Markov Decision Processes

Christophe Eloy

Outline

- Markov Processes
- Markov decision processes
  - Value functions
  - Bellman equation
- Dynamic programming
  - Value Iteration
  - Policy Iteration

Useful resources

- Course on Reinforcement Learning, David Silver (UCL), also on YouTube
- Book "Reinforcement Learning: An Introduction", Richard Sutton and Andrew Barto (PDF version is free online)

Reinforcement learning

What makes reinforcement learning different from other machine learning paradigms?

- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d data)
- Agent’s actions affect the subsequent data it receives

Examples

- Control of a wind turbine
- Playing a game
- Learn to walk
- Escape a maze
- Manage an investment portfolio

Rewards

- A reward R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximise cumulative reward
- Reinforcement learning is based on the reward hypothesis

Definition (Reward Hypothesis): All goals can be described by the maximisation of expected cumulative reward

Examples of rewards

- Control of a wind turbine: + for power production; - for safety issue
- Playing a game: +/- for winning/loosing a game
- Learn to walk: + for forward motion; - for falling down
- Escape a maze: + for escaping; - for encountering the minotaur
- Manage an investment portfolio: +/- for gaining/loosing money

Sequential decision making

- **Goal:** select actions to maximise total future reward
- Actions may have long term consequences
- Reward may be delayed
- It may be better to sacrifice immediate reward to gain more long-term reward
- Examples: Blocking opponent moves (might help winning chances many moves from now); A financial investment (may take months to mature)

