Reinforcement Learning

Markov Decision Process

Jordi Casas Roma

jordi.casas.roma@uab.cat

September 12, 2025

Universitat Autònoma de Barcelona

Table of Contents

Markov Process
Definition

A **Markov process** (also called **Markov chain**) is a random process (random phenomenon that changes over time) that is characterized by the following elements:

- State S_t
- Markov property
- Probability of transition between states $\mathcal{P}_{ss'}$ or $p(s'|s)$
- State transition matrix $\mathcal{P}$ or $\mathbf{P}$

Markov Process
Markov Property

Markov Property

A state S_t is called **Markov** (or **Markovian**) if and only if:

$$Pr\{S_{t+1}|S_t\} = Pr\{S_{t+1}|S_1, \ldots, S_t\}$$

- *"The future is independent of the past given the present"*.
- That is, the entire future history depends on the current state S_t and not on previous states that can therefore be discarded.

Markov Process
Transition probability

Given a Markov state $S_t = s$ and its successor state at time $S_{t+1} = s'$, the **probability of transition** from one state to another is defined as:

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

- The meaning of this transition probability can be seen as the probability of ending up in the state $S_{t+1} = s'$ from the state $S_t = s$.

Markov Process
State Transition Matrix

We can define the **state transition matrix** $\mathcal{P}$ (or $\mathbf{P}$) that contains the transition probabilities from all states $S_t = s$ to all their possible successor states $S_{t+1} = s'$.

- The matrix size is $n \times n$ (n is the number of possible states)

$$\mathcal{P} = \mathbf{P} = \begin{pmatrix} \mathcal{P}_{11} & \mathcal{P}_{12} & \cdots & \mathcal{P}_{1n} \\ \mathcal{P}_{21} & \mathcal{P}_{22} & \cdots & \mathcal{P}_{2n} \\ \vdots & \vdots & & \vdots \\ \mathcal{P}_{n1} & \mathcal{P}_{n2} & \cdots & \mathcal{P}_{nn} \end{pmatrix} \tag{1}$$

- Row: output probabilities
  - $\forall k$ it holds that $\sum_{i=1}^{n} \mathcal{P}_{ki} = 1$
- Column: input probabilities

Markov Process
Definition

Markov Process

A **Markov Process** (or **Markov Chain**) is defined by the tuple $< \mathcal{S}, \mathcal{P} >$, where:

- $\mathcal{S}$: finite set of Markov states
- $\mathcal{P}$ or $\mathbf{P}$: state transition probability matrix, where the elements are $\mathcal{P}_{ss'} = p(s'|s) = Pr\{S_{t+1} = s'|S_t = s\}$

- The **stationarity property** must be satisfied, which imposes that the transition probabilities $\mathcal{P}_{ss'} = p(s'|s)$ must remain constant over time.

Markov Process

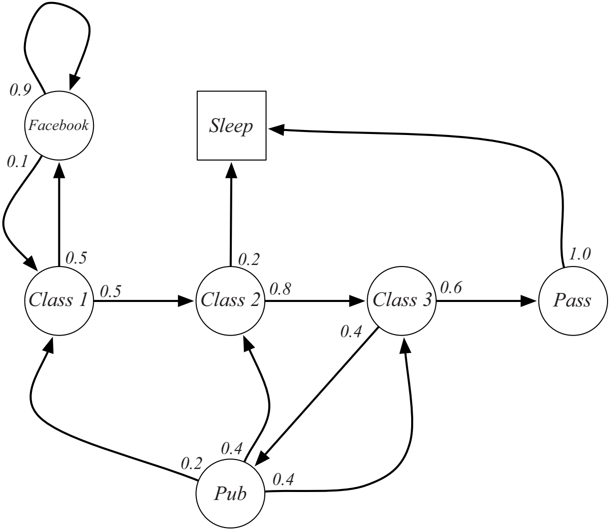
Example: Markov process of a student's daily life

The (7) states are defined as follows:

- $C1 = Class1$, **initial state** that represents the first class.
- $C2 = Class2$, state that represents the second class.
- $C3 = Class3$, state that represents the third class.
- $FB = Facebook$, a state that represents that the student connects to his Facebook (for example because he is bored with the subject).
- *Pub*, state that represents that the student goes to the bar.
- *Pass*, state that represents that the student has finished classes and is going home.
- *Sleep* **terminal state** that represents that the student is going to sleep.

Markov Process

Example: Markov process of a student's daily life



Markov Process

Example: Markov process of a student's daily life

We can represent the student's evolution through a sequence of states that always begins with the **initial state** $S_1 = C1$ and ends with the **terminal state** $S_T = Sleep$.

- Each of these sequences is called an **episode**.

