Reinforcement Learning

Lesson 3: Finite Markov Decision Processes

Edoardo Fazzari, 2022

Overview

- The Agent-Environment Interface
- Goals and Rewards
- Returns and Episodes
- Unified Notation for Episodic and Continuing Tasks
- Policies and Value Functions
- Optimal Policies and Optimal Value Functions

Finite Markov Decision Processes

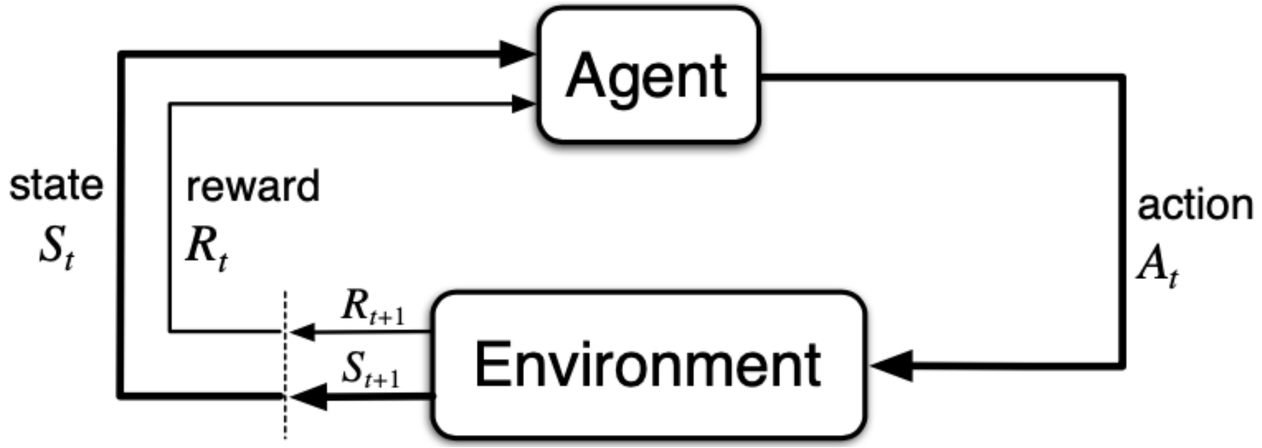
An introduction

- Classical formalization of sequential decision making, where actions influence not just immediate rewards, but also subsequent situations, or states, and through those future rewards.
- MDPs involve delayed reward and the need to trade off immediate and delayed reward
- In MDPs we estimate
    - the value $q_*(s, a)$ of each action *a* in each state *s*
    - or, the value $v_*(s)$ of each state given optimal action selections
- These state-dependent quantities are essential to accurately assigning credit for long-term consequences to individual action selections

The Agent-Environment Interface

Agent-Environment Interface

- The learner and decision maker is called the *agent*
- The thing it interacts with, comprising everything outside the agent, is called the *environment*
- The environment
  - responds to these actions presenting new situations to the agent
  - gives rise to rewards



Agent-Environment Interaction

- The agent and environment interact at each of a sequence of discrete time steps, *t=0,1,2…*
- At each time step *t*, the agent
    - receives some representation of the environment's *state* $S_t \in \mathcal{S}$
    - and on that basis selects an *action* $A_t \in \mathcal{A}(s)$
- One time step later, as a consequence of its action, the agent receives a numerical *reward* $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$
    - And finds itself in a new state S_{t+1}
- This create the sequence (called *trajectory*): $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$

Finite MDP

- In *finite* MDP the sets of states, action, and rewards ($\mathcal{S}$, $\mathcal{A}$, and $\mathcal{R}$) all have a finite number of elements.
    - In this case, the random variables R_t and S_t have well defined discrete probability distributions dependent only on the preceding state and action
- For particular values of these random variables, $s' \in \mathcal{S}$ and $r \in \mathcal{R}$, there is a probability of those values occurring at time *t*, given particular values of the preceding state and action:

$$p(s', r \mid s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\} \qquad (3.2)$$

