Introduction to Reinforcement Learning
2/10

Jean Martinet

MSc DSAI

2024 – 2025

Programme

- Introduction
  - Course 1 : Introduction to Reinforcement Learning (RL)
- Part I on tabular methods
  - Course 2 : Markov Decision Processes
  - Course 3 : Dynamic programming in RL
  - Course 4 : Temporal difference 1/2 (Q-learning)
  - Course 5 : Temporal difference 2/2 (SARSA)
- Part II on approximate methods
  - Course 6 : Value function approximation
  - Course 7 : Eligibility traces
  - Course 8 : Policy gradient 1/2 (REINFORCE)
  - Course 9 : Policy gradient 2/2 (actor-critic methods)
  - Course 10 : Projects presentation session

Reminder : think of a project topic

- **Choose from :**
  - Articles/advanced topics/applications
    - Conference paper or book chapter
    - Advanced theme (e.g. actor-critic, eligibility trace, etc.)
    - Application domain (e.g. temperature control, revenue management, etc.)
  - Deepening or exploration project
    - Subject to be chosen/defined and validated
- **Choice to be validated before session 4**
- **Expected result :**
  - Short 2-page max PDF report
  - Code (ipynb / py / git)
  - Short 10-min presentation during last / before last session

About the project

- Double objective
  - Dig deeper in a specific subject (discussed or not during the lectures)
  - Share your insights with other students (in a teacher mode)
- A bit hard to choose early, before having reviewed all topics
- If you can define what is the environment, the reward, the agent, and the actions, it is a good start
- Stay small, at least for a first version, then make it more complex if you have time
- An experimental contribution is needed
  - E.g. compare two algorithms
  - E.g. start from an existing approach, and monitor changes when parameters vary
- The project needs be ORIGINAL
  - You need an original contribution of your own
  - Make sure your project is different from what can be found online
- IMPORTANT : if you decide to use an existing work, it is MANDATORY to cite the source, and you need to state what your contribution is

Mini-quizz

$$V^{\pi}(s_t) = \mathbb{E}_{\pi}[G_t|s_t] = \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t]$$

- What does the value $V^{\pi}(s_t)$ tell us?
  -
- In a deterministic grid world, how many policies are there?
  -
- Is there a best policy?
  -
- Is it unique?
  -
  -

Mini-quizz

$$V^{\pi}(s_t) = \mathbb{E}_{\pi}[G_t | s_t] = \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t]$$

- What does the value $V^{\pi}(s_t)$ tell us ?
  - Expected disc. sum of rewards when starting from s_t and following π
- In a deterministic grid world, how many policies are there ?
  - There are $|A|^{|S|}$ possible policies
- Is there a best policy ?
  - Yes, it is $\pi^* = argmax_{\pi} V^{\pi}(s)$
- Is it unique ?
  - The value V^{π} is unique, but the optimal policy is not necessarily unique
  - (there can be several actions or policies yielding same value)

Today's menu

- Markov environments
- Bellman equations

Markov assumption

Andrey Andreyevich Markov (1856 – 1922) was a Russian mathematician
"The future is independent of the past given the present"

$$p(s_{t+1}|s_t) = p(s_{t+1}|s_1, s_2, ..., s_t)$$

- only the present matters
- the state captures all relevant information from the past (if needed)
- stationary (rules do not change)

Markov Decision Process (MDP)

A Process is a Markov Process if it satisfies the Markovian property
A Markov Reward Process is a MP with a reward at each state
A Markov Decision Process is a MRP with decisions

- Formal description of an environment for decision making / RL
- Tuple $\{S, A, P, R, \gamma\}$
  - **States :** s_t
  - **Action :** a_t
  - **Dynamics model (transitions) :** $P(s_t, a_t, s_{t+1}) \sim p(s_{t+1}|s_t, a_t)$
  - **Reward model :** $R(s_t)$ immediate reward
  - **Discount factor :** γ

- The interaction between the agent and the environment generates a sequence (or trajectory)

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$$

- We consider *finite* MDP (i.e. S, R, and A are finite sets)
  - Discrete variables R_t and S_t have well defined discrete probability distributions dependent only on S_{t-1} and A_{t-1}

$$p(s', r|s, a) = p(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a)$$

- p specifies a probability distribution for each choice of s and a

$$\sum_{s' \in S} \sum_{r \in R} p(s', r|s, a) = 1, \text{ for all } s \in S, a \in A$$