Below are some examples of episodes:

- C1, FB, C1, C2, Sleep
- C1, FB, FB, C1, C2, C3, Pass, Sleep
- C1, C2, C3, Pub, C2, Sleep
- C1, FB, FB, C1, C2, C3, Pub, C1, FB, FB, C1, C2, Sleep

Markov Process

Example: Markov process of a student's daily life

The **state transition matrix** for this example is defined as:

$$
\mathcal{P} = \begin{pmatrix}
\mathcal{P}_{C1C1} & \mathcal{P}_{C1C2} & \mathcal{P}_{C1C3} & \mathcal{P}_{C1Pass} & \mathcal{P}_{C1Pub} & \mathcal{P}_{C1FB} & \mathcal{P}_{C1Sleep} \\
\mathcal{P}_{C2C1} & \mathcal{P}_{C2C2} & \cdots & & & & \\
\mathcal{P}_{C3C1} & \mathcal{P}_{C3C2} & \cdots & & & & \\
\mathcal{P}_{PassC1} & \mathcal{P}_{PassC2} & \cdots & & & & \\
\mathcal{P}_{PubC1} & \mathcal{P}_{PubC2} & \cdots & & & & \\
\mathcal{P}_{FBC1} & \mathcal{P}_{FBC2} & \cdots & & & & \\
\mathcal{P}_{SleepC1} & \mathcal{P}_{SleepC2} & \cdots & & & &
\end{pmatrix}
$$

Markov Process

Example: Markov process of a student's daily life

...and with numerical values:

$$\mathcal{P} = \begin{pmatrix} 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0.8 & 0 & 0 & 0 & 0.2 \\ 0 & 0 & 0 & 0.6 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.0 \\ 0.2 & 0.4 & 0.4 & 0 & 0 & 0 & 0 \\ 0.1 & 0 & 0 & 0 & 0 & 0.9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.0 \end{pmatrix}$$

- It is important to note that the final state *Sleep* cannot lead to another state, so all the transition probabilities from that state $p(s'|Sleep)$ are null, except to itself $p(Sleep|Sleep) = 1$.

Table of Contents

Markov Reward Process

Definition

A **Markov Reward Process (MRP)** is a Markov process that includes a scalar signal, called reward R_t, associated with each state.

Markov Reward Process

A Markov Reward Process is defined by the tuple $< \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma >$, where:

- $\mathcal{S}$: finite set of Markov states.
- $\mathcal{P}$ or $\mathbf{P}$: state transition probability matrix, where the elements are $\mathcal{P}_{ss'} = p(s'|s) = Pr\{S_{t+1} = s'|S_t = s\}$.
- $\mathcal{R}$: is a reward function that allows us to define the average reward of a state $r(s) = \mathbb{E}[R_{t+1}|S_t = s]$.
- γ: it is discount factor ($\gamma \in [0, 1]$).

Markov Reward Process
Return

The **accumulated reward**, also called **return**, can be defined for time t as:

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

- The **return** signal G_t is defined as the accumulated value of all future rewards.
- Note that since these rewards are random, the return is also random.
- The parameter γ (**discount factor**, $0 \leq \gamma \leq 1$) allows the convergence of the return value in problems where the number of future rewards is infinite.

Markov Reward Process

Return

The **accumulated reward** (**return** can be expressed recursively as follows::

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \dots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Markov Reward Process
Return

The **accumulated reward** (**return** can be expressed recursively as follows::

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots \\ &= R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \dots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Thus, **we can express the return** at the instant t from the immediate reward R_{t+1} and the return at the next instant of time G_{t+1}:

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

- This recursion is the basis of the Bellman equations.

Markov Reward Process

Return

The **discount factor** (γ) values:

- If $\gamma \simeq 0$ only short-term rewards matter, since $G_t \simeq R_{t+1}$.
- If $\gamma \simeq 1$ long-term rewards are just as important as short-term rewards. This is what is known as an “optimistic” long-term return.
- Usually, γ values are in range $0 < \gamma < 1$ to avoid these extreme cases.

Markov Reward Process

Expected immediate reward

Expected immediate reward of a state:

- The expected immediate reward of a state s is defined as the expected value of the reward R_{t+1} if we start from the state $S_t = s$

$$
\begin{aligned}
r(s) &= \mathbb{E}[R_{t+1} | S_t = s] \\
&= \sum_{\forall s' \in \mathcal{R}} r(s') Pr\{R_{t+1} = s' | S_t = s\} \\
&= \sum_{\forall s' \in \mathcal{R}} r(s') p(s' | s)
\end{aligned}
$$

- where $r(s')$ is the reward of a state s'.

Markov Reward Process

Definition of value function of a state

Definition of **value function of a state**:

- Immediate rewards are random, and therefore the return G_t is also random. Thus, we can calculate its expected value.
- This is what is known as **value function of a state** $v(s)$.

Definition

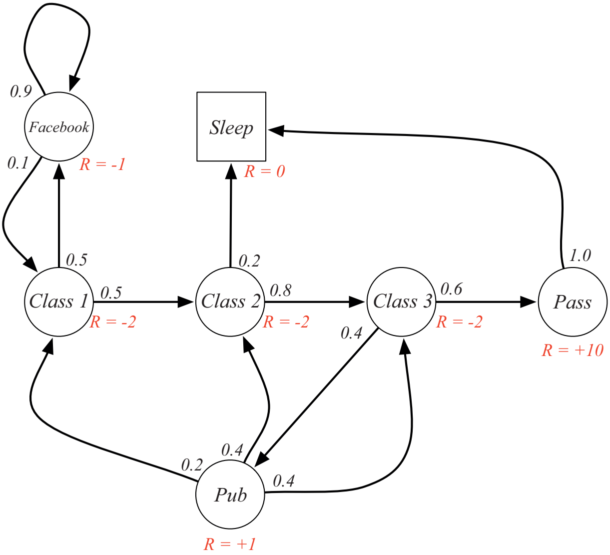
The value function of a state $v(s)$ for an MRP is the expected value of the return G_t if we start from a state $S_t = s$.

$$v(s) = \mathbb{E}[G_t | S_t = s]$$

- We can interpret the value function $v(s)$ as the long-term value of state s.

