma E_{\pi_{i+1}}[r_{t+1}|s_t] + \gamma^2 E_{\pi_{i+1}}[V^{\pi_i}(s_{t+2})|s_t] && (119)\\ &= E_{\pi_{i+1}}[r_t + \gamma r_{t+1}|s_t] + \gamma^2 E_{\pi_{i+1}}[V^{\pi_i}(s_{t+2})|s_t] && (120)\\ &\leq ... && (121)\\ &= E_{\pi_{i+1}}[G_t|s_t] \\ &= V^{\pi_{i+1}}(s_t). && (122) \end{aligned}$$

(2) We show that if the policy does not change at a given iteration ($\pi_i = \pi_{i+1}$), it cannot change after that and the algorithm has converged to an optimal policy

- If $\pi_{i+1}(s) = \pi_i(s)$ for all s, then

$$\pi_{i+2}(s) = \underset{a}{\operatorname{argmax}}\, Q^{\pi_{i+1}}(s, a) = \underset{a}{\operatorname{argmax}}\, Q^{\pi_i}(s, a) = \pi_{i+1}(s) \quad (123)$$

- In such case, the policy is optimal,

$$\begin{aligned} V^{\pi_i}(s) &= Q^{\pi_i}(s, \pi_i(s)) && (124)\\ &= Q^{\pi_i}(s, \pi_{i+1}(s)) && (125)\\ &= \max_a Q^{\pi_i}(s, a) && (126) \end{aligned}$$

We found a policy π_i such that $V^{\pi_i}(s) = Q^{\pi_i}(s, a)$ for all $s \in S$, which implies that π_i is optimal.

4.5.3 Value iteration

Value iteration (algorithm 3) is another technique aimed at computing the optimal value and an optimal policy. This technique consists building a sequence of value-functions $V_1, V_2, ..., V_k, ...$ that converges towards the optimal value function V^*. This process can be seen as iteratively applying the Bellman optimality backup operator until convergence, and equation 127 can be seen as $V_{k+1} = BV_k$.

Unlike policy iteration, no explicit policy is constructed throughout the iterative. However, once we obtain V^*, we can compute the optimal policy using the following Bellman optimality equations,

$$\begin{aligned} \pi^*(s) &= \underset{a}{\operatorname{argmax}}\, Q^*(s, a) && (128)\\ &= \underset{a}{\operatorname{argmax}} \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V^*(s') \right) && (129) \end{aligned}$$

Algorithm 3: Value iteration (VI)

1. Set $k = 1$
2. $V_0(s) = 0$ for all $s \in S$
3. Loop until convergence criterion (or until end of episode)
   - For all $s \in S$

$$V_{k+1}(s) = \max_a \Big(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_k(s') \Big). \tag{127}$$

   - $k = k + 1$

4.5.4 Contraction mapping

We prove that the Bellman optimality backup operator is a contraction mapping if $\gamma \leq 1$. To do so, we use the following property,

$$|\max_x F(x) - \max_x G(x)| \leq \max_x |F(x) - G(x)|. \tag{130}$$

We now show that B is a contraction with the infinity norm (all norms in finite-dimensional vector spaces are equivalent) by proving the following inequality:

$$||BV - BV'||_\infty \leq \gamma ||V - V'||_\infty, \tag{131}$$

where $||F|| = \max_x F(x)$.

$$\begin{aligned}
|BV(s) - BV'(s)| &= \Big| \max_a \Big(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V(s') \Big) - \max_a \Big(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V'(s') \Big) \Big| && (132)\\
&\leq \max_a \Big| R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V(s') - \Big(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V'(s') \Big) \Big| && (133)\\
&= \gamma \max_a \Big| \sum_{s' \in S} P_{ss'}^a \big(V(s') - V'(s') \big) \Big| && (134)\\
&\leq \gamma \max_a \sum_{s' \in S} P_{ss'}^a \max_{s'} |V(s') - V'(s')| && (135)\\
&= \gamma \max_{s'} |V(s') - V'(s')| \, \big(\max_a \sum_{s' \in S} P_{ss'}^a \big) && (136)\\
&= \gamma \max_{s'} |V(s') - V'(s')| && (137)\\
&= \gamma \, ||V - V'||_\infty. && (138)
\end{aligned}$$

Therefore,

$$\begin{aligned}
|BV(s) - BV'(s)| &\leq \gamma \, ||V - V'||_\infty && (139)\\
\max_s |BV(s) - BV'(s)| &\leq \gamma \, ||V - V'||_\infty. && (140)
\end{aligned}$$

Finally,

$$||BV(s) - BV'(s)||_\infty \leq \gamma \, ||V - V'||_\infty. \tag{141}$$

5 Exercises