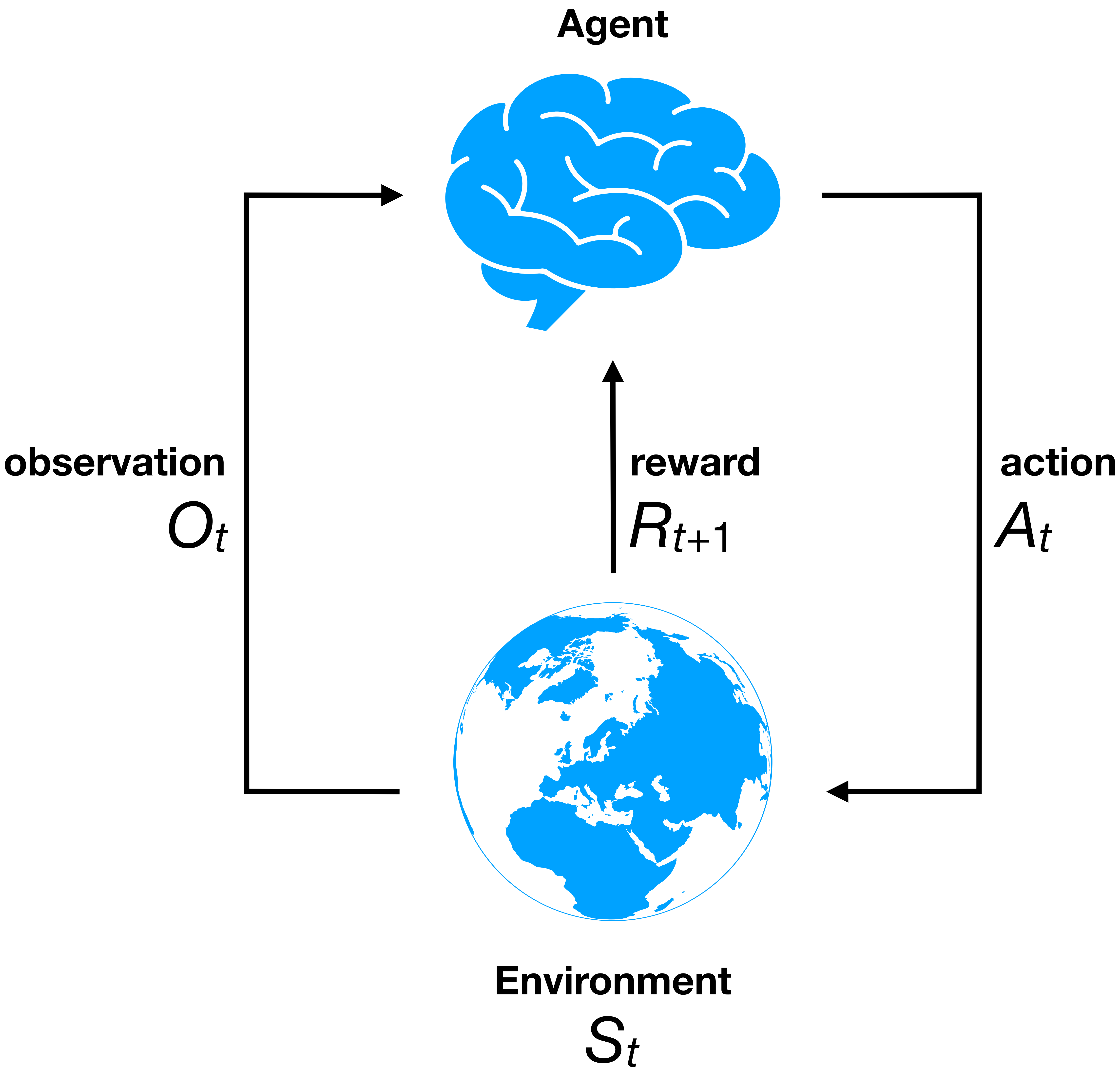
Agent interacts with environment

At each step t

- The agent:
  - Receives observation O_t
  - Receives scalar reward R_{t+1}
  - Executes action A_t

- The environment:
  - Receives action A_t
  - Changes state from S_t to S_{t+1}
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}

Time t increments at each step



Markov Process

- Definition: A Markov process (aka Markov chain) is a memoryless sequence of random states S_1, S_2… with Markov property.
- A Markov process is defined by
  - a (finite) set of states $\mathcal{S}$
  - a transition matrix $\mathcal{P}$
- Markov property

$$\mathbb{P}[S_{t+1} \,|\, S_t] = \mathbb{P}[S_{t+1} \,|\, S_1, \ldots, S_t]$$

- State transition matrix

$$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \,|\, S_t = s\right]$$

Markov reward process

- **Policy** is the function that pick agent's action as a function of its state

$$a = \pi(s) \qquad \text{(deterministic)}$$

$$\pi(a \mid s) = \mathbb{P}\left[A_t = a \mid S_t = s\right] \qquad \text{(stochastic)}$$

- **Value function** is a prediction of future (discounted) rewards

$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s\right]$$

- A **model** predicts what the environment will do next

$$\mathscr{P}^a_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right] \qquad \text{(predicts next state)}$$

$$\mathscr{R}^a_s = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right] \qquad \text{(predicts next reward)}$$

Markov Reward Process

- Finite set of states $\mathcal{S}$
- State transition matrix

$$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \,|\, S_t = s\right]$$

- Reward function

$$\mathcal{R}_s = \mathbb{E}\left[R_{t+1} \,|\, S_t = s\right]$$

- Discount factor: $\gamma \in [0,1]$

Markov Decision Process

- Finite set of states $\mathcal{S}$

- Finite set of actions $\mathcal{A}$

- State transition matrix

$$\mathcal{P}_{ss'}^{a} = \mathbb{P}\left[S_{t+1} = s' \,|\, S_t = s, A_t = a\right]$$

- Reward function

$$\mathcal{R}_{s}^{a} = \mathbb{E}\left[R_{t+1} \,|\, S_t = s, A_t = a\right]$$

- Discount factor: $\gamma \in [0,1]$

Policy and value functions

- Stochastic policy

$$\pi(a \mid s) = \mathbb{P}\left[A_t = a \mid S_t = s\right]$$

- Return: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$

- State-value function

$$v_\pi(s) = \mathbb{E}_\pi\left[G_t \mid S_t = s\right]$$

- Action-value function (Q-function)

$$q_\pi(s, a) = \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right]$$

Bellman equations

- Bellman equation for state-value and action-value functions

$$v_\pi(s) = \sum_a \pi(a \mid s) \left[\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a v_\pi(s') \right]$$

$$q_\pi(s, a) = \mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a \sum_{a'} \pi(a' \mid s') q_\pi(s', a')$$

- Bellman optimality equations ($v_*(s) = \max_a q_*(s, a)$)

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

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

Value Iteration

- Value Iteration algorithm (predicts $v_\pi(s)$ for a given policy $\pi(a \mid s)$)
  - Initialize value vector (for instance, $v_0(s) = 0$ for all states)
  - Iterate the vector v_k using the Bellman equation

$$v_{k+1}(s) := \sum_a \pi(a \mid s) \left[\mathscr{R}_s^a + \gamma \sum_{s'} \mathscr{P}_{ss'}^a v_k(s') \right]$$

  - Stop when convergence is reached: $\|v_{k+1} - v_k\|_\infty < \epsilon$

Policy Iteration

- Policy iteration algorithm (predict the optimal deterministic policy $\pi^*(s)$)
    - Initialize policy (with a random policy for instance)
    - Evaluate the policy with Value Iteration
    - Improve the policy by acting greedily:

$$a = \pi(s) := \text{argmax}_a \left[\mathscr{R}_s^a + \gamma \sum_{s'} \mathscr{P}_{ss'}^a v_k(s') \right]$$

    - Stop when convergence is reached