Markov Reward Process

Example: Markov reward process of a student's daily life



Markov Reward Process

Example: Markov reward process of a student's daily life

For example, if we want to calculate the return for the initial state $S_1 = C1$:

$$G_1 = R_2 + \gamma R_3 + \gamma^2 R_{t+3} + \ldots + \gamma^{T-2} R_T$$

- where $R_T = 0$ is the reward of the terminal state $S_T = Sleep$.

Markov Reward Process

Example: Markov reward process of a student's daily life

For example, if we want to calculate the return for the initial state $S_1 = C1$:

$$G_1 = R_2 + \gamma R_3 + \gamma^2 R_{t+3} + \ldots + \gamma^{T-2} R_T$$

- where $R_T = 0$ is the reward of the terminal state $S_T = Sleep$.

If we choose a discount factor $\gamma = \frac{1}{2}$, we can calculate the return in each of the following episodes:

C1 FB C1 C2 Sleep $\rightarrow$ $G_1 = -1 - 2 \times \frac{1}{2} - 2 \times \frac{1}{4} = -2.5$

Markov Reward Process

Example: Markov reward process of a student's daily life

For example, if we want to calculate the return for the initial state $S_1 = C1$:

$$G_1 = R_2 + \gamma R_3 + \gamma^2 R_{t+3} + \ldots + \gamma^{T-2} R_T$$

- where $R_T = 0$ is the reward of the terminal state $S_T = Sleep$.

If we choose a discount factor $\gamma = \frac{1}{2}$, we can calculate the return in each of the following episodes:

C1 FB C1 C2 Sleep	$\rightarrow$	$G_1 = -1 - 2 \times \frac{1}{2} - 2 \times \frac{1}{4} = -2.5$
C1 FB FB C1 C2 C3 Pass Sleep	$\rightarrow$	$G_1 = -1 - 1 \times \frac{1}{2} - 2 \times \frac{1}{4} - 2 \times \frac{1}{8} - \ldots = -2.0625$

Markov Reward Process

Example: Markov reward process of a student's daily life

For example, if we want to calculate the return for the initial state $S_1 = C1$:

$$G_1 = R_2 + \gamma R_3 + \gamma^2 R_{t+3} + \ldots + \gamma^{T-2} R_T$$

- where $R_T = 0$ is the reward of the terminal state $S_T = Sleep$.

If we choose a discount factor $\gamma = \frac{1}{2}$, we can calculate the return in each of the following episodes:

C1 FB C1 C2 Sleep	$\rightarrow$	$G_1 = -1 - 2 \times \frac{1}{2} - 2 \times \frac{1}{4} = -2.5$
C1 FB FB C1 C2 C3 Pass Sleep	$\rightarrow$	$G_1 = -1 - 1 \times \frac{1}{2} - 2 \times \frac{1}{4} - 2 \times \frac{1}{8} - \ldots = -2.0625$
C1 C2 C3 Pub C2 Sleep	$\rightarrow$	$G_1 = -2 - 2 \times \frac{1}{2} + 1 \times \frac{1}{4} - 2 \times \frac{1}{8} = -3$

- If we could simulate all possible episodes, calculate the corresponding returns and average the results, we would obtain the C1-state value function $v(C1)$.

Markov Reward Process
The Bellman Equation

The **Bellman equation** for the **value function** of an MRP:

$$\begin{aligned} v(s) &= \mathbb{E}[G_t | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \end{aligned}$$

Markov Reward Process

The Bellman Equation

The **Bellman equation** for the **value function** of an MRP:

$$
\begin{aligned}
v(s) &= \mathbb{E}[G_t | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s]
\end{aligned}
$$

This equation can be calculated, based on the expected immediate reward and state transition probabilities, as:

$$v(s) = r(s) + \gamma \sum_{\forall s' \in \mathcal{S}} p(s'|s) v(s')$$

- This equation is **very important**, since it allows us to decompose the value function of state s into the sum of two terms:
  1. the expected immediate reward of state s,
  2. the mean (weighted by the discount factor) of the value functions of all possible states immediately following s'.

Markov Reward Process

The Bellman Equation: How to solve it?

The **Bellman equation** can be expressed in matrix form:

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

- Where $\mathbf{v}$ is a column vector containing the value function of all possible states,
- $\mathbf{r}$ is a column vector whose elements are the expected immediate rewards of each state, and
- $\mathbf{P} = \mathcal{P}$ is the state transition matrix.

Markov Reward Process

The Bellman Equation: How to solve it?

The **Bellman equation** can be expressed in matrix form:

$$\mathbf{v} = \mathbf{r} + \gamma \mathbf{P} \mathbf{v} \tag{2}$$

- Where $\mathbf{v}$ is a column vector containing the value function of all possible states,
- $\mathbf{r}$ is a column vector whose elements are the expected immediate rewards of each state, and
- $\mathbf{P} = \mathcal{P}$ is the state transition matrix.

Markov Reward Process

The Bellman Equation: How to solve it?

