Reinforcement Learning

Lectured by Paul Bilokun

Typed by Aris Zhu Yi Qing

October 28, 2022

Contents

1 Markov Concept Definitions

- **Markov Process**: a tuple $(\mathcal{S}, \mathcal{P})$, where:
  - $\mathcal{S}$ – a set of states
  - $\mathcal{P}_{ss'} = P[S_{t+1} = s' | S_t = s]$ – a state transition probability matrix
  - $\sum_{s'} \mathcal{P}_{ss'} = 1$ (to satisfy the probability axiom)
- A state s_t is **Markov** $\iff$ $P[s_{t+1}|s_t] = P[s_{t+1}|s_1, \dots, s_t]$
  - once the state is known, then any data of the history is no longer needed
- **Stationarity (Homogeneous)**: $P[s_{t+1}|s_t]$ doesn't depend on t, but only on the origin and destination states.
- **Markov Reward Process (MRP)**: a tuple $(\mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma)$, where:
  - $\mathcal{S}$ – a set of states
  - $\mathcal{P}_{ss'}$ – a state transition probability matrix
  - $\mathcal{R}_s = \mathbb{E}[r_{t+1}|S_t = s]$ – an expected immediate reward that we collect upon departing state s, whose collection occurs at time step $t+1$
  - $\gamma \in [0, 1]$ – a discount factor
- **Return** (R_t): the total *discounted* reward from time-step t:

$$R_t = r_{t+1} + \gamma r_{t+1} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

- γ close to 0 leads to myopic evaluation
- γ close to 1 leads to far-sighted evaluation
- if T is the time to reach the terminal state, then

$$R_1 = r_2 + \gamma r_3 + \cdots + \gamma^{T-2} r_T.$$

- **State Value Function** $v(s)$ of an MRP: the expected return R starting from state s at time t:

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

- **Bellman Equation** for MRPs:

$$\begin{aligned} v(s) &= \mathbb{E}[R_t | S_t = s] \\ &= \mathbb{E}[r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \cdots) | S_t = s] \\ &= \mathbb{E}[r_{t+1} + \gamma R_{t+1} | S_t = s] \\ &= \mathbb{E}[r_{t+1} + \gamma v(S_{t+1}) | S_t = s] \end{aligned}$$

i.e. $v(s)$ decomposes into

- immediate reward r_{t+1}
- discounted return of successor state $\gamma v(S_{t+1})$.

Alternative forms:

- sum notation: $v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')$
- **vector notation**: $\mathbf{v} = \mathcal{R} + \gamma \mathcal{P} \mathbf{v}$, where $\mathbf{v} \in \mathbb{R}^n$.

Solving the equation in vector notation, we obtain

$$\mathbf{v} = (\mathbb{1} - \gamma \mathcal{P})^{-1} \mathcal{R}$$

while the iterative methods to solve this include:

1. dynamic programming
2. Monte-Carlo evaluation
3. temporal-difference learning

- **Policy**: the conditional probability distribution to execute an action $a \in \mathcal{A}$ given that one is in state $s \in \mathcal{S}$ at time t:

$$\pi_t(a, s) = P[A_t = a | S_t = s]$$

This is considered as **probabilistic** or **stochastic**.

- Policy is **deterministic** if $\pi(a, s) = 1$ and $\pi(a', s) = 0 \ \forall a \neq a'$.
- Alternative notation for deterministic policy: $\pi_t(s) = a$.

Optimal policy (action) maximises expected return.

2 Markov Decision Process

2.1 Definition

- $\mathcal{S}$ – State space
- $\mathcal{A}$ – Action space
- $\mathcal{P}^a_{ss'}$ – transition probability $p(s_{t+1} | s_t, a_t)$
- $\gamma \in [0, 1]$ – discount factor
- $\mathcal{R}^a_{ss'} = r(s, a, s')$ – immediate/instantaneous reward function.
  - temporal notation: $r_{t+1} = r(s_{t+1}, s_t, a_t)$
- π – policy
  - stochastic: $\mathbf{a} \sim p_\pi(\mathbf{a}|\mathbf{s}) \equiv \pi(\mathbf{a}|\mathbf{s}) \equiv \pi(a, s)$
    * each entry of the distribution is $\pi(a_1|\mathbf{s})$ or $\pi(a_1|s)$, depending on whether it is a collection of states of different objects $\mathbf{s}$ or just state of one object s.
  - deterministic: $\mathbf{a} = \pi(\mathbf{s})$

2.2 State Value Function (Bellman Equation)

$$\begin{aligned} V^\pi(s) &= \mathbb{E}[R_t | S_t = s] \\ &= E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = s\right] \\ &= E\left[r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | S_t = s\right] \\ &= \sum_{a \in \mathcal{A}} \pi(a, s) \left\{ \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | S_{t+1} = s' \right] \right) \right\} \end{aligned}$$

$$= \sum_{a\in\mathcal{A}} \pi(a,s) \left\{ \sum_{s'\in\mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V^\pi(s') \right) \right\}$$

where, for instance,

$$\mathbb{E}[r_{t+1}|S_t = s] = \sum_{a\in\mathcal{A}} P[a|s] \left(\sum_{s'\in\mathcal{S}} P[s'|s,a] r(s,a,s') \right).$$

On the other hand, the **Bellman Optimality Equation** for V is

$$V^*(s) = \max_a \sum_{s'} \mathcal{P}^a_{ss'} (\mathcal{R}^a_{ss'} + \gamma V^*(s'))$$

saying that the value of a state under an optimal policy π^* must equal the expected return for the best action from that state.