- For all $s', s \in \mathcal{S}$, $r \in \mathcal{R}$, and $a \in \mathcal{A}(s)$
- p: $\mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0,1]$ and it denotes the *dynamics* of the MDP

Markov Property

- The state must include information about all aspects of the past agent-environment interaction that make a difference for the future
- If it does, then the state is said to have the *Markov property*

State-Transition Probabilities

- From the *dynamic function p,* we can compute the *state-transition probabilities:*

$$p(s'|s,a) \doteq \Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r|s, a) \qquad (3.4)$$

- p: $\mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0,1]$

Expected rewards

Also from *p*

- We can compute the expected rewards for state-action pairs as a two-argument function r: $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$:

$$r(s,a) \doteq \mathbb{E}[R_t \mid S_{t-1}=s, A_{t-1}=a] = \sum_{r\in\mathcal{R}} r \sum_{s'\in\mathcal{S}} p(s', r \mid s, a) \tag{3.5}$$

- And the expected rewards for state-action-next state triples as a three-argument function r: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$:

$$r(s,a,s') \doteq \mathbb{E}[R_t \mid S_{t-1}=s, A_{t-1}=a, S_t=s'] = \sum_{r\in\mathcal{R}} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)} \tag{3.6}$$

Some considerations

Part 1

- The general rule we follow is that anything that cannot be changed arbitrarily by the agent is considered to be outside of it and thus part of its environment

- We do not assume that everything in the environment is unknown to the agent
    - In some cases the agent may know *everything* about how its environment works and still face a difficult RL task (e.g., Rubik's cube)

- The agent-environment boundary represents the limit of the agent's *absolute control*, not of its knowledge

Some considerations

Part 2:

- The MDP framework is a considerable abstraction of the problem of goal-directed learning from interaction
- Any problem of learning goal-directed behavior can be reduced to three signals passing back and forth between an agent and its environment:
    - one signal to represent the choices made by the agent (the actions)
    - one signal to represent the basis on which the choices are made (the states)
    - and one signal to define the agent's goal (the rewards)

- **Anyway!** This framework may not be sufficient to represent all decision-learning problems usefully, but it has proved to be widely useful and applicable

Goals and Rewards

The Reward Hypothesis

Idea

- *All of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward)*

The goal

- The agent always learns to maximize its reward. If we want it to do something for us, we must provide rewards to it in such a way that in maximizing them the agent will also achieve our goals

The reward is not the place to impart to the agent prior knowledge about *how* to achieve what we want it to do

But *WHAT* you want achieved

Returns and Episodes

Expected Return G_t

Defined as some specific function of the reward sequence

- In the simple case the return is the sum of the rewards:

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \ldots + R_T \qquad (3.7)$$

Final time step

- This approach makes sense in applications in which there is a natural notion of final time step:
  - When the agent-environment interaction breaks naturally into subsequences, which we call *episodes*
- Each episode ends in a special state called the *terminal state*, followed by a reset to a standard starting state or to a sample from a standard distribution of starting states

Episodic Tasks

- The next episodes begins independently of how the previous one ended
  - Thus, the episodes can all be considered to end in the same terminal state, with different rewards for the different outcomes
- Task episodes of this kind are called *episodic tasks*
  - In them, we sometimes need to distinguish the set of all nonterminal states, denoted $\mathcal{S}$, from the set of all states plus the terminal state, denoted $\mathcal{S}^+$.
  - The time of termination, T, is a random variable that normally varies from episode to episode

Continuing tasks

- In many cases the agent-environment interaction does not break naturally into identifiable episodes, but goes on continually without limit
    - We call these *continuing tasks*

- The return formulation (3.7) is problematic in this case because the final time step would be $T = \infty$, and the return could easily be infinite