For an **MRP with n states** we obtain:

$$\begin{pmatrix} v(1) \\ v(2) \\ \vdots \\ v(n) \end{pmatrix} = \begin{pmatrix} r(1) \\ r(2) \\ \vdots \\ r(n) \end{pmatrix} + \gamma \begin{pmatrix} \mathcal{P}_{11} & \mathcal{P}_{12} & \cdots & \mathcal{P}_{1n} \\ \mathcal{P}_{21} & \mathcal{P}_{22} & \cdots & \mathcal{P}_{2n} \\ \vdots & \vdots & & \vdots \\ \mathcal{P}_{n1} & \mathcal{P}_{n2} & \cdots & \mathcal{P}_{nn} \end{pmatrix} \begin{pmatrix} v(1) \\ v(2) \\ \vdots \\ v(n) \end{pmatrix}$$

Markov Reward Process

The Bellman Equation: How to solve it?

The matrix equation above is a **linear equation** that can be solved directly:

$$
\begin{aligned}
\mathbf{v} &= \mathbf{r} + \gamma \mathbf{P}\mathbf{v} \\
\mathbf{v} - \gamma \mathbf{P}\mathbf{v} &= \mathbf{r} \\
(\mathbf{I} - \gamma \mathbf{P})\mathbf{v} &= \mathbf{r}
\end{aligned}
$$

Resulting in:

$$\mathbf{v} = (\mathbf{I} - \gamma \mathbf{P})^{-1}\mathbf{r}$$

- where $\mathbf{I}$ is the identity matrix.

Markov Reward Process

The Bellman Equation: How to solve it?

The matrix equation above is a **linear equation** that can be solved directly:

$$\begin{aligned} \mathbf{v} &= \mathbf{r} + \gamma \mathbf{P} \mathbf{v} \\ \mathbf{v} - \gamma \mathbf{P} \mathbf{v} &= \mathbf{r} \\ (\mathbf{I} - \gamma \mathbf{P}) \mathbf{v} &= \mathbf{r} \end{aligned}$$

Resulting in:

$$\mathbf{v} = (\mathbf{I} - \gamma \mathbf{P})^{-1} \mathbf{r}$$

- where $\mathbf{I}$ is the identity matrix.

Attention

The inversion of a matrix of dimension $(n \times n)$ has a computational complexity of $O(n^3)$. Therefore, it is only possible if **the number of states** n **is relatively small**.

Table of Contents

Markov Decision Process
Definition

Markov Decision Process (MDP) = MRP + action

Markov Decision Process

A Markov Decision Process is defined by the tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$, where:

- $\mathcal{S}$: finite set of Markov states.
- $\mathcal{A}$: finite set of actions that the random variable A_t can take. If the set of actions changes based on the current state, then $\mathcal{A} = \mathcal{A}(s)$.
- $\mathcal{P}$ or $\mathbf{P}$: state transition probability matrix, where:

$$p(s', r|s, a) = Pr\{S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\}$$
$$or$$
$$p(s'|s, a) = Pr\{S_{t+1} = s'|S_t = s, A_t = a\}$$

- $\mathcal{R}$: is the finite set of all possible rewards.
- γ: it is discount factor ($\gamma \in [0, 1]$).

Markov Decision Process

Dynamics

The rewards R_t and the states S_t are **random variables (VA)** with well-defined probability functions that only depend on the preceding actions and states.

$$p(s', r|s, a) = Pr\{S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\} \tag{3}$$

- This function $p(s', r|s, a)$ can be understood as the probability of ending up in state s' and receiving the reward r starting from the preceding state s and executing the action a.

Markov Decision Process
Dynamics

The rewards R_t and the states S_t are **random variables (VA)** with well-defined probability functions that only depend on the preceding actions and states.

$$p(s', r|s, a) = Pr\{S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\} \tag{3}$$

- This function $p(s', r|s, a)$ can be understood as the probability of ending up in state s' and receiving the reward r starting from the preceding state s and executing the action a.

And this function satisfies the **following property**:

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

Markov Decision Process
Dynamics

From the equation 3 we can obtain any environment statistics, such as the **probability of state transition**:

$$\begin{aligned} p(s'|s,a) &= Pr\{S_{t+1} = s'|S_t = s, A_t = a\} \\ &= \sum_{\forall r \in \mathcal{R}} p(s', r|s,a) \end{aligned} \tag{4}$$

Markov Decision Process
Dynamics

Another commonly example derived from equation 3 is the **expected (immediate) reward for a given state-action pair**:

$$
\begin{aligned}
r(s,a) &= \mathbb{E}[R_{t+1} | S_t = s, A_t = a] \\
&= \sum_{\forall r \in \mathcal{R}} r \; p(r|s,a) \\
&= \sum_{\forall r \in \mathcal{R}} \sum_{\forall s' \in \mathcal{S}} r \; p(s', r|s,a)
\end{aligned} \tag{5}
$$

Table of Contents

Markov Decision Process

Policy of an MDP

Definition

A **policy** π is a correspondence between each of the possible states $S_t = s$, $\forall s \in \mathcal{S}$ and each possible action $A_t = a$, $\forall a \in \mathcal{A}(s)$.

This **correspondence** can be of two types:

- **Deterministic**:
  - $a = \pi(s)$
  - In this case, each state corresponds uniquely to a certain action.
- **Stochastic**:
  - $\pi(a|s) = Pr\{A_t = a | S_t = s\}$
  - It is the most common case. The policy gives us the probability of selecting a certain action $A_t = a$ if we are in state $S_t = s$.