Note about states and observations

- Value function assumes that the environment is completely known
- In real-life, the agent does not *know* the world, but only *observes* it
  - Sensory inputs gives only partial information about the world
- It can also be assumed that the reward is a direct, known function of the observation
- The environmental interaction would becomes

$$A_0, O_1, A_1, O_2, A_2, O_3, A_3, O_4, \ldots$$

- See extension of MDP to *Partially Observable* MDP – POMDP

Examples

- Hypertension control
  - State = Current blood blood pressure
  - Action = Take medication or not
  - Markov ?
- Website shopping
  - State = Current product viewed by the customer
  - Action = What other products are recommended
  - Markov ?
- Note that if you embed the whole history in your state, everything can be Markov

$$H_t \doteq A_0, O_1, A_1, O_2, A_2, O_3, A_3, O_4, \ldots$$

  - However it raises issues : complexity, generalisation, data required

Boundary agent / environment

- Everything that the agent has no absolute control over – not knowledge – is environnement
  - Ex : machinery of a robot
- The boundary is closer to the agent than one could think
- General rule
  - Everything that the agent can not *change* is considered outside of it and thus part of its environment
- The boundary represents the limit of *control*, not *knowledge*
  - In fact, the agent may know everything about its environment
  - But the environment will not be considered a part of the agent, the boundary still exists

Recursive relationship of G_t

- Important property for theory and algorithms of RL
- Remember the *discounted return*

$$G_t \doteq r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \dots$$
$$= \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

- Link between returns at successive time steps

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

Remember from last week

- Remember the **state** value function of s_t under π :

$$
\begin{aligned}
V^{\pi}(s) &\doteq \mathbb{E}_{\pi}[G_t | S_t = s] \\
&= \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = s]
\end{aligned}
$$

- Also remember the **action** value function of taking a_t in s_t under π :

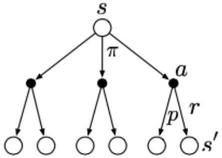
$$
\begin{aligned}
Q^{\pi}(s, a) &\doteq \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a] \\
&= \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = s, A_t = a]
\end{aligned}
$$

- Fundamental property of value functions in RL : they satisfy recursive relationships (like G_t)

$$
\begin{aligned}
V^\pi(s) &\doteq \mathbb{E}_\pi[G_t|S_t = s] \\
&= \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1}|S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s']] \\
&= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V^\pi(s')] \text{ for all } s \in S
\end{aligned}
$$

- This is the *Bellman equation* for $V^\pi(s)$ (relation between the value of s and the value of its successors)

Backup diagram for V^π



- Looking ahead from a state to its possible successor states
  - Open circle= state, solid circle = state–action pair
- From s, the agent could take any of the 3 actions based on π
  - From each of these, the environment could respond with one of several next states (two are shown in the figure), along with a reward, r, depending on its dynamics given by the function p
- Bellman equation : weighted average over all the possibilities
  - It states that the value of the start state must equal the (discounted) value of the expected next state, + the reward expected along the way

Bellman optimality equations

- Value functions define a partial ordering over policies
- We define $\pi \geq \pi'$ iff $V^{\pi}(s) \geq V^{\pi'}(s)$ for all $s \in S$
- Optimal policies are denoted π^*, they share the same V^* and Q^*

$$V^*(s) \doteq \max_{\pi} V^{\pi}(s)$$

$$Q^*(s, a) \doteq \max_{\pi} Q^{\pi}(s, a)$$

Bellman optimality equations for V^* and Q^*

- Bellman optimality equation for V^*
- (without reference to any specific policy)
  - $V^{\pi^*}(s)$ must equal the expected return for the best action from s

$$
\begin{aligned}
V^*(s) &= \max_{a \in A} Q^{\pi^*}(s, a) \\
&= \max_a \mathbb{E}_{\pi^*}[G_t | S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi^*}[r_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\
&= \max_a \mathbb{E}[r_{t+1} + \gamma V^*(S_{t+1}) | S_t = s, A_t = a] \\
&= \max_a \sum_{s', r} p(s', r | s, a)[r + \gamma V^*(s')]
\end{aligned}
$$

- Bellman optimality equation for Q^*

$$
\begin{aligned}
Q^*(s) &= \mathbb{E}[r_{t+1} + \gamma \max_{a'} Q^*(S_{t+1}, a') | S_t = s, A_t = a] \\
&= \sum_{s', r} p(s', r | s, a)[r + \gamma \max a' Q^*(s', a')]
\end{aligned}
$$

Today's lab

- Optional : finish implementing TicTacToe with 1/2 player(s)
  - We'll come back to this later
- Do the exercises in the notebook