Discounting

The discount rate - Part 1

- According to this approach, the agent tries to select actions so that the sum of the *discounted rewards* it received over the future is maximized
- In particular, it chooses A_t to maximize the expected *discounted return:*

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \qquad (3.8)$$

- Where γ is a parameter, $0 \leq \gamma \leq 1$, called *discount rate*

Discounting

The discount rate - Part 2

- If $\gamma < 1$, the infinite sum in (3.8) has a finite value as long as the reward sequence $\{R_k\}$ is bounded
- If $\gamma = 0$, the agent is *"myopic"* in being concerned only with maximizing immediate rewards (choose A_t to maximize only R_{t+1})
    - In general, acting to maximize immediate reward can reduce access to future rewards so that the return is reduced
- As γ approaches 1, the return objective takes future rewards into account more strongly
    - The agent becomes more farsighted

Discounted Return Reformulation

- Returns at successive time steps are related to each other:

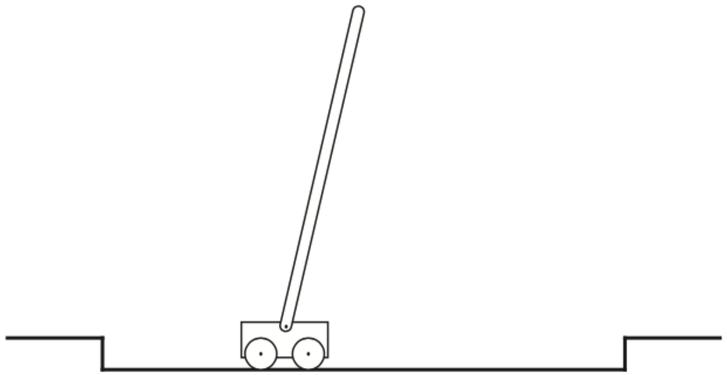
$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots \right) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned} \tag{3.9}$$

- *Note* that this works for all time steps $t < T$, even if termination occurs at *t+1* provided we define $G_T = 0$

Pole-Balancing

Example

- The objective in this task is to apply forces to a cart moving along a track so as to keep a pole hinged to the cart from falling over: A failure is said to occur if the pole falls past a given angle from vertical or if the cart runs off the track. The pole is reset to vertical after each failure. This task could be treated as episodic, where the natural episodes are the repeated attempts to balance the pole. The reward in this case could be +1 for every time step on which failure did not occur, so that the return at each time would be the number of steps until failure. In this case, successful balancing forever would mean a return of infinity. Alternatively, we could treat pole-balancing as a continuing task, using discounting. In this case the reward would be -1 on each failure and zero at all other times. The return at each time would then be related to $-\gamma^{K-1}$, where K is the number of time steps before failure (as well as to the times of later failures). In either case, the return is maximized by keeping the pole balanced for as long as possible.



Unified Notation for Episodic and Continuing Tasks

A problem of notation

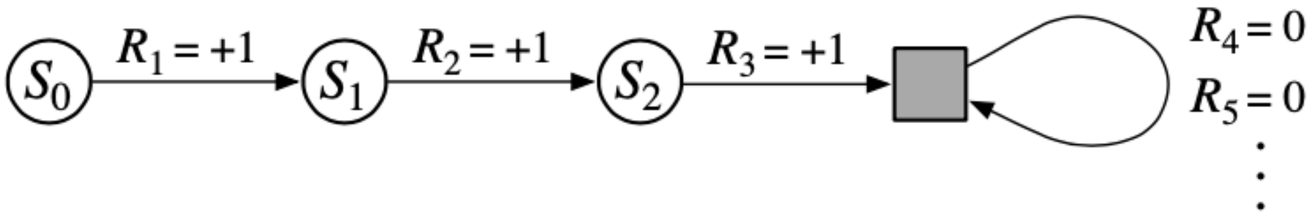
Sometimes we talk about episodic and continuing tasks simultaneously