Markov Decision Process
Policy of an MDP

From an MDP and its policy π:

- The **sequence of states** $\{S_0, S_1, S_2, \ldots\}$ constitutes a *Markov Process* (**MP**).
- The **sequence of states and rewards** $\{S_0, R_1, S_1, R_2, S_2, \ldots\}$ constitutes a *Markov Reward Process* (**MRP**), where the state transition probability is:

$$\mathcal{P}^{\pi}_{ss'} = p_{\pi}(s'|s) = \sum_{\forall a \in \mathcal{A}} \pi(a|s) p(s'|s,a) \tag{6}$$

and the expected immediate reward:

$$r_{\pi}(s) = \sum_{\forall a \in \mathcal{A}} \pi(a|s) r(s,a) \tag{7}$$

Markov Decision Process
Policy of an MDP

RL algorithms use the **experience** they accumulate through their interaction with the environment to change the agent's policy.

$$
\begin{aligned}
\mathcal{P}^{\pi}_{ss'} &= p_{\pi}(s'|s) \\
&= \sum_{\forall a \in \mathcal{A}} p(s', a|s) \\
&= \sum_{\forall a \in \mathcal{A}} \frac{p(s', a, s)}{p(s)} \\
&= \sum_{\forall a \in \mathcal{A}} \frac{p(s'|s, a)p(s, a)}{p(s)} \\
&= \sum_{\forall a \in \mathcal{A}} \pi(a|s)p(s'|s, a)
\end{aligned}
$$

Markov Decision Process

Policy of an MDP

RL algorithms use the **experience** they accumulate through their interaction with the environment to change the agent's policy.

$$
\begin{aligned}
r_\pi(s) &= \mathbb{E}_\pi[R_{t+1}|S_t = s] \\
&= \sum_{\forall r \in \mathcal{R}} \sum_{\forall a \in \mathcal{A}} \sum_{\forall s' \in \mathcal{S}} r \; p(a, r, s'|s) \\
&= \sum_{\forall a \in \mathcal{A}} \pi(a|s) \sum_{\forall r \in \mathcal{R}} \sum_{\forall s' \in \mathcal{S}} r \; p(r, s'|s, a) \\
&= \sum_{\forall a \in \mathcal{A}} \pi(a|s) \sum_{\forall r \in \mathcal{R}} r \; p(r|s, a) \\
&= \sum_{\forall a \in \mathcal{A}} \pi(a|s) r(s, a)
\end{aligned}
$$

Table of Contents

Markov Decision Process
Value functions of an MDP

State value function of an MDP:

Definition

The **state value function** of an MDP $v_\pi(s)$ tells us how good it is, for the agent, to be in a certain state in terms of expected future rewards or, specifically, in terms of the expected return.

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] \tag{8}$$

- This function calculates the expected return if we start from a state s and follow a policy π from that moment on.

Markov Decision Process

Value functions of an MDP

Value function of an action (or a state-action pair) of an MDP:

Definition

The **value function of an action** (or a **state-action pair**) of an MDP $q_\pi(s, a)$ tells us how good it is to perform a given action in a given state.

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \tag{9}$$

- This function calculates the expected return if, starting from state s, we perform action a and follow a policy π from that moment on.

The value functions defined in the equations 8 and 9 can be expressed as a **function of each other**.

In this way, we can express the relationship between $v_\pi(s)$ and $q_\pi(s,a)$:

$$v_\pi(s) = \sum_{\forall a \in \mathcal{A}} \pi(a|s) q_\pi(s,a) \tag{10}$$

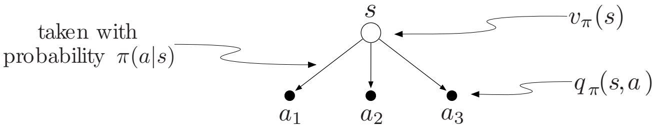
- From this equation, we can extract that the value function of a state $v_\pi(s)$ is an average of the value function of all possible actions $q_\pi(s,a)$ for a given state.

Markov Decision Process

Value functions of an MDP

Backup diagram:

- A way to visualize this relationship.
- Interpretation from the bottom to the top (bottom-up).



Sutton & Barto, 2018

- States are represented by white circles and actions by black circles.
- We want to calculate the value function of the root node $v_\pi(s)$.
- We must consider all the action value functions that are derived from this state ($q_\pi(s, a)$) weighting them by their corresponding policy $\pi(a|s)$.

Markov Decision Process
Value functions of an MDP

Similarly, we can establish the inverse relationship and express $q_\pi(s, a)$ as a function of $v_\pi(s)$:

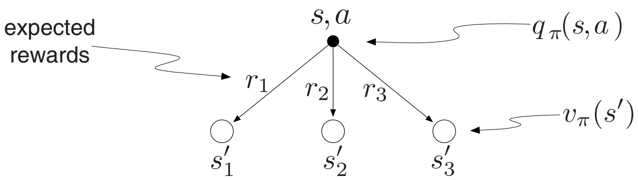
$$\begin{aligned} q_\pi(s, a) &= r(s, a) + \gamma \sum_{\forall s' \in \mathcal{S}} p(s'|s, a) v_\pi(s') \\ &= \sum_{\substack{\forall r \in \mathcal{R} \\ \forall s' \in \mathcal{S}}} p(s', r|s, a)[r + \gamma v_\pi(s')] \end{aligned} \tag{11}$$

- We can interpret the value function of an action $q_\pi(s, a)$ as the sum of the expected immediate reward, obtained
  1. starting from a certain state s and performing a certain action a,
  2. plus the average (weighted by the discount factor γ) of all value functions of all possible immediate successor states s'.