2.3 Policy Evaluation (Prediction Problem)

- **Iterative** policy evaluation: $V_1(s), V_2(s), \ldots, V_k(s)$
  - Pseudocode:

```
Input $\pi$ the policy to be evaluated
Initialize $V(s) = 0$, $\forall s \in \mathcal{S}^+$
Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V(s')]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number as a threshold)
Output $V \approx V^\pi$
```

2.4 State-Action Value Function (Cost-To-Go)

- Definition:

$$Q^\pi(s,a) = \mathbb{E}[R_t|S_t = s, A_t = a] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = s, A_t = a \right]$$

- relation to State Value Function:

$$V^\pi(s) = \sum_{a\in\mathcal{A}} \pi(s,a) Q^\pi(s,a)$$

- **Bellman Optimality Equation** for Q^π:

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

2.5 Optimal Value Function

- **Optimal State Value Function**:

$$V^*(s) = \max_\pi V^\pi(s),\ \forall s \in \mathcal{S}$$

  The policy π^* that maximises the value function is the **optimal policy**.

- **Optimal State-Action Value Function**:

$$Q^*(s,a) = \max_\pi Q^\pi(s,a),\ \forall s \in \mathcal{S}, a \in \mathcal{A}$$

  which then we also have

$$Q^*(s,a) = \mathbb{E}[r_{t+1} + \gamma V^*(s_{t+1})|S_t = s, A_t = a].$$

2.6 Bellman Optimality Equation Convergence Theorem

For an MDP with a finite state and action space

1. The Bellman (Optimality) equations have a unique solution
2. The values produced by value iteration converge to the solution of the Bellman Equation

3 Dynamic Programming

3.1 Policy Improvement

- Let π and π' be any two deterministic policies s.t. $\forall s \in \mathcal{S}$, $Q^\pi(s, \pi'(s)) \geq V^\pi(s)$,
- then π' must be as good or better than π, i.e. $\forall s \in \mathcal{S}$, $V^{\pi'}(s) \geq V^\pi(s)$.

3.2 Policy Iteration

- Once a policy π has been improved to be π' s.t. $V^{\pi} \leq V^{\pi'}$, and continues, this forms a sequence of monotonically improving policies and value functions.
- This eventually finds the optimal policy π^*, and the process is called policy iteration.
- Pseudocode:

  Policy evaluation: same as above 2.3.

  Policy Improvement:

```
policy-stable ← true
For each $s \in \mathcal{S}$:
  $b \leftarrow \pi(s)$
  $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
  If $b \neq \pi(s)$, then policy-stable ← false
If policy-stable, then stop; else go to evaluation.
```

- What the Policy Improvement step is doing is that it takes the optimal action a' from the current state to the next state s', which forms the new policy π', which in the next iteration find the optimal action a'' forming the new policy π'', etc.

3.3 Bellman's Priciple of Optimality

A policy $\pi(a|s)$ achieves the optimal value from state s, $V^{\pi}(s') = V^*(s')$, iff

- for any state s' reachable from s,
- π achieves the optimal value from state s', $V^{\pi}(s') = V^*(s')$.

3.4 Value Iteration

- Pseudocode:

  Initialize V arbitrarily $\forall s \in \mathcal{S}^+$.

```
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \mathbf{max}_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
```

  Then perform Policy Extraction s.t.

$$\pi(s) = \operatorname*{argmax}_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right].$$

- Policy iteration v.s. Value iteration:
  - **Policy iteration** includes: policy evaluation + policy improvement
  - **Value iteration** includes: finding optimal value function + one policy extraction.
  - Finding optimal value function can be seen as a combination of policy improvement and policy evaluation
  - Policy iteration and finding optimal value function are highly similar except for a max operation.
  - It is more likely that policy iteration is faster than value iteration due to faster convergence.
- This is similar to what we normally understand about dp coding problems, where at each iteration we look for optimal computed value (although most likely we will figure out the optimal policy first).

3.5 Sync/Async backups

- **Synchronous backups**:
  - all states are backed up in parallel,
  - thereby requiring two copies of the value function
- **Asynchronous backups**:
  - not going through all states,
  - or not in any ordering,
  - with any available state/state-action values
  - one copy of value function, in-place update
  - e.g. value iteration with only one state updated per iteration

4 Model-Free Learning

4.1 Monte Carlo (MC) Learning

4.1.1 Introduction

- MC methods learn directly form episodes of experience
- MC is **model-free**: no knowledge of MDP transitions/rewards needed
- MC learns from complete episodes: no bootstrapping, i.e. not updating value estimates based on other value estimates
- main idea: value of staste = mean return
  - an obvious way thus is to estimate from experience – average the returns observed after visits to that state
  - as more returns are observed, the average should converge to the expected value
- can only be applied to **episodic** MDPs that have terminal states

4.1.2 MC Policy Evaluation

- Goal: learn V^π from traces τ of episodes of length T that we experience under policy π, where

$$\tau \equiv s_1, a_1, r_1, s_2, \ldots, s_k.$$

- Pseudocode:

```
$\hat{V}(s) \leftarrow$ arbitrary value, $\forall s \in \mathcal{S}$
Returns($s$) $\leftarrow$ an empty list, $\forall s \in \mathcal{S}$
repeat
  Get trace $\tau$ using $\pi$
  for all $s$ appearing in $\tau$ do
    $R \leftarrow$ return from first appearance of $s$ in $\tau$.
    Append $R$ to Returns($s$)
    $\hat{V}(s) \leftarrow$ average(Returns($s$))
until forever
```