- Episodic tasks requires some additional notation
    - We need to consider a series of episodes, each of which consists of a finite sequence of time steps
    - We number the time steps of each episode starting anew from zero
    - Therefore, we have to refer to $S_{t,i}$, the state representation at time *t* of episode *i*
    - Similarly for $A_{t,i}$, $R_{t,i}$, $\pi_{t,i}$, etc.
- *However*, when we discuss episodic tasks we almost never have to distinguish between different episodes, thus the explicit reference to the episode number is dropped

The single notation

Part 1

- We have defined the return as a sum over a finite number of terms in one case (3.7) and as a sum over an infinite number of terms in the other (3.8)
    - These two can be unified by considering episode termination to be entering of a special *absorbing state* that transition only to itself and that generates only rewards of zero



The single notation

Part 2

- We can define the return, *in general*, according to (3.8), using the convention of omitting episode numbers when they are not needed and including the possibility that $\gamma = 1$ if the sum remains defined (e.g., because all episode terminate)
- Alternatively, we can write:

$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \qquad (3.11)$$

- Including the possibility that $T = \infty$ or $\gamma = 1$ (*but not both!*)

Policies and Value Functions

Policy

- A *policy* is a mapping from states to probabilities of selecting each possible action.
- If the agent is following policy π at time t, then $\pi(a \mid s)$ is the probability that $A_t = a$ if $S_t = s$

State-Value Function for Policy π

- The *value function* of a state *s* under a policy π, denoted $v_\pi(s)$, is the expected return when starting in *s* and following π thereafter
- For MDPs, we can define

State-value function for policy π

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S} \qquad (3.12)$$

- Where $\mathbb{E}_\pi[]$ denotes the expected value of a random variable given that the agent follows policy π, and *t* is any time step
- *Note* that the value of the terminal state, if any, is always zero

Action-Value Function for Policy π

- We define the value of taking action *a* in state *s* under a policy π as the expected return starting from *s,* taking the action *a*, and thereafter following policy π:

Action-value function for policy π

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t=s, A_t=a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \,\middle|\, S_t=s, A_t=a\right] \quad (3.13)$$

Estimation of v_π and q_π

Can be estimated from experience (Monte Carlo methods)

- If an agent follow policy π and maintains an average, for each state encounter, of the actual returns that have followed that state, then the average will converge to the state's value $v_\pi(s)$ as the number of times that state is encountered approaches infinity
- If separate averages are kept for each action taken in each state, then these averages will similarly converge to the action values $q_\pi(s, a)$
- We call estimation methods of this kind *Monte Carlo methods* because they involve averaging over many random samples of actual returns

Estimation of v_π and q_π

Can be estimated from experience (Approximation methods methods)

- If there are very many states, then it may not be practical to keep separate averages for each state individually
- Instead, the agent would have to maintain v_π and q_π as parameterized functions (with fewer parameters than states) and adjust the parameters to better match the observed returns
- This can also produce accurate estimates, although much depends on the nature of the parameterized function approximator

Bellman Equation

It expresses a relationship between the value of a state and the values of its successor states

- For any policy π and any state s, the following consistency condition holds between the value of s and the value of its possible successor states:

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \,|\, s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big] \qquad (3.14) \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r \,|\, s, a) \Big[r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S}
\end{aligned}
$$

- Where the actions, *a,* are taken form the set $\mathcal{A}(s)$, that the next states, s', are taken form the set $\mathcal{S}$ (or from $\mathcal{S}^+$ in the case of an episodic problem), and that the rewards, *r*, are taken from the set $\mathcal{R}$

Bellman Equation Considerations

- The Bellman equation:
  - averages over all the possibilities weighting each by its probability of occurring
  - States that the value of the start state must equal the (discounted) value of the expected next state, plus the reward expected along the way
- The value function v_π is the unique solution to its Bellman equation
  - *We will see later ways to compute, approximate, and learn it!*

Optimal Policies and Optimal Value Functions

Optimal Policy

For finite MDPs we can precisely define an optimal policy

- Value functions define a partial ordering over policies
    - A policy π is defined to be better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states

$$\pi \geq \pi' \qquad \textit{iff} \qquad v_\pi(s) \geq v'_\pi(s) \quad \forall s \in \mathcal{S}$$

- There is always at least one policy that is better than or equal to all other policies
    - This is an *optimal policy* π_*
    - There can be more than one!