Markov Decision Process
Value functions of an MDP

Backup diagram:

- The **backup diagram** that represents the equation 11:



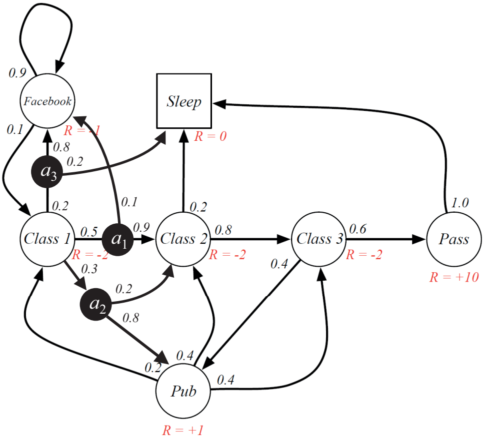
Sutton & Barto, 2018

- To calculate $q_\pi(s, a)$, we must average all the value functions of all possible successor states $v_\pi(s')$ (weighted by the discount factor) plus the corresponding rewards.

Markov Decision Process

Example: Markov decision process of a student's daily life

- **Adding actions** (marked with black circles on the graph).
- The Markov reward process (MRP) becomes a Markov decision process (MDP).



Markov Decision Process

Example: Markov decision process of a student's daily life

- Policy: $\pi(a_1|C1) = 0.5$ or $\pi(a_2|C1) = 0.3$
- Probabilities $p(r, s'|s, a)$: $p(-2, C2|C1, a_1) = 0.9$

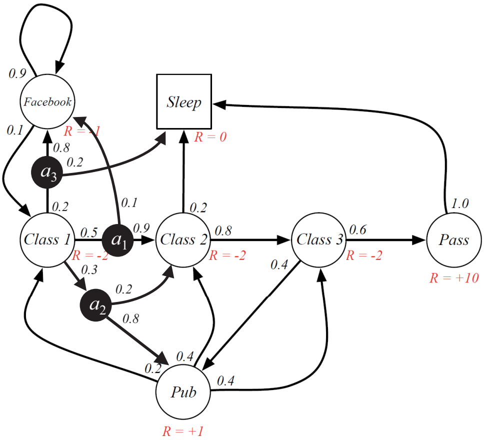


Table of Contents

Bellman equations for an MDP

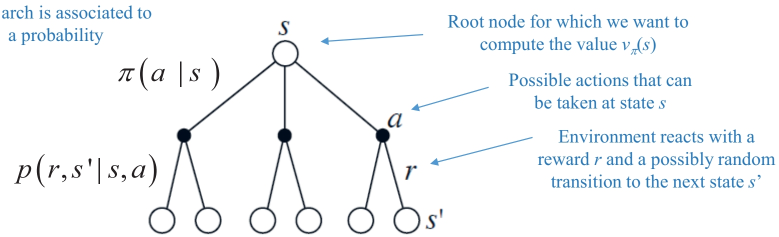
Bellman equation for the state value function v

Bellman equation for the **state value function** v:

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \\
&= \sum_{\forall a \in \mathcal{A}} \pi(a|s) \sum_{\substack{\forall r \in \mathcal{R} \\ \forall s' \in \mathcal{S}}} p(r, s'|s, a)[r + \gamma v_\pi(s')]
\end{aligned} \tag{12}
$$

- We can better understand the formula obtained by looking at the backup diagram.

Bellman equations for an MDP
Bellman equation for the state value function v



Vidal, Cabrera y Giró, 2020

Bellman equations for an MDP

Bellman equation for the state value function v

We can see how the calculation of the **value function** from one state s is propagated to the following states s'.

In this way, for each node we want to calculate the state value function $v_\pi(s)$:

- We select an action using a stochastic policy $\pi(a|s)$.
- For each of the possible actions a of that state, the environment responds with a reward r and a random transition to the next state s' quantified by the probability $p(r, s'|s, a)$.
- If we add all the nodes from bottom to top, weighting them by the aforementioned probabilities, we obtain the **Bellman equation for** $v_\pi(s)$ (equation 12).

Bellman equations for an MDP

Bellman equation for the state value function v

Important!

In an Markov Decision Process (MDP), two forms of randomness appear:

- Randomness due to the environment, reflected in the probabilities $p(r, s'|s, a)$,
- Randomness due to the agent's decision making, expressed through the stochastic policy $\pi(a|s)$.