Optimal State-Value and Action-Value Functions

Optimal policies share the same v_* and q_*

- The *optimal state-value function* is defined as:

$$v_*(s) \doteq \max_\pi v_\pi(s) \tag{3.15}$$

- The *optimal action-value function* is defined as:

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a) \tag{3.16}$$

- For all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$

Optimal Action-Value Function

- For the state-action pair (s, a), the function (3.16) gives the expected return for taking action *a* in state *s* and thereafter following an optimal policy
- We can write q_* in terms of v_* as follows:

$$q_*(s, a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \,|\, S_t = s, A_t = a] \qquad (3.17)$$

Bellman Optimality Equation

Part 1

- Because v_* is the value function for a policy, it must satisfy the self-consistency condition given by the Bellman equation for state values
- The *Bellman Optimality Equation* expresses the fact that the value of a state under an optimal policy must equal the expected return for the best action from that state:

$$
\begin{aligned}
v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \qquad (3.18) \\
&= \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma v_*(s')\big]. \qquad (3.19)
\end{aligned}
$$

Bellman Optimality Equation

Part 2

- The last two equations are two forms of the Bellman optimality equation for v_*
- The Bellman optimality equation for q_* is:

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\middle|\; S_t = s, A_t = a\right] \\ &= \sum_{s',r} p(s', r \,|\, s, a)\left[r + \gamma \max_{a'} q_*(s', a')\right] \end{aligned} \tag{3.20}$$

Bellman Optimality Equation

Part 3

- For finite MDPs, the Bellman optimality equation for v_* has a unique solution
- The Bellman optimality equation is a system of equations, one for each state, so if there are *n* states, then there are *n* equations in *n* unknowns
- If the dynamics *p* of the environment are known, then one can solve this system of equations for v_* using any one of a variety of methods for solving systems of nonlinear equations
    - One can solve a related set of equations for q_*

Bellman Optimality Equation

Part 4

- Once one has v_*, it is easy to determine an optimal policy
    - For each state *s*, there will be one or more actions at which the maximum is obtained in the Bellman optimality equation
    - Any policy that assigns nonzero probability only to these actions is an optimal policy
- The beauty of v_* is that
    - If one uses it to evaluate the short-term consequences of actions (the one-step consequences) then a greedy policy is actually optimal in the long-term sense in which we are interested
    - Because v_* already takes into account the reward consequences of all possible future behavior

Bellman Optimality Equation

Part 5

- Having q_* makes choosing optimal actions even easier
- With q_* the agent does not even have to do a one-step-ahead search
    - For any state *s*, it can simply find any action that maximizes $q_*(s, a)$
- The action-value function effectively caches the results of all one-step-ahead searches
- It provides the optimal expected long-term return as a value that is locally and immediately available for each state-action pair
- At the cost of representing a function of state-action pairs the optimal action-value function allows optimal actions to be selected without having to know anything about the environment's dynamic

Bellman Optimality Equation

Part 6

- Explicitly solving the Bellman optimality equation provides one route to finding an optimal policy
- This solution relies on at least three assumptions that are *rarely true in practice*:
    - The dynamics of the environment are accurately known
    - Computational resources are sufficient to complete the calculation
    - The states have the Markov property

It is usually not possible to simple compute an optimal policy by solving the Bellman optimality equation!

Bibliography:

Reinforcement Learning An Introduction (Second Edition), R. S. Sutton & A. G. Barto