Bellman equations for an MDP
Bellman equation for the action value function q

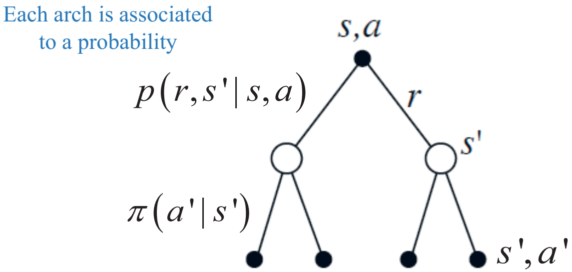
Bellman equation for the **action value function** q:

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\ &= \sum_{\substack{\forall r \in \mathcal{R} \\ \forall s' \in \mathcal{S}}} p(r, s'|s,a)[r + \gamma \sum_{\forall a' \in \mathcal{A}} \pi(a'|s') q_\pi(s', a')] \end{aligned} \tag{13}$$

- We can better understand the formula obtained by looking at the backup diagram.

Bellman equations for an MDP

Bellman equation for the action value function q



Vidal, Cabrera y Giró, 2020

Bellman equations for an MDP

Bellman equation for the state value function v

We can see how the computation of the **value function for a or state-action pair** s, a is propagated to subsequent pairs s', a'.

In this way, for each node that we want to calculate the function $q_\pi(s, a)$:

- The environment responds with a reward r and a random transition to the next state s' quantified by the probability $p(r, s'|s, a)$.
- For each possible successor state s', the agent chooses an action through a stochastic policy $\pi(a'|s')$.
- If we add all the nodes from bottom to top, weighting them by the aforementioned probabilities, we obtain the **Bellman equation for** $q_\pi(s, a)$ (equation 13).

Bellman equations for an MDP

Bellman equations in matrix form

The Bellman equations for MDPs are **linear equations** that can be written in **matrix form**.

$$v_\pi(s) = r_\pi(s) + \gamma \sum_{\forall s' \in \mathcal{S}} p_\pi(s'|s) v_\pi(s') \tag{14}$$

is a linear equation that can be written in matrix form as:

$$\mathbf{v}_\pi = \mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v}_\pi \tag{15}$$

- Where $\mathbf{v}_\pi$ is a column vector containing the value function of all possible states,
- $\mathbf{r}_\pi$ is a column vector whose elements are the expected immediate rewards of each state following the policy π,
- and $\mathbf{P}_\pi$ is the state transition matrix following the policy π, $p_\pi(s'|s)$.

Bellman equations for an MDP

Bellman equations in matrix form

For an **MDP with n states** we obtain:

$$\begin{pmatrix} v_\pi(1) \\ v_\pi(2) \\ \vdots \\ v_\pi(n) \end{pmatrix} = \begin{pmatrix} r_\pi(1) \\ r_\pi(2) \\ \vdots \\ r_\pi(n) \end{pmatrix} + \gamma \begin{pmatrix} p_\pi(1|1) & p_\pi(2|1) & \cdots & p_\pi(n|1) \\ p_\pi(1|2) & p_\pi(2|2) & \cdots & p_\pi(n|2) \\ \vdots & \vdots & & \vdots \\ p_\pi(1|n) & p_\pi(2|n) & \cdots & p_\pi(n|n) \end{pmatrix} \begin{pmatrix} v_\pi(1) \\ v_\pi(2) \\ \vdots \\ v_\pi(n) \end{pmatrix}$$

The previous matrix equation is a linear equation that can be solved:

$$\mathbf{v}_\pi = (\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1} \mathbf{r}_\pi \tag{16}$$

Bellman equations for an MDP

Bellman equations in matrix form

The **inversion of a matrix** ($n \times n$), whose computational complexity is $O(n^3)$, causes that finding a closed solution of the value function $v_\pi(s)$ is only possible if the number of states n is relatively small.

If we want to solve the Bellman equation for an MDP with a high number of states, iterative methods can be used such as:

- **Dynamic Programming**
- **Monte Carlo methods**
- **Temporal-Difference Learning** (TD learning)

Table of Contents

Bellman equations for an MDP

Optimality

Optimal State Value Function

The **optimal state value function** $v_*(s)$ is the state value function $v_\pi(s)$ whose value is the maximum over all possible policies:

$$v_*(s) = \max_\pi v_\pi(s) \tag{17}$$

Optimal Action Value Function

The **optimal action value function** $q_*(s,a)$ is that action value function $q_\pi(s,a)$ whose value is the maximum over all possible policies:

$$q_*(s,a) = \max_\pi q_\pi(s,a) \tag{18}$$

Bellman equations for an MDP
Optimality

How do we compare two policies?

- The solution is to define some type of order between them.
- The order between two policies π and π' is established as follows:

$$\pi \geq \pi' \quad \leftrightarrow \quad v_\pi(s) \geq v_{\pi'}(s), \forall s \tag{19}$$

- The above equation states that a policy π is superior to another policy π' if, for all states s, the state value function following policy π is greater than or equal to the function of state value following policy π'.

Bellman equations for an MDP
Optimality

Fundamental theorem

For any Markov decision process (MDP):

- There is at least one optimal policy π_* that is better than or equal to the rest of the existing policies, $\pi_* \geq \pi, \forall \pi$.
- Every optimal policy produces an optimal state value function, $v_{\pi_*}(s) = v_*(s)$.
- Every optimal policy produces an optimal action value function, $q_{\pi_*}(s, a) = q_*(s, a)$.

Bellman equations for an MDP
Optimality

From the **previous theorem**, we can establish a way to find a **deterministic optimal policy**:

- choose that action a, among all possible ones, that maximizes the optimal action value function $q_*(s, a)$.

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_{a \in \mathcal{A}} q_*(s, a) \\ 0 & \text{other case} \end{cases}$$

- There is always a deterministic optimal policy, which leads us to the best action a for each state s.
- If we know the optimal action value function $q_*(s, a)$, we also know the optimal policy $\pi_*(a|s)$.
- If there is more than one action that maximizes $q_*(s, a)$, we can also obtain a random optimal policy by assigning a non-zero probability value to said actions and zero to the rest.

Bellman equations for an MDP
Optimality

Important!

It may seem that to find the optimal policy a greedy algorithm is applied, since the policy is obtained by searching for the maximum of the immediate actions of the state without taking into account subsequent actions.

However, **this is not the case** because the value function $q_*(s, a)$ already takes into account the agent's actions in subsequent states (remember that it is the expected value of the return).

Bellman equations for an MDP

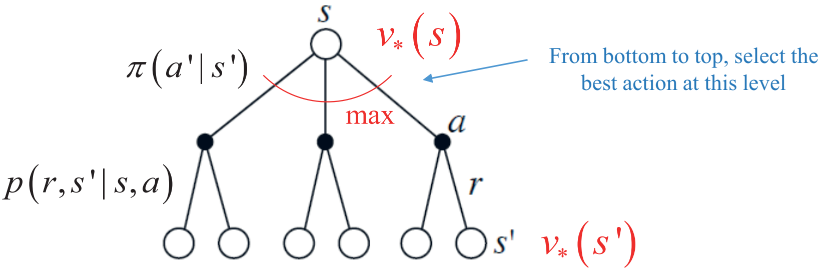
Bellman optimization equations

The equation for the **optimal state value function** $v_*(s)$:

$$
\begin{aligned}
v_*(s) &= \max_a q_*(s,a) \\
&= \max_a \sum_{\substack{\forall r \in \mathcal{R} \\ \forall s' \in \mathcal{S}}} p(r,s'|s,a)[r + \gamma v_*(s')]
\end{aligned}
\tag{20}
$$

Bellman equations for an MDP
Bellman optimization equations

The equation for the **optimal state value function** $v_*(s)$:



Vidal, Cabrera y Giró, 2020

Bellman equations for an MDP

Bellman optimization equations

In the previous diagram, we can see how the calculation of the **optimal state value function** from a state s to the following states s' is carried out.

For each node, we calculate the optimal state value function $v_*(s)$:

- We explore all possible actions a from state s.
- For each of the possible actions a, the environment responds with a reward r and a random transition to the next state s' ($p(r, s'|s, a)$)
- We add all the nodes from bottom to top, weighting them by the aforementioned probabilities.
- By maximizing the weighted sum relative to the possible initial actions, we obtain the **Bellman equation for** $v_*(s)$ (equation 20).

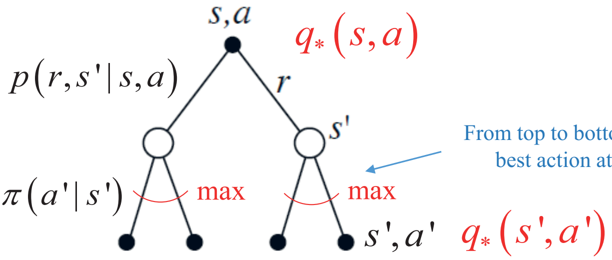
Bellman equations for an MDP
Bellman optimization equations

The equation for the **optimal action value function** $q_*(s, a)$:

$$q_*(s, a) = \sum_{\substack{\forall r \in \mathcal{R} \\ \forall s' \in \mathcal{S}}} p(r, s'|s, a)[r + \gamma \max_{a'} q_*(s', a')] \tag{21}$$

Bellman equations for an MDP
Bellman optimization equations

The equation for the **optimal action value function** $q_*(s, a)$:



Vidal, Cabrera y Giró, 2020

Bellman equations for an MDP

Bellman optimization equations

In the previous diagram, we can see how the calculation of the **optimal action value function** for a state-action pair s, a is propagated to the following pairs s', a'.

For each node, we want to calculate the function $q_*(s, a)$:

- The environment responds with a reward r and a random transition to the next state s' ($p(r, s'|s, a)$).
- For each possible successor state s', the agent must explore all possible successor actions a' and select the one that maximizes the optimal action value function of the new state.
- If we do the bottom-up weighted sum of all maximum nodes, we obtain the **Bellman equation for** $q_*(s, a)$ (equation 21).

Bellman equations for an MDP
Bellman optimization equations

- The **Bellman optimization equations** are nonlinear.
- In general, they do not have a closed form solution.
  - Furthermore, in those cases in which there is a closed solution for these systems of equations, the algorithms to solve them have a high computational cost.
- Thus, we can only find optimal policies in some MDPs with a reduced number of states.

Therefore, many iterative methods have been developed to approximate the **solution of the Bellman optimization equations**:

- Value Iteration
- Policy Iteration
- Q-learning
- Sarsa

Table of Contents

Biliography

References

Some relevant references:

1. **R. S. Sutton, A. G. Barto**. (2018). *Reinforcement Learning: An Introduction (Second edition)*. MIT Press, Cambridge, MA.
2. **M. Lapan**. (2024). *Deep Reinforcement Learning Hands-On (Third Edition)*. Packt Publishing.